APPLIED DIGITAL SIGNAL PROCESSING

Dimitris Manolakis and Vinay Ingle

This page intentionally left blank

Applied Digital Signal Processing

Master the basic concepts and methodologies of digital signal processing with this systematic introduction, without the need for an extensive mathematical background. The authors lead the reader through the fundamental mathematical principles underlying the operation of key signal processing techniques, providing simple arguments and cases rather than detailed general proofs. Coverage of practical implementation, discussion of the limitations of particular methods, and plentiful MATLAB illustrations allow readers to better connect theory and practice. A focus on algorithms that are of theoretical importance or useful in real-world applications ensures that students cover material relevant to engineering practice, and equips students and practitioners alike with the basic principles necessary to apply DSP techniques to a variety of applications. Chapters include worked examples, problems, and computer experiments, helping students to absorb the material they have just read. Lecture slides for all figures and solutions to the numerous problems are available to instructors.

Dimitris G. Manolakis is currently a Member of Technical Staff at MIT Lincoln Laboratory in Lexington, Massachusetts. Prior to this he was a Principal Member of Research Staff at Riverside Research Institute. Since receiving his Ph.D. in Electrical Engineering from the University of Athens in 1981, he has taught at various institutions including Northeastern University, Boston College, and Worcester Polytechnic Institute, and co-authored two textbooks on signal processing. His research experience and interests include the areas of digital signal processing, adaptive filtering, array processing, pattern recognition, remote sensing, and radar systems.

Vinay K. Ingle is currently an Associate Professor in the Department of Electrical and Computer Engineering at Northeastern University, where he has worked since 1981 after receiving his Ph.D. in Electrical and Computer Engineering from Rensselaer Polytechnic Institute. He has taught both undergraduate and graduate courses in many diverse areas including systems, signal/image processing, communications, and control theory, and has co-authored several textbooks on signal processing. He has broad research experience in the areas of signal and image processing, stochastic processes, and estimation theory. Currently he is actively involved in hyperspectral imaging and signal processing.

Applied Digital Signal Processing

THEORY AND PRACTICE

DIMITRIS G. MANOLAKIS

Massachusetts Institute of Technology
Lincoln Laboratory

VINAY K. INGLE

Northeastern University, Boston

CAMBRIDGE
UNIVERSITY PRESS

To my wife and best friend Anna
and in memory of Eugenia, Gregory, and Elias
DGM

To my loving wife Usha and daughters
Natasha and Trupti for their endless support.
VKI

CONTENTS

PREFACE

During the last three decades Digital Signal Processing (DSP) has evolved into a core area of study in electrical and computer engineering. Today, DSP provides the methodology and algorithms for the solution of a continuously growing number of practical problems in scientific, engineering, and multimedia applications.

Despite the existence of a number of excellent textbooks focusing either on the theory of DSP or on the application of DSP algorithms using interactive software packages, we feel there is a strong need for a book bridging the two approaches by combining the best of both worlds. This was our motivation for writing this book, that is, to help students and practicing engineers understand the fundamental mathematical principles underlying the operation of a DSP method, appreciate its practical limitations, and grasp, with sufficient details, its practical implementation.

Objectives

The principal objective of this book is to provide a systematic introduction to the basic concepts and methodologies for digital signal processing, based whenever possible on fundamental principles. A secondary objective is to develop a foundation that can be used by students, researchers, and practicing engineers as the basis for further study and research in this field. To achieve these objectives, we have focused on material that is fundamental and where the scope of application is not limited to the solution of specialized problems, that is, material that has a broad scope of application. Our aim is to help the student develop sufficient intuition as to how a DSP technique works, be able to apply the technique, and be capable of interpreting the results of the application. We believe this approach will also help students to become intelligent users of DSP techniques and good critics of DSP techniques performed by others.

Pedagogical philosophy

Our experience in teaching undergraduate and graduate courses in digital signal processing has reaffirmed the belief that the ideal blend of simplified mathematical analysis and computer-based reasoning and simulations enhances both the teaching and the learning of digital signal processing. To achieve these objectives, we have used mathematics to support underlying intuition rather than as a substitute for it, and we have emphasized practicality without turning the book into a simplistic "cookbook." The purpose of MATLAB® code integrated with the text is to illustrate the implementation of core signal processing algorithms; therefore, we use standard language commands and functions that have remained relatively stable during the most recent releases. We also believe that in-depth

understanding and full appreciation of DSP is not possible without familiarity with the fundamentals of continuous-time signals and systems. To help the reader grasp the full potential of DSP theory and its application to practical problems, which primarily involve continuous-time signals, we have integrated relevant continuous-time background into the text. This material can be quickly reviewed or skipped by readers already exposed to the theory of continuous-time signals and systems. Another advantage of this approach is that some concepts are easier to explain and analyze in continuous-time than in discrete-time or vice versa.

Instructional aids

We have put in a considerable amount of effort to produce instructional aids that enhance both the teaching and learning of DSP. These aids, which constitute an integral part of the textbook, include:

- **Figures** The graphical illustrations in each figure are designed to provide a mental picture of how each method works or to demonstrate the performance of a specific DSP method.
- **Examples** A large number of examples are provided, many generated by MATLAB® to reflect realistic cases, which illustrate important concepts and guide the reader to easily implement various methods.
- MATLAB® **functions and scripts** To help the reader apply the various algorithms and models to real-world problems, we provide MATLAB® functions for all major algorithms along with examples illustrating their use.
- **Learning summaries** At the end of each chapter, these provide a review of the basic yet important concepts discussed in that chapter in the form of a bullet point list.
- **Review questions** Conceptual questions are provided at the end of each chapter to reinforce the theory, clarify important concepts, and help relate theory to applications.
- **Terms and concepts** Important phrases and notions introduced in the chapter are again explained in a concise manner for a quick overview.
- **Problems** A large number of problems, ranging from simple applications of theory and computations to more advanced analysis and design tasks, have been developed for each chapter. These problems are organized in up to four sections. The first set of problems termed as Tutorial Problems contains problems whose solutions are available on the website. The next section, Basic Problems, belongs to problems with answers available on the website. The third section, Assessment Problems, contains problems based on topics discussed in the chapter. Finally, the last section, Review Problems, introduces applications, review, or extension problems.
- **Book website** This website will contain additional in-depth material, signal datasets, MATLAB® functions, power-point slides with all figures in the book, etc., for those who want to delve intensely into topics. This site will be constantly updated. It will also provide tutorials that support readers who need a review of background material.
- **Solutions manual** This manual, which contains solutions for all problems in the text, is available to instructors from the publisher.

Audience and prerequisites

The book is primarily aimed as a textbook for upper-level undergraduate and for first-year graduate students in electrical and computer engineering. However, researchers, engineers, and industry practitioners can use the book to learn how to analyze or process data for scientific or engineering applications. The mathematical complexity has been kept at a level suitable for seniors and first-year graduate students in almost any technical discipline. More specifically, the reader should have a background in calculus, complex numbers and variables, and the basics of linear algebra (vectors, matrices, and their manipulation).

Course configurations

The material covered in this text is intended for teaching to upper-level undergraduate or first-year graduate students. However, it can be used flexibly for the preparation of a number of courses. The first six chapters can be used in a junior level signals and systems course with emphasis on discrete-time. The first 11 chapters can be used in a typical one-semester undergraduate or graduate DSP course in which the first six chapters are reviewed and the remaining five chapters are emphasized. Finally, an advanced graduate level course on modern signal processing can be taught by combining some appropriate material from the first 11 chapters and emphasizing the last four chapters. The pedagogical coverage of the material also lends itself to a well-rounded graduate level course in DSP by choosing selected topics from all chapters.

Feedback

Experience has taught us that errors – typos or just plain mistakes – are an inescapable byproduct of any textbook writing endeavor. We apologize in advance for any errors you may find and we urge you to bring them or additional feedback to our attention at vingle@ece.neu.edu

Acknowledgments

We wish to express our sincere appreciation to the many individuals who have helped us with their constructive comments and suggestions. Special thanks go to Sidi Niu for the preparation of the *Solutions Manual*. Phil Meyler persuaded us to choose Cambridge University Press as our publisher, and we have been happy with that decision. We are grateful to Phil for his enthusiasm and his influence in shaping the scope and the objectives of our book. The fine team at CUP, including Catherine Flack, Chris Miller, and Richard Smith, has made the publication of this book an exciting and pleasant experience. Finally, we express our deepest thanks to our wives, Anna and Usha, for their saintly understanding and patience.

Dimitris G. Manolakis
Vinay K. Ingle

1 Introduction

Signal processing is a discipline concerned with the acquisition, representation, manipulation, and transformation of signals required in a wide range of practical applications. In this chapter, we introduce the concepts of signals, systems, and signal processing. We first discuss different classes of signals, based on their mathematical and physical representations. Then, we focus on continuous-time and discrete-time signals and the systems required for their processing: continuous-time systems, discrete-time systems, and interface systems between these classes of signal. We continue with a discussion of analog signal processing, digital signal processing, and a brief outline of the book.

Study objectives

After studying this chapter you should be able to:

- Understand the concept of signal and explain the differences between continuous-time, discrete-time, and digital signals.
- Explain how the physical representation of signals influences their mathematical representation and vice versa.
- Explain the concepts of continuous-time and discrete-time systems and justify the need for interface systems between the analog and digital worlds.
- Recognize the differences between analog and digital signal processing and explain the key advantages of digital over analog processing.

1.1 Signals

For our purposes a *signal* is defined as any physical quantity that varies as a function of time, space, or any other variable or variables. Signals convey information in their patterns of variation. The manipulation of this information involves the acquisition, storage, transmission, and transformation of signals.

There are many signals that could be used as examples in this section. However, we shall restrict our attention to a few signals that can be used to illustrate several important concepts and they will be useful in later chapters. The speech signal, shown as a *time waveform* in Figure 1.1, represents the variations of acoustic pressure converted into an electric signal by a microphone. We note that different sounds correspond to different patterns of temporal pressure variation.

To better understand the nature of and differences between analog and digital signal processing, we shall use an analog system which is near extinction and probably unknown to many readers. This is the magnetic tape system, used for recording and playback of sounds such as speech or music, shown in Figure 1.2(a). The recording process and playback process, which is the inverse of the recording process, involve the following steps:

- Sound waves are picked up by a microphone and converted to a small analog voltage called the audio signal.
- The audio signal, which varies continuously to "mimic" the volume and frequency of the sound waves, is amplified and then converted to a magnetic field by the recording head.
- As the magnetic tape moves under the head, the intensity of the magnetic field is recorded ("stored") on the tape.
- As the magnetic tape moves under the read head, the magnetic field on the tape is converted to an electrical signal, which is applied to a linear amplifier.
- The output of the amplifier goes to the speaker, which changes the amplified audio signal back to sound waves. The volume of the reproduced sound waves is controlled by the amplifier.
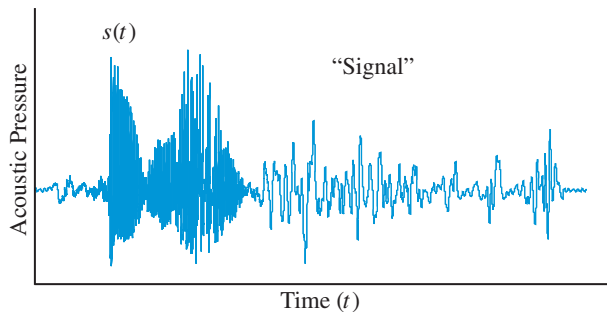
Figure 1.1 Example of a recording of speech. The time waveform shows the variation of acoustic pressure as a function $s(t)$ of time for the word "signal."
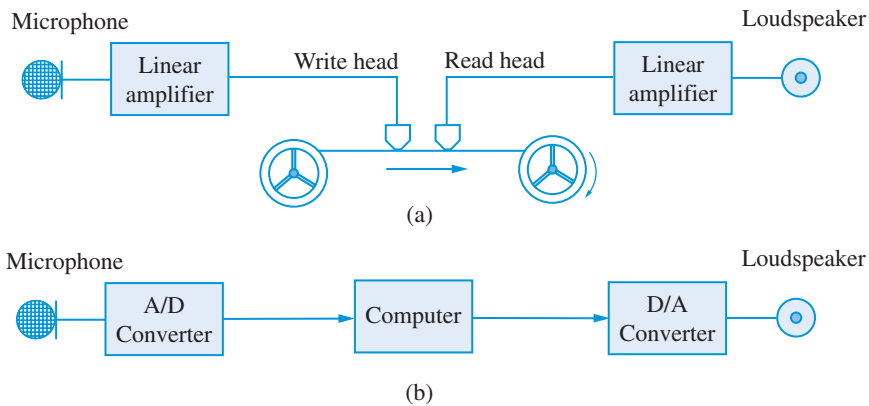
Figure 1.2 Block diagrams of (a) an analog audio recording system using magnetic tape and (b) a digital recording system using a personal computer.

Consider next the system in Figure 1.2(b), which is part of any personal computer. Sound recording and playback with this system involve the following steps:

- The sound waves are converted to an electrical audio signal by the microphone. The audio signal is amplified to a usable level and is applied to an analog-to-digital converter.
- The amplified audio signal is converted into a series of numbers by the analog-to-digital converter.
- The numbers representing the audio signal can be stored or manipulated by software to enhance quality, reduce storage space, or add special effects.
- The digital data are converted into an analog electrical signal; this signal is then amplified and sent to the speaker to produce sound waves.

The major limitation in the quality of the analog tape recorder is imposed by the recording medium, that is, the magnetic tape. As the magnetic tape stretches and shrinks or the speed of the motor driving the tape changes, we have distortions caused by variations in the time scale of the audio signal. Also, random changes in the strength of the magnetic field lead to amplitude distortions of the audio signal. The quality of the recording deteriorates with each additional playback or generation of a copy. In contrast, the quality of the digital audio is determined by the accuracy of numbers produced by the analog-to-digital conversion process. Once the audio signal is converted into digital form, it is possible to achieve error-free storage, transmission, and reproduction. An interesting discussion about preserving information using analog or digital media is given by Bollacker (2010). Every personal computer has a sound card, which can be used to implement the system in Figure 1.2(b); we shall make frequent use of this system to illustrate various signal processing techniques.

1.1.1 Mathematical representation of signals

To simplify the analysis and design of signal processing systems it is almost always necessary to represent signals by mathematical functions of one or more independent variables. For example, the speech signal in Figure 1.1 can be represented mathematically by a function $s(t)$ that shows the variation of acoustic pressure as a function of time. In contrast,
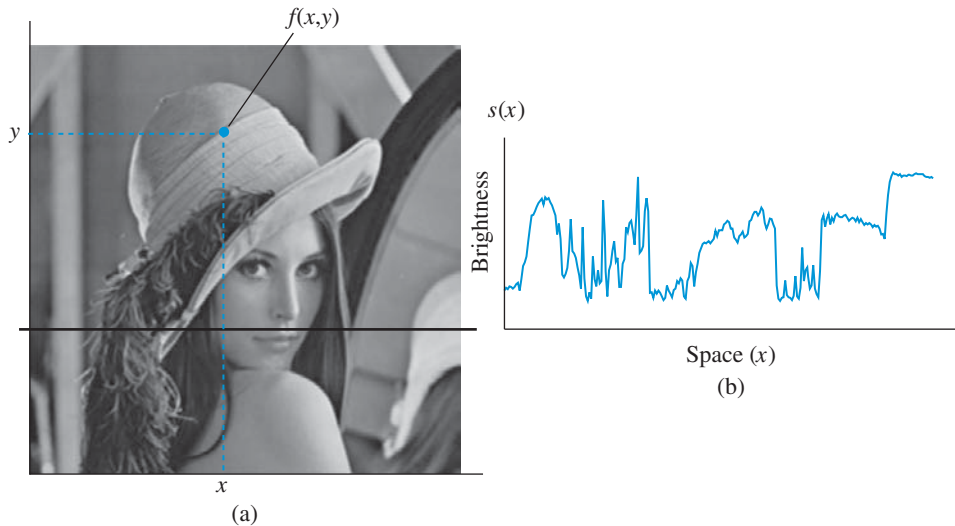
Figure 1.3 Example of a monochrome picture. (a) The brightness at each point in space is a scalar function $f(x, y)$ of the rectangular coordinates x and y. (b) The brightness at a horizontal line at $y = y_0$ is a function $s(x) = f(x, y = y_0)$ of the horizontal space variable x, only.

the monochromatic picture in Figure 1.3 is an example of a signal that carries information encoded in the spatial patterns of brightness variation. Therefore, it can be represented by a function $f(x, y)$ describing the brightness as a function of two spatial variables x and y. However, if we take the values of brightness along a horizontal or vertical line, we obtain a signal involving a single independent variable x or y, respectively. In this book, we focus our attention on signals with a single independent variable. For convenience, we refer to the dependent variable as *amplitude* and the independent variable as *time*. However, it is relatively straightforward to adjust the notation and the vocabulary to accommodate signals that are functions of other independent variables.

Signals can be classified into different categories depending on the values taken by the amplitude (dependent) and time (independent) variables. Two natural categories, that are the subject of this book, are continuous-time signals and discrete-time signals.

The speech signal in Figure 1.1 is an example of a *continuous-time signal* because its value $s(t)$ is defined for every value of time t. In mathematical terms, we say that $s(t)$ is a function of a continuous independent variable. The amplitude of a continuous-time signal may take any value from a continuous range of real numbers. Continuous-time signals are also known as *analog signals* because their amplitude is "analogous" (that is, proportional) to the physical quantity they represent.

The mean yearly number of dark spots visible on the solar disk (sunspots), as illustrated in Figure 1.4, is an example of a discrete-time signal. *Discrete-time signals* are defined only at discrete times, that is, at a discrete set of values of the independent variable. Most signals of practical interest arise as continuous-time signals. However, the use of digital signal processing technology requires a discrete-time signal representation. This is usually done by *sampling* a continuous-time signal at isolated, equally spaced points in time
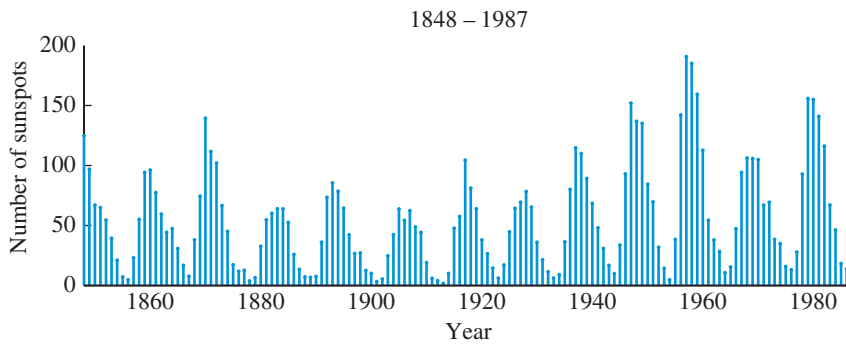
Figure 1.4 Discrete-time signal showing the annual mean sunspot number determined using reliable data collected during the 13 cycles from 1848 to 1987.

(periodic sampling). The result is a sequence of numbers defined by

$$s[n] \triangleq s(t)|_{t=nT} = s(nT), \tag{1.1}$$

where n is an integer $\{\ldots, -1, 0, 1, 2, 3, \ldots\}$ and T is the *sampling period*. The quantity $F_s \triangleq 1/T$, known as *sampling frequency* or *sampling rate*, provides the number of samples per second. The relationship between a continuous-time signal and a discrete-time signal obtained from it by sampling is a subject of great theoretical and practical importance. We emphasize that the value of the discrete-time signal in the interval between two sampling times is not zero; simply, it is *not* defined. Sampling can be extended to two-dimensional signals, like images, by taking samples on a rectangular grid. This is done using the formula $s[m, n] \triangleq s(m\Delta x, n\Delta y)$, where Δx and Δy are the horizontal and vertical sampling periods. The image sample $s[m, n]$ is called a *picture element* or *pixel*, for short.

In this book continuous independent variables are enclosed in parentheses ( ), and discrete-independent variables in square brackets [ ]. The purpose of these notations is to emphasize that parentheses enclose real numbers while square brackets enclose integers; thus, the notation in (1.1) makes sense. Since a discrete-time signal $s[n]$ is a sequence of real numbers, the terms "discrete-time signal" and "sequence" will be used interchangeably. We emphasize that a discrete-time signal $s[n]$ is defined *only* for integer values of the independent variable.

A discrete-time signal $s[n]$ whose amplitude takes values from a finite set of K real numbers $\{a_1, a_2, \ldots, a_K\}$, is known as a *digital signal*. All signals stored on a computer or displayed on a computer screen are digital signals.

To illustrate the difference between the different signal categories, consider the continuous-time signal defined by

$$s(t) = \begin{cases} e^{-2t} \cos(3\pi t), & t \geq 0 \\ 0, & t < 0. \end{cases} \tag{1.2}$$

The continuous-time character of $s(t)$ is depicted graphically using a solid line, as shown in Figure 1.5(a).
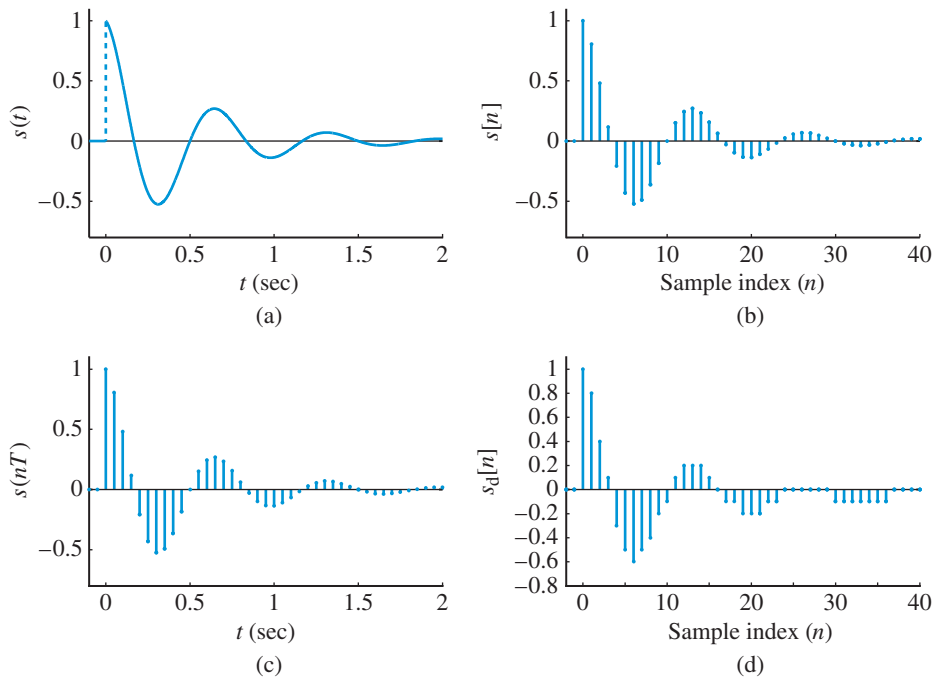
Figure 1.5 Plots illustrating the graphical representation of continuous-time signals (a), discrete-time signals (b) and (c), and digital signals (d).

To plot $s(t)$ on a computer screen, we can only compute its values at a finite set of discrete points. If we sample $s(t)$ with a sampling period $T = 0.1$ s, we obtain the discrete-time signal

$$s[n] = s(nT) = \begin{cases} e^{-0.2n} \cos(0.3\pi n), & n \geq 0 \\ 0, & n < 0 \end{cases} \tag{1.3}$$

which is shown graphically as a stem plot in Figure 1.5(b). Each value of the sequence is represented by a vertical line with a dot at the end (stem). The location of each sample is labeled by the value of the discrete-time index n. If we wish to know the exact time instant $t = nT$ of each sample, we plot $s(nT)$ as a function of t, as illustrated in Figure 1.5(c).

Suppose now that we wish to represent the amplitude of $s[n]$ using only one decimal point. For example, the value $s[2] = 0.4812$ is approximated by $s_d[2] = 0.4$ after truncating the remaining digits. The resulting digital signal $s_d[n]$, see Figure 1.5(d), can only take values from the finite set $\{-0.6, -0.5, \ldots, 1\}$, which includes $K = 17$ distinct signal amplitude levels. All signals processed by computers are digital signals because their amplitudes are represented with finite precision fixed-point or floating-point numbers.

1.1.2 Physical representation of signals

The storage, transmission, and processing of signals require their representation using physical media. There are two basic ways of representing the numerical value of physical quantities: analog and digital:

1. In *analog representation* a quantity is represented by a voltage or current that is *proportional* to the value of that quantity. The key characteristic of analog quantities is that they can vary over a continuous range of values.

2. In *digital representation* a quantity is represented *not* by a proportional voltage or current but by a combination of ON/OFF pulses corresponding to the digits of a binary number. For example, a bit arrangement like $b_1 b_2 \cdots b_{B-1} b_B$ where the B *bi*nary digi*ts* (*bits*) take the values $b_i = 0$ or $b_i = 1$ can be used to represent the value of a binary integer as

$$D = b_1 2^{B-1} + b_2 2^{B-2} + \cdots + b_{B-1} 2^1 + b_B 2^0, \tag{1.4}$$

or the value of a B-bit fraction as

$$D = b_1 2^{-1} + b_2 2^{-2} + \cdots + b_{B-1} 2^{-(B-1)} + b_B 2^{-B}. \tag{1.5}$$

The physical representation of analog signals requires using the physical characteristics of the storage medium to create two "continuous analogies:" one for the signal amplitude, and the other for time. For example, in analog tape recording, time is represented by increasing linear distance along magnetic tape; the amplitude of the original signal is represented by the magnetic field of the tape. In practice, all analog physical representation techniques suffer from two classes of problem: those which affect the "analog of time" (for example, variations in the speed of motor driving the tape), and those which affect the "analog of amplitude" (for example, variations in the magnetic field of the tape). The meaning of analog in this connotation is "continuous" because its amplitude can be varied continuously or in infinitesimally small steps. Theoretically, an analog signal has infinite resolution or, in other words, can represent an uncountably infinite number of values. However, in practice, the accuracy or resolution is limited by the presence of noise.

Binary numbers can be represented by any physical device that has only two operating states or physical conditions. There are numerous devices that satisfy this condition: switch (on or off), diode (conducting or nonconducting), transistor (cut off or saturated), spot on a magnetic disk (magnetized or demagnetized). For example, on a compact disc binary data are encoded in the form of pits in the plastic substrate which are then coated with an aluminum film to make them reflective. The data are detected by a laser beam which tracks the concentric circular lines of pits.

In electronic digital systems, binary information is represented by two nominal voltages (or currents) as illustrated in Figure 1.6. The exact value of the voltage representing the binary 1 and binary 0 is not important as long as it remains within a prescribed range. In a digital signal, the voltage or current level represents no longer the magnitude of a variable, because there are only two levels. Instead, the magnitude of a variable is represented by a combination of several ON/OFF levels, either simultaneously on different lines (parallel transmission) or sequentially in time on one line (serial transmission). As a result, a digital signal has only a finite number of values, and can change only in discrete steps. A digital signal can always provide any desired precision if a sufficient number of bits is provided for each value.

In analog systems, the exact value of the voltage is important because it represents the value of the quantity. Therefore, analog signals are more susceptible to noise (random fluctuations). In contrast, once the value of the data in a digital representation is determined,
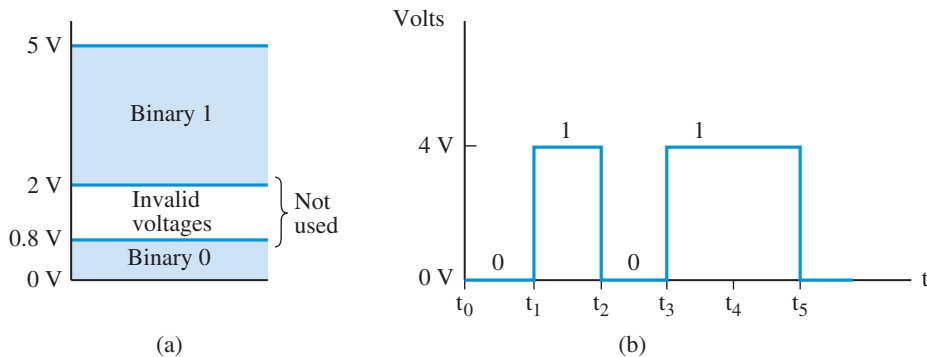
Figure 1.6 Digital signals and timing diagrams. (a) Typical voltage assignments in digital system; (b) typical digital signal timing diagram.

it can be copied, stored, reproduced, or modified without degradation. This is evident if we consider the difference in quality between making a copy of a compact disc and making a copy of an audio cassette.

The digital signals we process and the programs we use to manipulate them are stored as a sequence of bits in the memory of a computer. A typical segment of computer memory might look as follows:

$$\ldots 01101001111010000100101111010101110 \ldots$$

This collection of bits at this level is without structure. The first step in making sense of this bit stream is to consider the bits in aggregates referred to as *bytes* and *words*. Typically, a byte is composed of 8 bits and a word of 16 or 32 bits. Memory organization allows us to access its contents as bytes or words at a particular address. However, we still cannot speak meaningfully of the contents of a byte or word. To give numerical meaning to a given byte, we must know the type of the value being represented. For example, the byte "00110101" has the value 53 if treated as integer or the value 0.2070 if treated as a fraction. Each computer language has different types of integer and floating representations of numbers. Different types of number representation and their properties are discussed in Chapter 15. We shall use the term *binary code* to refer to the contents of a byte or word or its physical representation by electronic circuits or other physical media.

1.1.3 Deterministic and random signals

The distinction between continuous-time signals and discrete-time signals has important implications in the mathematical tools used for their representation and analysis. However, a more profound implication stems from the distinction between deterministic signals and random signals. The behavior of deterministic signals is completely predictable, whereas the behavior of random signals has some degree of uncertainty associated with them. To make this distinction more precise, suppose that we know all past values of a signal up to the present time. If, by using the past values, we can predict the future values of the signal exactly, we say that the signal is *deterministic*. On the other hand, if we cannot predict the future values of the signal exactly, we say that the signal is *random*. In practice, the distinction between these two types of signal is not sharp because every signal

is corrupted by some amount of unwanted random noise. Nevertheless, the separation into deterministic and random signals has been widely adopted when we study the mathematical representation of signals.

Deterministic signals can be described, at least in principle, by mathematical functions. These functions can often take the form of explicit mathematical formulas, as for the signals shown in Figure 1.5. However, there are deterministic signals that cannot be described by simple equations. In principle, we assume that each deterministic signal is described by a function $s(t)$, even if an explicit mathematical formula is unavailable. In contrast, random signals cannot be described by mathematical functions because their future values are unknown. Therefore, the mathematical tools for representation and analysis of random signals are different from those used for deterministic signals. More specifically, random signals are studied using concepts and techniques from the theory of probability and statistics. In this book, we mainly focus on the treatment of deterministic signals; however, a brief introduction to the mathematical description and analysis of random signals is provided in Chapters 13 and 14.

1.2   Systems

In Merriam-Webster's dictionary, a system is broadly defined as a "regularly interacting or interdependent group of items forming a unified whole." In the context of signal processing, a *system* is defined as a process where a signal called *input* is transformed into another signal called *output*. Systems are classified based on the category of input and output signals.

1.2.1   Continuous-time systems

A *continuous-time system* is a system which transforms a continuous-time input signal $x(t)$ into a continuous-time output signal $y(t)$. For example, the continuous-time system described by the formula

$$y(t) = \int_{-\infty}^{t} x(\tau)\mathrm{d}\tau \tag{1.6}$$

produces an output signal which is the integral of the input signal from the start of its operation at $t = -\infty$ to the present time instant t. Symbolically, the input-output relation of a continuous-time system is represented by

$$x(t) \xmapsto{\mathcal{H}} y(t) \quad \text{or} \quad y(t) = \mathcal{H}\{x(t)\}, \tag{1.7}$$

where $\mathcal{H}$ denotes the mathematical operator characterizing the system. A pictorial representation of a continuous-time system is shown in Figure 1.7(a).

Continuous-time systems are physically implemented using analog electronic circuits, like resistors, capacitors, inductors, and operational amplifiers. The physical implementation of a continuous-time system is known as an *analog system*. Some common analog systems are audio amplifiers, AM/FM receivers, and magnetic tape recording and playback systems.
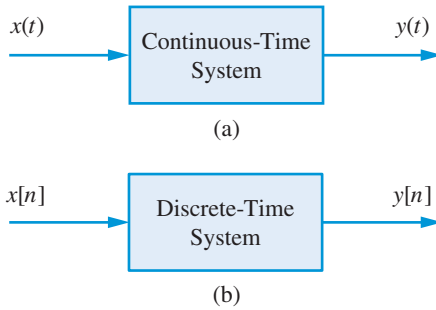
Figure 1.7 Pictorial or block-diagram representation of a continuous-time system (a) and a discrete-time system (b).

1.2.2 Discrete-time systems

A system that transforms a discrete-time input signal $x[n]$ into a discrete-time output signal $y[n]$, is called a *discrete-time system*. A pictorial representation of a discrete-time system, denoted symbolically by

$$x[n] \xmapsto{\mathcal{H}} y[n] \quad \text{or} \quad y[n] = \mathcal{H}\{x[n]\}, \tag{1.8}$$

is shown in Figure 1.7(b). The discrete-time equivalent of the continuous-time integrator system (1.6) is the accumulator system

$$y[n] = \sum_{k=-\infty}^{n} x[k]. \tag{1.9}$$

We note that the integral in (1.6), which is an operator applicable to continuous functions, is replaced by summation, which is a discrete operation.

The physical implementation of discrete-time systems can be done either in software or hardware. In both cases, the underlying physical systems consist of digital electronic circuits designed to manipulate logical information or physical quantities represented in digital form by binary electronic signals. Numerical quantities represented in digital form can take on only discrete values, or equivalently are described with finite precision. Therefore, in practice every discrete-time system has to be implemented by a *digital system*. The term digital is derived from the way computers perform operations, by counting digits.

1.2.3 Interface systems

An analog system contains devices that manipulate physical quantities that are represented in analog form. In an analog system, the amplitude of signals can vary over a continuous range of values. In contrast, a digital system is a combination of devices designed to manipulate physical quantities that are represented in digital form using logical operations. Therefore, there is a need for systems that provide the interface between analog and digital signals.

(a)

(b)
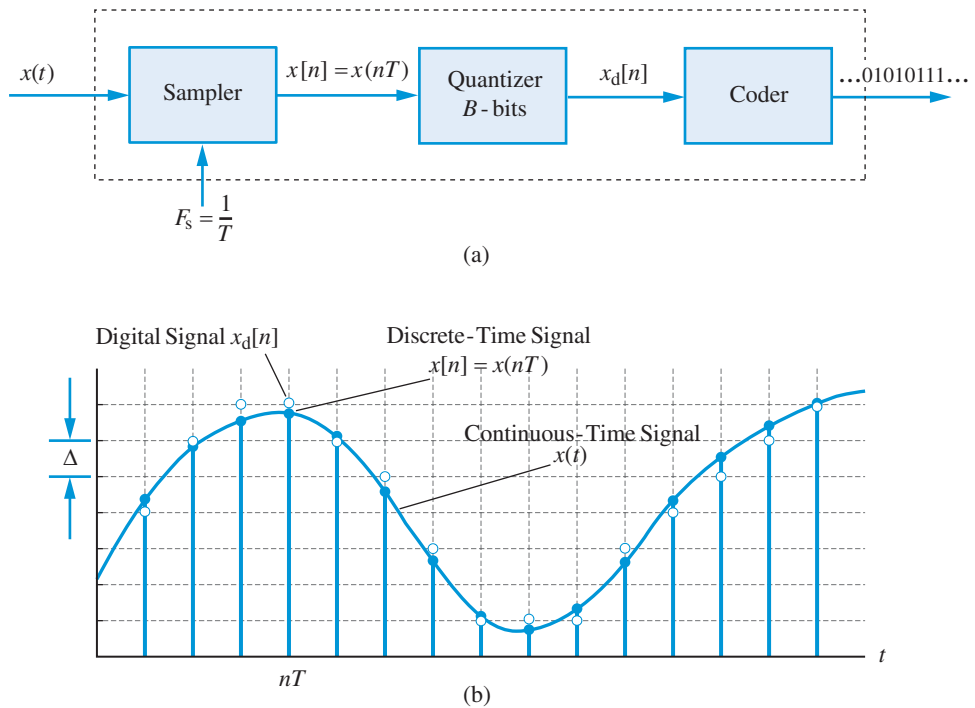
Figure 1.8 (a) Block diagram representation of the analog-to-digital conversion process. (b) Examples of the signals $x(t)$, $x[n]$, and $x_d[n]$ involved in the process. The amplitude of $x[n]$ is known with infinite precision, whereas the amplitude of $x_d[n]$ is known with finite precision Δ (quantization step or resolution).

Analog-to-digital conversion Conceptually, the conversion of an analog (continuous-time, continuous-amplitude) signal into a digital (discrete-time, discrete-amplitude) signal, is a simple process; it consists of two parts: sampling and quantization. *Sampling* converts a continuous-time signal to a discrete-time signal by measuring the signal value at regular intervals of time. *Quantization* converts a continuous-amplitude x into a discrete-amplitude x_d. The result is a digital signal that is different from the discrete-time signal by the quantization error or noise. These operations are implemented using the system illustrated in Figure 1.8. In theory, we are dealing with discrete-time signals; in practice, we are dealing with digital signals.

A practical A/D converter (ADC) accepts as input an analog signal A and analog reference R and after a certain amount of time (conversion time) provides as output a digital signal D such that

$$A \approx RD = R(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_B 2^{-B}). \tag{1.10}$$

The output of ADC is a digital word (ON/OFF signal) representing the B-bit number $b_1 b_2 \cdots b_B$. The value D is the closest approximation to the ratio A/R within the resolution $\Delta = 2^{-B}$. This process is repeated at each sampling interval. To obtain an accurate conversion, the input signals are often switched into an analog storage circuit and held constant during the time of the conversion (acquisition time) using a sample-and-hold circuit.

Continuous-time signal

$x(t)$

Ideal Analog-to-Digital Converter

Discrete-time signal

$x[n]$

(a)

Analog input

Analog-to-Digital Converter

Digital output

0 1 0 0 0 1 0 1 0 1 1 0

(b)
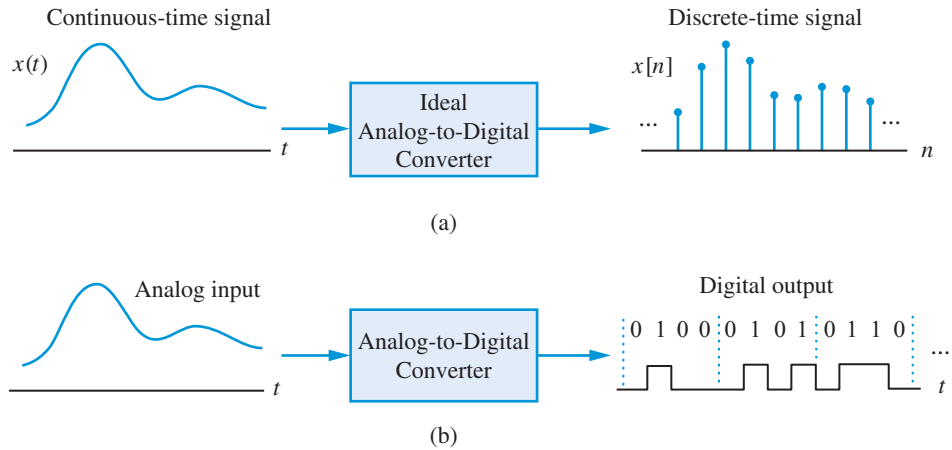
Figure 1.9 Block diagram representation of an ideal (a) and a practical (b) analog-to-digital converter, and the corresponding input and output signals. The input to the ideal ADC is a function and the output is a sequence of numbers; the input to the practical ADC is an analog signal and the output is a sequence of binary code words. The number of bits B, in each word, determines the accuracy of the converter.

As the number of bits B increases, the accuracy Δ of the quantizer increases, and the difference between discrete-time and digital signals diminishes. For this reason, we usually refer to the sampler as an *ideal analog-to-digital (A/D) converter*. Ideal and practical A/D converters and the corresponding input and output signals are illustrated in Figure 1.9.

Digital-to-analog conversion The conversion of a discrete-time signal into continuous time form is done with an interface system called *digital-to-analog* (D/A) *converter* (DAC). The *ideal D/A converter* or *interpolator* is essentially filling the gaps between the samples of a sequence of numbers to create a continuous-time function (see Figure 1.10(a)). A practical DAC takes a value represented in digital code and converts it to a voltage or current that is proportional to the digital value. More specifically, a D/A converter accepts a digital code D and an analog reference R as inputs, and generates an analog value $\hat{A} = RD$ as output. For example, if the digital signal D represents a fractional binary number, as in (1.5), then the output of the D/A converter is

$$\hat{A} = R(b_1 2^{-1} + b_2 2^{-2} + \cdots + b_B 2^{-B}) \approx A. \qquad (1.11)$$

This process is repeated at each sampling interval. Most practical D/A converters convert the binary input to the corresponding analog level and then hold that value until the next sample producing a staircase waveform (see Figure 1.10(b)). This staircase output is subsequently smoothed using an analog filter.

Summary Based on the type of input and output signal, there are three classes of practical system: analog systems, digital systems, and analog-digital interface systems. From a hardware point of view, A/D and D/A converters are interface systems that link the analog (physical) world to the domain of discrete numbers and computers. Quantization of analog
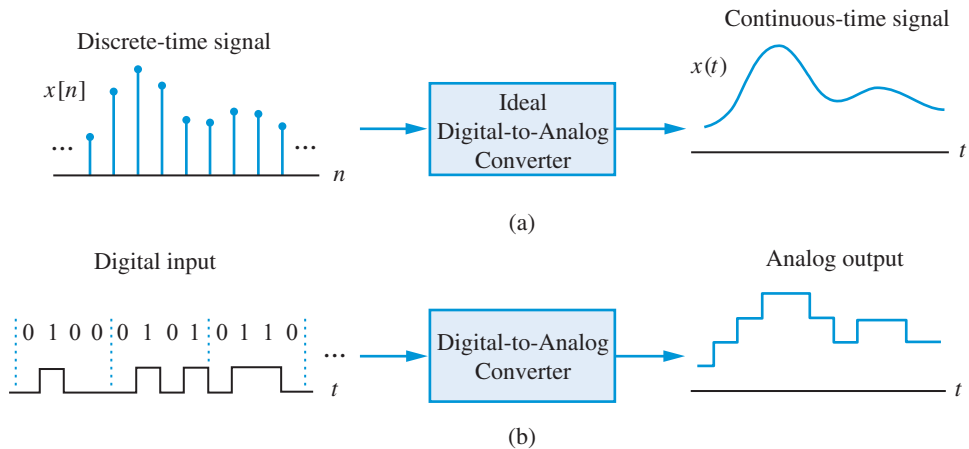
Figure 1.10 Block diagram representation of an ideal D/A converter (a) and a practical D/A converter (b) with the corresponding input and output signals. In most practical applications, the staircase output of the D/A converter is subsequently smoothed using an analog reconstruction filter.

quantities is a nonlinear operation which complicates the analysis and design of digital signal processing systems. The usual practice, which we adopt in this book, is to deliberately ignore the effects of quantization. Thus, the entire book (except Chapter 15) deals with continuous-time systems, discrete-time systems, and ideal A/D and D/A converters; the effects of quantization are considered separately and are taken into account later, if necessary. The effects of quantization on discrete-time signals and systems are discussed in Chapter 15.

The different types of system are summarized in Figure 1.11. We emphasize that each class of system differs in terms of physical implementation, mathematical representation, and the type of mathematics required for its analysis. Although in this book we focus on discrete-time systems, we review continuous-time systems when it is necessary. Chapters 6 and 15 provide a thorough treatment of sampling, quantization, and analog-digital interface systems.

1.3   Analog, digital, and mixed signal processing

Signal processing is a discipline concerned with the acquisition, representation, manipulation, and transformation of signals. Signal processing involves the physical realization of mathematical operations and it is essential for a tremendous number of practical applications. Some key objectives of signal processing are to improve the quality of a signal or extract useful information from a signal, to separate previously combined signals, and to prepare signals for storage and transmission.

Analog signal processing Since most physical quantities are nonelectric, they should first be converted into an electric signal to allow electronic processing. *Analog Signal*
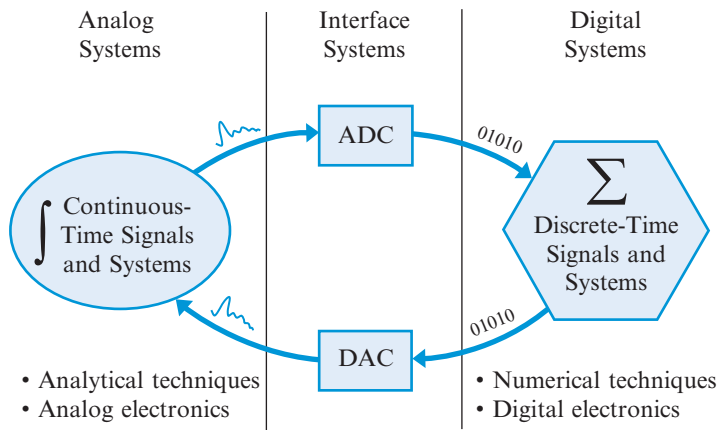
Figure 1.11 The three classes of system: analog systems, digital systems, and interface systems from analog-to-digital and digital-to-analog.
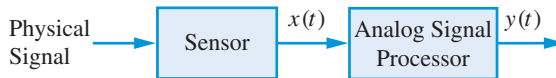
Figure 1.12 Simplified block diagram of an analog signal processing system.

Processing (ASP) is concerned with the conversion of analog signals into electrical signals by special transducers or sensors and their processing by analog electrical and electronic circuits. The output of the sensor requires some form of conditioning, usually amplification, before it can be processed by the analog signal processor. The parts of a typical analog signal processing system are illustrated in Figure 1.12.

Digital signal processing The rapid evolution of digital computing technology which started in the 1960s, marked the transition from analog to digital signal processing. *Digital Signal Processing (DSP)* is concerned with the representation of analog signals by sequences of numbers, the processing of these sequences by numerical computation techniques, and the conversion of such sequences into analog signals. Digital signal processing has evolved through parallel advances in signal processing theory and the technology that allows its practical application.

In theory, where we concentrate on the essential mathematical aspects of signal processing, we deal with ideal (infinite precision) discrete-time signal processing systems, and ideal A/D and D/A converters. A typical system for discrete-time processing of continuous-time signals is shown in Figure 1.13(a).

In practice, due to inherent real-world limitations, a typical system for the digital processing of analog signals includes the following parts (see Figure 1.13(b)):

1. A sensor that converts the physical quantity to an electrical variable. The output of the sensor is subject to some form of conditioning, usually amplification, so that the voltage of the signal is within the voltage sensitivity range of the converter.
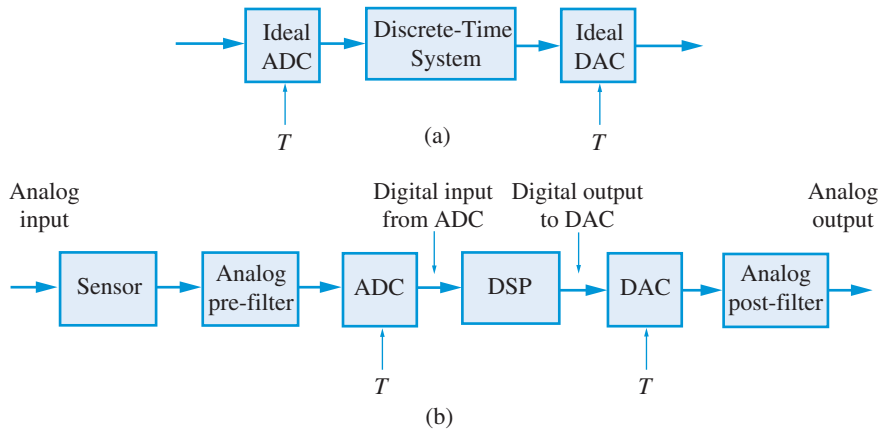
Figure 1.13 Simplified block diagram of idealized system for (a) continuous-time processing of discrete-time signals, and (b) its practical counterpart for digital processing of analog signals.

2. An analog filter (known as pre-filter or antialiasing filter) used to "smooth" the input signal before sampling to avoid a serious sampling artifact known as aliasing distortion (see Chapter 6).

3. An A/D converter that converts the analog signal to a digital signal. After the samples of a discrete-time signal have been stored in memory, time-scale information is lost. The sampling rate and the number of bits used by the ADC determine the accuracy of the system.

4. A digital signal processor (DSP) that executes the signal processing algorithms. The DSP is a computer chip that is similar in many ways to the microprocessor used in personal computers. A DSP is, however, designed to perform certain numerical computations extremely fast. Discrete-time systems can be implemented in real-time or off-line, but ADC and DAC always operate in real-time. *Real-time* means completing the processing within the allowable or available time between samples.

5. A D/A converter that converts the digital signal to an analog signal. The DAC, which reintroduces the lost time-scale information, is usually followed by a sample-and-hold circuit. Usually, the A/D and D/A converters operate at the same sampling rate.

6. An analog filter (known as reconstruction or anti-imaging filter) used to smooth the staircase output of the DAC to provide a more faithful analog reproduction of the digital signal (see Chapter 6).

We note that the DAC is required only if the DSP output must be converted back into an analog signal. There are many applications, like speech recognition, where the results of processing remain in digital form. Alternatively, there are applications, such as CD players, which do not require an ADC.

The fundamental distinction between digital signal processing and discrete-time signal processing, is that the samples of digital signals are described and manipulated with finite numerical accuracy. Because the discrete nature of signal amplitudes complicates the analysis, the usual practice is to deal with discrete-time signals and then to consider the

effects of the discrete amplitude as a separate issue. However, as the accuracy of number representation and numerical computations increases this distinction is blurred and the discrete-time nature of signals becomes the dominant factor. In this book, we focus on discrete-time signal processing; finite accuracy effects are discussed in Chapter 15.

Digital signal processing has many advantages compared to analog signal processing. The most important are summarized in the following list:

1. Sophisticated signal processing functions can be implemented in a cost-effective way using digital techniques.
2. There exist important signal processing techniques that are difficult or impossible to implement using analog electronics.
3. Digital systems are inherently more reliable, more compact, and less sensitive to environmental conditions and component aging than analog systems.
4. The digital approach allows the possibility of time-sharing a single processing unit among a number of different signal processing functions.

Application of digital signal processing to the solution of real-world problems requires more than knowledge of signal processing theory. Knowledge of hardware, including computers or digital signal processors, programming in C or MATLAB, A/D and D/A converters, analog filters, and sensor technology are also very important.

Mixed-signal processing The term *mixed-signal processing* is sometimes used to describe a system which includes both analog and digital signal processing parts. Although, strictly speaking, the system in Figure 1.13(b) is a mixed-processing system, we often use this term to emphasize that both analog and digital components are implemented on the same integrated circuit. Once we have decided to use DSP techniques, the critical question is how close to the sensor to put the ADC. Given the existing technology trends, the objective is to move the ADC closer to the sensor, and replace as many analog operations before the ADC with digital operations after the ADC. Indeed, with the development of faster and less expensive A/D converters, more and more of the analog front end of radar and communication systems is replaced by digital signal processing, by moving the ADC closer to the antenna.

1.4 Applications of digital signal processing

Digital signal processing has an extremely diverse range of applications, from consumer electronics to radar systems. A look at the list in Table 1.1, which is by no means complete, shows the importance of digital signal processing technology in real-world applications.

In terms of computational requirements, digital signal processing applications can be classified in three major classes: (a) low-cost high-volume embedded systems, for example, modems and cellular phones, (b) computer-based multimedia, for example, modems, audio and video compression and decompression, and music synthesis, and (c) high-performance applications involving processing large volumes of data with complex algorithms, for example, radar, sonar, seismic imaging, hyperspectral imaging, and speech recognition. The first two classes rely on inexpensive digital signal processors, whereas the third

Table 1.1 Examples of digital signal processing applications and algorithms.

Application area	DSP algorithm
Key operations	convolution, correlation, filtering, finite discrete transforms, modulation, spectral analysis, adaptive filtering
Audio processing	compression and decompression, equalization, mixing and editing, artificial reverberation, sound synthesis, stereo and surround sound, and noise cancelation
Speech processing	speech synthesis, compression and decompression, speech recognition, speaker identification, and speech enhancement
Image and video processing	image compression and decompression, image enhancement, geometric transformations, feature extraction, video coding, motion detection, and tomographic image reconstruction
Telecommunications (transmission of audio, video, and data)	modulation and demodulation, error detection and correction coding, encryption and decryption, acoustic echo cancelation, multipath equalization, computer networks, radio and television, and cellular telephony
Computer systems	sound and video processing, disk control, printer control, modems, internet phone, radio, and television
Military systems	guidance and navigation, beamforming, radar and sonar processing, hyperspectral image processing, and software radio

class requires processors with maximum performance, ease of use, user-friendly software development tools, and support for multiprocessor configurations.

Instead of listing more applications, we discuss in more detail how a digital signal processor is used in a digital cellular telephone. Figure 1.14 shows a simplified block diagram of a digital cell phone. The audio signal from the microphone is amplified, filtered, converted to digital form by the ADC, and then goes to the DSP for processing. From the DSP, the digital signal goes to the RF (radio-frequency) unit where it is modulated and prepared for transmission by the antenna. Incoming RF signals containing voice data are picked up by the antenna, demodulated, and converted to digital form. After processing by the DSP, the modified digital signal is converted back to the original audio signal by the DAC, filtered, amplified, and applied to the speaker. The DSP processor performs several functions, including: speech compression and decompression, error detection and correction, encryption, multipath equalization, signal strength and quality measurements, modulation and demodulation, co-channel interference cancelation, and power management. We will have the opportunity to progressively introduce specific digital signal processing algorithms, for several of these functions, concurrently with the development of the underlying theoretical concepts and mathematical tools. We emphasize that, despite the overwhelming number of applications, there is a fundamental set of theoretical DSP tools and operations that are used repeatedly to address the majority of practical problems.
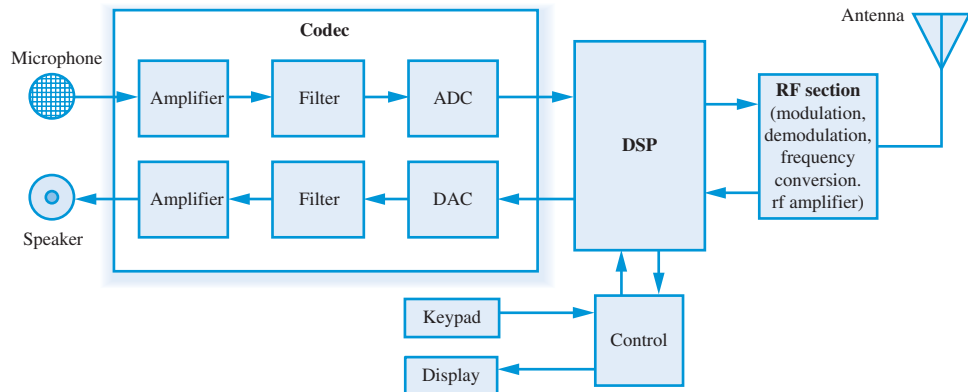
Figure 1.14 Simplified block diagram of a digital cellular phone.

1.5   Book organization

Chapter 1 Introduction Chapter 1 (this chapter) provides an introduction to the concepts of signals, systems, and signal processing in both the continuous-time and discrete-time domains. The topics of analog and digital signals, analog and digital systems, and analog-digital interface systems are also discussed.

Chapter 2 Discrete-time signals and systems The subject of Chapter 2 is the mathematical properties and analysis of linear time-invariant systems with emphasis on the convolution representation. A detailed discussion of the software implementation of convolution and difference equations is also provided.

Chapter 3 The z-transform Chapter 3 introduces the z-transform of a sequence and shows how the properties of the sequence are related to the properties of its z-transform. The z-transform facilitates the representation and analysis of LTI systems using the powerful concepts of system function, poles, and zeros.

Chapter 4 Fourier representation of signals All signals of practical interest can be expressed as a superposition of sinusoidal components (Fourier representation). Chapter 4 introduces the mathematical tools, Fourier series and Fourier transforms, for the representation of continuous-time and discrete-time signals.

Chapter 5 Transform analysis of LTI systems Chapter 5 introduces the concept of frequency response function and shows a close coupling of its shape to the poles and zeros of the system function. This leads to a set of tools which are then utilized for the analysis and design of LTI systems. A section reviewing similar techniques for continuous-time systems is included at the end of the chapter.

Chapter 6 Sampling of continuous-time signals This chapter is mainly concerned with the conditions that should be satisfied for the accurate representation of baseband and bandpass continuous-time signals by discrete-time signals. However, the treatment is extended to the sampling and reconstruction of discrete-time signals.

Chapter 7 The Discrete Fourier Transform Any finite number N of consecutive samples from a discrete-time signal can be uniquely described by its N-point Discrete Fourier Transform (DFT). Chapter 7 introduces the DFT, its properties, and its relationship to the Fourier transform representations introduced in Chapter 4.

Chapter 8 Computation of the Discrete Fourier Transform In Chapter 8, a number of efficient algorithms for the computation of DFT in practical applications are presented. These fast algorithms allow the efficient implementation of FIR filters in the frequency domain for applications that require filters with long impulse responses.

Chapter 9 Structures for discrete-time systems Chapter 9 is concerned with different structures for the representation and implementation of discrete-time systems described by linear constant-coefficient difference equations.

Chapter 10 Design of FIR filters Chapters 5 and 9 discussed techniques for the analysis and implementation of systems described by linear constant-coefficient difference equations with known coefficients. Chapter 10 presents procedures (design techniques) for obtaining values of FIR filter coefficients to approximate a desired frequency response function. Design techniques such as window technique, frequency-sampling technique, and Parks–McClellan algorithm are discussed.

Chapter 11 Design of IIR filters Chapter 11 presents design techniques for IIR systems with rational system functions. It begins with analog filter design and then continues with the transformation of analog lowpass filters to digital lowpass filters and then concludes with the filter-band transformation to obtain other frequency-selective digital filters.

Chapter 12 Multirate signal processing The first part introduces techniques for changing the sampling rate of a discrete-time signal using DSP algorithms. Special emphasis is given to the cases of decimation and interpolation of discrete-time signals and the design of digital filters for changing the sampling rate by a rational factor. The second part deals with the design and implementation of discrete-time filter banks. Both two-channel and multichannel filter banks with perfect reconstruction are discussed. The main emphasis is on filter banks used in practical applications.

Chapter 13 Random signals The main objective of Chapter 13 is to explain the nature of random signals and to introduce the proper mathematical tools for the description and analysis of random signals in the time and frequency domains.

Chapter 14 Random signal processing This chapter provides an introduction to spectral estimation techniques and the design of optimum filters (matched filters, Wiener filters, and linear predictors) and the Karhunen–Loève transform for random signals.

Chapter 15 Finite word length effects In practice, the samples of discrete-time signals, trigonometric numbers in Fourier transform computations, and filter coefficients are represented with finite precision (that is, by using a finite number of bits). Furthermore, all computations are performed with finite accuracy. Chapter 15 is devoted to the study of finite precision effects on digital signal processing operations.

Learning summary

- Signals are physical quantities that carry information in their patterns of variation. Continuous-time signals are continuous functions of time, while discrete-time signals are sequences of real numbers. If the values of a sequence are chosen from a finite set of numbers, the sequence is known as a digital signal. Continuous-time, continuous-amplitude signals are also known as analog signals.

- A system is a transformation or operator that maps an input signal to an output signal. If the input and output signals belong to the same class, the system carries the name of the signal class. Thus, we have continuous-time, discrete-time, analog, and digital systems. Systems with input and output signals from different classes are known as interface systems or converters from one signal type to another.

- Signal processing is concerned with the acquisition, representation, manipulation, transformation, and extraction of information from signals. In analog signal processing these operations are implemented using analog electronic circuits. Digital signal processing involves the conversion of analog signals into digital, processing the obtained sequence of finite precision numbers using a digital signal processor or general purpose computer, and, if necessary, converting the resulting sequence back into analog form.

TERMS AND CONCEPTS

Analog representation The physical representation of a continuous-time signal by a voltage or current proportional to its amplitude.

Analog-to-digital converter (ADC) A device used to convert analog signals into digital signals.

Analog signal Continuous-time signals are also called analog signals because their amplitude is "analogous" (that is, proportional) to the physical quantity they represent.

Analog signal processing (ASP) The conversion of analog signals into electrical signals by special transducers or sensors and their processing by analog electrical and electronic circuits.

Analog system See continuous-time system.

Binary code A group of bits (zeros and ones) representing a quantized numerical quantity.

Continuous-time signal A signal whose value $s(t)$ (amplitude) is defined for every value of the independent variable t (time).

Continuous-time system A system which transforms a continuous-time input signal into a continuous-time output signal.

Deterministic signal A signal whose future values can be predicted exactly from past values.

Digital representation The physical representation of a digital signal by a combination of ON/OFF pulses corresponding to the digits of a binary number.

Digital signal   A discrete-time signal whose amplitude $s[n]$ takes values from a finite set of real numbers.

Digital signal processing (DSP)   The representation of analog signals by sequences of numbers, the processing of these sequences by numerical computation techniques, and the conversion of such sequences into analog signals.

Digital-to-analog converter (DAC)   A device used to convert digital signals into analog signals.

Discrete-time signal   A signal whose value $s[n]$ is defined only at a discrete set of values of the independent variable n (usually the set of integers).

Discrete-time system   A system which transforms a discrete-time input signal into a discrete-time output signal.

Digital system   A system which transforms a digital input signal into a digital output signal.

Random signal   A signal whose future values cannot be predicted exactly from past values.

Quantization   The process of representing the samples of a discrete-time signal using binary numbers with a finite number of bits (that is, with finite accuracy).

Sampling   The process of taking instantaneous measurements (samples) of the amplitude of a continuous-time signal at regular intervals of time.

Sampling period   The time interval between consecutive samples of a discrete-time signal.

Sampling rate   The number of samples per second obtained during periodic sampling.

Signal   Any physical quantity that varies as a function of time, space, or any other variable or variables.

Signal processing   A discipline concerned with the acquisition, representation, manipulation, and transformation of signals.

System   An interconnection of elements and devices for a desired purpose.

FURTHER READING

1. A more detailed introduction to signals and systems can be found in many books, including Oppenheim *et al.* (1997) and Haykin and Van Veen (2003).
2. More advanced and broader treatments of discrete-time signal processing can be found in many textbooks. Oppenheim and Schafer (2010) and Proakis and Manolakis (2007) are closer to the approach followed in this book.
3. A detailed treatment of practical digital signal processors is provided in Kuo and Gan (2005), Kuo *et al.* (2006), and Welch *et al.* (2006).
4. A variety of digital signal processing applications are discussed in the following texts: image processing in Gonzalez and Woods (2008) and Pratt (2007), digital communication in Rice (2009), digital control in Dorf and Bishop (2008), digital audio and video in Zölder (2008) and Fischer (2008), computer music in Moore (1990), and radar in Richards (2005).

Review questions

1. What is a signal and how does it convey information?
2. Describe various different ways a signal can be classified.
3. What is the difference between a mathematical and physical representation of a signal?

4. Explain the differences between continuous-time, discrete-time, and digital signals in terms of mathematical and physical representations.

5. Describe the concept of a system and explain how it is represented mathematically.

6. What is a continuous-time system? A discrete-time system? Provide one example of each.

7. A continuous-time system is also called an analog system. Do you agree or disagree?

8. Why do we need interface systems and where do we need them? Provide a block-diagram description of such systems needed in signal processing.

9. Describe an analog-to-digital (A/D) converter.

10. Describe a digital-to-analog (D/A) converter.

11. What is the difference between a practical and an ideal A/D converter? Between a practical and ideal D/A converter?

12. What is signal processing and what are its different forms used in practice? Give one example of each form.

13. Describe analog signal processing (ASP) with the help of its simplified block diagram.

14. Describe digital signal processing (DSP) with the help of its simplified block diagram.

15. Why is DSP preferred over ASP? Are there any disadvantages?

2 Discrete-time signals and systems

In this chapter we discuss the basic concepts and the mathematical tools that form the basis for the representation and analysis of discrete-time signals and systems. We start by showing how to generate, manipulate, plot, and analyze basic signals and systems using MATLAB. Then we discuss the key properties of causality, stability, linearity, and time-invariance, which are possessed by the majority of systems considered in this book. We continue with the mathematical representation, properties, and implementation of linear time-invariant systems. The principal goal is to understand the interaction between signals and systems to the extent that we can adequately predict the effect of a system upon the input signal. This is extremely difficult, if not impossible, for arbitrary systems. Thus, we focus on linear time-invariant systems because they are amenable to a tractable mathematical analysis and have important signal processing applications.

Study objectives

After studying this chapter you should be able to:

- Describe discrete-time signals mathematically and generate, manipulate, and plot discrete-time signals using MATLAB.
- Check whether a discrete-time system is linear, time-invariant, causal, and stable; show that the input-output relationship of any linear time-invariant system can be expressed in terms of the convolution sum formula.
- Determine analytically the convolution for sequences defined by simple formulas, write computer programs for the numerical computation of convolution, and understand the differences between stream and block processing.
- Determine numerically the response of discrete-time systems described by linear constant-coefficient difference equations.

2.1   Discrete-time signals

A discrete-time signal $x[n]$ is a sequence of numbers defined for every value of the integer variable n. We will use the notation $x[n]$ to represent the nth sample of the sequence, $\{x[n]\}_{N_1}^{N_2}$ to represent the samples in the range $N_1 \leq n \leq N_2$, and $\{x[n]\}$ to represent the entire sequence. When the meaning is clear from the context, we use $x[n]$ to represent either the nth sample or the entire sequence. A discrete-time signal is *not* defined for noninteger values of n. For example, the value of $x[3/2]$ is not zero, just undefined. In this book, we use the terms *discrete-time signal* and *sequence* interchangeably.

When $x[n]$ is obtained by sampling a continuous-time signal $x(t)$, the interval T between two successive samples is known as the *sampling period* or *sampling interval*. The quantity $F_s = 1/T$, called the *sampling frequency* or *sampling rate*, equals the number of samples per unit of time. If T is measured in seconds, the units of F_s are number of samples per second (sampling rate) or Hz (sampling frequency).

Signal representation There are several ways to represent a discrete-time signal. The more widely used representations are illustrated in Table 2.1 by means of a simple example. Figure 2.1 also shows a pictorial representation of a sampled signal using index n as well as sampling instances $t = nT$. We will use one of the two representations as appropriate in a given situation.

The *duration* or *length L_x* of a discrete-time signal $x[n]$ is the number of samples from the first nonzero sample $x[n_1]$ to the last nonzero sample $x[n_2]$, that is $L_x = n_2 - n_1 + 1$. The range $n_1 \leq n \leq n_2$ is denoted by $[n_1, n_2]$ and it is called the *support* of the sequence.

Table 2.1 Discrete-time signal representations.

Representation	Example
Functional	$x[n] = \begin{cases} \left(\frac{1}{2}\right)^n, & n \geq 0 \\ 0, & n < 0 \end{cases}$
Tabular	$\begin{array}{c\|cccccccc} n & \ldots & -2 & -1 & 0 & 1 & 2 & 3 & \ldots \\ \hline x[n] & \ldots & 0 & 0 & 1 & \frac{1}{2} & \frac{1}{4} & \frac{1}{8} & \ldots \end{array}$
Sequence	$x[n] = \left\{ \ldots\ 0\ \underset{\uparrow}{1}\ \frac{1}{2}\ \frac{1}{4}\ \frac{1}{8}\ \ldots\ \right\}$
Pictorial	

[1] The symbol $\uparrow$ denotes the index $n = 0$; it is omitted when the table starts at $n = 0$.
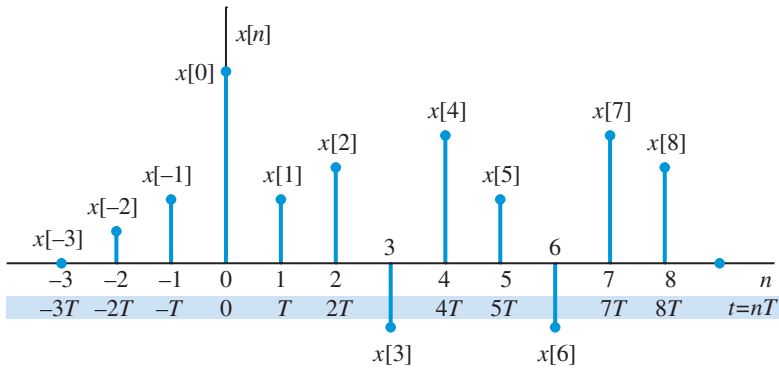
Figure 2.1 Representation of a sampled signal.

We shall use the notation $n \in [n_1, n_2]$ or $n \notin [n_1, n_2]$ to denote that n is inside or outside the range of support, respectively.

Energy and power The *energy* of a sequence $x[n]$ is defined by the formula

$$\mathcal{E}_x \triangleq \sum_{n=-\infty}^{\infty} |x[n]|^2. \tag{2.1}$$

Similarly, the *power* of a sequence $x[n]$ is defined as the average energy per sample

$$\mathcal{P}_x \triangleq \lim_{L \to \infty} \left[\frac{1}{2L+1} \sum_{n=-L}^{L} |x[n]|^2 \right]. \tag{2.2}$$

When $x[n]$ represents a physical signal, both quantities are directly related to the energy and power of the signal. Finite duration sequences have finite energy but zero power. However, when the duration of a sequence increases, the energy or power may or may not remain finite. Other characteristics and properties of discrete-time signals will be introduced as needed.

Elementary discrete-time signals Although practical signals are complicated and cannot be described by mathematical functions, there are some simple signals, see Figures 2.2 and 2.3, that are useful in the representation and analysis of discrete-time signals and systems.

Unit sample sequence The simplest discrete-time signal is the *unit sample* or *unit impulse* sequence, defined by

$$\delta[n] \triangleq \begin{cases} 1, & n = 0 \\ 0. & n \neq 0 \end{cases} \tag{2.3}$$

Unit step sequence The *unit step* sequence is given by

$$u[n] \triangleq \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases} \tag{2.4}$$

and can be thought of as an infinite succession of unit samples starting at $n = 0$.
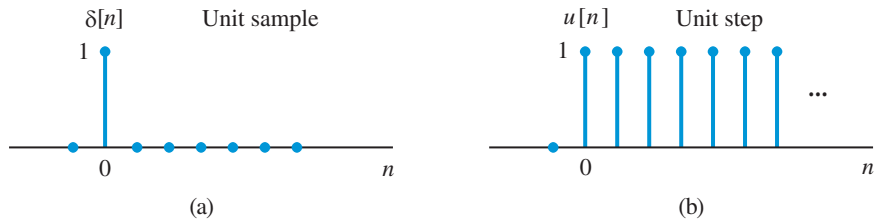
Figure 2.2 Some elementary discrete-time signals.
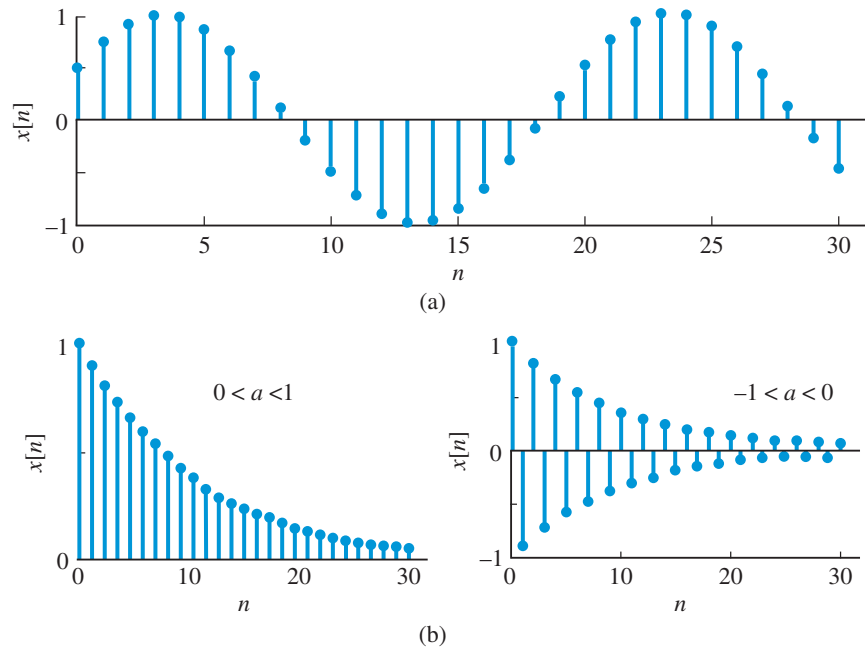
Figure 2.3 Examples of a discrete-time sinusoidal signal (a), and two real exponential sequences (b).

Sinusoidal sequence The *real sinusoidal* sequence has the general form

$$x[n] = A\cos(\omega_0 n + \phi), \quad -\infty < n < \infty \tag{2.5}$$

where A (*amplitude*) and ϕ (*phase*) are real constants. The quantity ω_0 is the *frequency* of the sinusoid and has units of radians per sampling interval. The values of this sequence keep on oscillating between $\pm|A|$ as shown in Figure 2.3(a) for $A = 1$.

Exponential sequence The *exponential* sequence has the general form defined by

$$x[n] \triangleq Aa^n, \quad -\infty < n < \infty \tag{2.6}$$

where A and a can take real or complex values.

- If both A and a are real numbers in (2.6) then $x[n]$ is termed as a *real exponential* sequence. For $-1 < a < 1$ ($a > 1$ or $a < -1$) the absolute value $|x[n]|$ of the

sequence decreases (increases) in magnitude with increasing n (see Figure 2.3(b) for $0 < a < 1$ and $-1 < a < 0$). The values of $x[n]$ alternate in sign when a is negative.

- If both $A = |A|\,\mathrm{e}^{\mathrm{j}\phi}$ and $a = \sigma_0 + \mathrm{j}\omega_0$ in (2.6) are complex-valued, then we have

$$x[n] = |A|\mathrm{e}^{\mathrm{j}\phi}\,\mathrm{e}^{\sigma_0 n + \mathrm{j}\omega_0 n} = |A|\mathrm{e}^{\sigma_0 n}\mathrm{e}^{\mathrm{j}(\omega_0 n + \phi)} \tag{2.7a}$$

$$= |A|\mathrm{e}^{\sigma_0 n}\cos(\omega_0 n + \phi) + \mathrm{j}|A|\mathrm{e}^{\sigma_0 n}\sin(\omega_0 n + \phi). \tag{2.7b}$$

For $\sigma_0 \neq 0$, the real and imaginary values of this sequence oscillate but with decreasing magnitude. For $\sigma_0 < 0$ ($\sigma_0 > 0$) the absolute value $|x[n]|$ of the sequence decreases (increases) in magnitude with increasing n.

Complex sinusoidal sequence One special case of the exponential sequence in (2.6) is when A is real-valued but $a = \mathrm{e}^{\mathrm{j}\omega_0}$ is complex-valued, that is,

$$x[n] = A\mathrm{e}^{\mathrm{j}\omega_0 n} = A\cos(\omega_0 n) + \mathrm{j}A\sin(\omega_0 n). \tag{2.8}$$

We will also refer to this sequence as the *complex exponential* sequence. Note that the sinusoidal sequence in (2.5) is the real part of (2.8) with $\phi = 0$.

Periodic sequence A sequence $x[n]$ is called *periodic* if

$$x[n] = x[n + N]. \quad \text{all } n \tag{2.9}$$

The smallest value of N for which (2.9) holds is known as the *fundamental period* or simply period of $x[n]$.

The sinusoidal sequence (2.5) is periodic, if $\cos(\omega_0 n + \phi) = \cos(\omega_0 n + \omega_0 N + \phi)$. This is possible if $\omega_0 N = 2\pi k$, where k is an integer. When k and N are prime numbers, N is equal to the number of samples in one fundamental period of the sequence. Figure 2.3 shows a discrete-time sinusoid with frequency $\omega_0 = 2\pi/15$ radians/sampling interval and phase $\phi = 2\pi/5$ radians. Thus, the period is $N = 15$ samples and the phase corresponds to a delay of $15 * (2\pi/5)/(2\pi) = 3$ sampling intervals. Sinusoidal sequences and complex exponential sequences obtained using Euler's identity $\mathrm{e}^{\mathrm{j}\theta} = \cos\theta + \mathrm{j}\sin\theta$ play a central role in the analysis of discrete-time signals and systems.

2.2 Signal generation and plotting in MATLAB

MATLAB provides a natural framework for the generation, plotting, and processing of discrete-time signals. Although the reader is assumed familiar with the fundamentals of MATLAB, the following observations will be helpful when we use MATLAB for digital signal processing applications:

- The only numerical data type utilized in MATLAB is the $N \times M$ matrix, that is, an array of numbers with N rows and M columns. Thus, a scalar is a 1×1 matrix, a column vector an $N \times 1$ matrix, and a row vector a $1 \times M$ matrix.
- The first element of a matrix is indexed by $(1, 1)$. Zero or negative indexes are *not* allowed. Thus, the sample $x[0]$ is stored as x(1) in MATLAB.

- MATLAB treats the elements of a matrix as complex numbers. Real numbers are treated as complex numbers with imaginary part equal to zero.
- The power of MATLAB lies in the high-level matrix-vector oriented operations.
- In MATLAB the user does not need to declare any variables; they are created by a memory manager when they are needed. The user can free space by removing unneeded variables from memory using the `clear` command.

Although in theory we are required to define a signal in the range $(-\infty, \infty)$, in MATLAB we can represent only a part of a signal within a finite range as a vector with a finite number of elements. We can use either column or row vectors.

Signal generation Any finite duration sequence $\{x[n]\}_{N_1}^{N_2}$ can be stored in MATLAB as a vector `x = [x(1) x(2) ... x(N)]` where `x(1)` $= x[N_1]$, etc. and $N = N_2 - N_1 + 1$. The timing information $N_1 \leq n \leq N_2$ is lost. If time information is needed, it can be saved at another vector `n = [N1 N1+1 ... N2]` and manipulated separately. Therefore, a complete representation of a sequence in MATLAB requires a *data* vector and an *index* vector. Clearly, infinite duration sequences *cannot* be saved and manipulated in MATLAB.

For example, to generate the sequence

$$x[n] = 2\cos(2\pi 0.05n), \quad -10 \leq n \leq 10$$

we can use the following MATLAB statements

```
n=(-10:10); x=2*cos(2*pi*0.05*n);
```

If we replace `n=(-10:10)` by `n=(-10:10)'`, then both `n` and `x` are column vectors. The statement `x=0.9.^n` creates a column vector containing the values of the sequence $x[n] = 0.9^n$, $-10 \leq n \leq 10$. In general, when the argument of a function, like `cos`, is a vector, the resulting sequence is a vector of the same size.

To sample a continuous-time signal from time t_1 to t_2 every T seconds, we define a vector of sampling points by `t=(t1:T:t2)`. To obtain a specific number of samples N, it is more convenient to use the statement `t=linspace(t1,t2,N)`. The values of the sampled sequence can be generated by a statement like `x=cos(2*pi*f0*t)`.

The following functions will be frequently used to generate some very useful sequences

```
[x,n] = delta(n1,n0,n2); % Unit impulse sequence
[x,n] = unitstep(n1,n0,n2); % Unit step sequence
[x,n] = unitpulse(n1,n2,n3,n4); % Unit pulse sequence
    x = persegen(xp,Np,Nps); % Periodic sequence
```

Functions `delta` and `unitstep` generate a unit sample and unit step sequences in the range `n=(n1:n2)`. The unit sample is located at `n=n0` and the unit step starts at `n=n0` (`n1` $\leq$ `n0` $\leq$ `n2`). The `unitpulse` function creates a rectangular pulse of unit amplitude from `n2` to `n3` and zero elsewhere (`n1` $\leq$ `n2` $\leq$ `n3` $\leq$ `n4`). Finally, `persegen` generates `Nps` periods of a periodic signal with period `Np`. The vector `xp`, appended with zeros when `length(xp)` $\leq$ `Np`, determines one period of the sequence.

Operations on sequences Addition, subtraction, multiplication, division, and scaling of sequences can be performed on a sample-by-sample basis:

$$y[n] = x_1[n] + x_2[n], \qquad \text{(signal addition)} \qquad (2.10)$$

$$y[n] = x_1[n] - x_2[n], \qquad \text{(signal subtraction)} \qquad (2.11)$$

$$y[n] = x_1[n] \cdot x_2[n], \qquad \text{(signal multiplication)} \qquad (2.12)$$

$$y[n] = x_1[n]/x_2[n], \qquad \text{(signal division)} \qquad (2.13)$$

$$y[n] = a \cdot x_2[n]. \qquad \text{(signal scaling)} \qquad (2.14)$$

Since MATLAB is a vector language, it provides powerful commands for such operations as long as the sequences have the same length and are defined in the same range (that is, have the same index vector). Otherwise, we must first properly augment the sequences with zeros using the function

```
[y1,y2,n]=timealign(x1,n1,x2,n2);
```

which properly inserts zeros to create two sequences with the same support. Then, we can use the statements `y1+y2`, `y1-y2`, `y1.*y2`, and `y1./y2` to perform element wise, that is, sample-by-sample, operations. Using vector operations we can compute the energy or power of a signal stored at vector `x` by

```
Ex=sum(x.*conj(x));        Px=Ex/length(x);
```

A type of slightly more complicated operation involves transformations of the independent variable n. Two important time-based transformations are:

- *Time-reversal* or *folding*, which is an operation defined by $y[n] = x[-n]$, reflects the sequence $x[n]$ about the origin $n = 0$. Folding a sequence in MATLAB is done using the function `[y,ny]=fold(x,nx)`. This time-reversal operation, which obviously *cannot* be done in real-time, is illustrated in Figure 2.4(b). If $x[-n] = x[n]$ the sequence is called *even* or *symmetric*; if $x[-n] = -x[n]$ it is called *odd* or *antisymmetric*.
- *Time-shifting* is defined by the formula $y[n] = x[n - n_0]$. For $n = n_0$ we have, $y[n_0] = x[0]$; thus, the sequence $x[n]$ is shifted by n_0 samples so that the sample $x[0]$ is moved to $n = n_0$. If $n_0 > 0$, the sequence $x[n]$ is shifted to the right; because the sequence "appears later," the shift corresponds to a *time-delay*. If $n_0 < 0$, the sequence is shifted to the left; because the sequence "appears earlier," the shift amounts to a *time-advance*. Time-shifting is illustrated in Figure 2.4(c), (d). In MATLAB, we can shift a sequence using the function `[y,n]=shift(x,n,n0)`.

The operations of shifting and folding are not commutative. Indeed, we have

$$x[n] \overset{\text{shift}}{\to} x[n - n_0] \overset{\text{fold}}{\to} x[-n - n_0] \neq x[n] \overset{\text{fold}}{\to} x[-n] \overset{\text{shift}}{\to} x[-n + n_0].$$

This important result is pictorially illustrated in Tutorial Problem 2.

(a) $x[n]$

(b) $x[-n]$ Folding

(c) $x[n-3]$ Time-delay

(d) $x[n+2]$ Time-advance
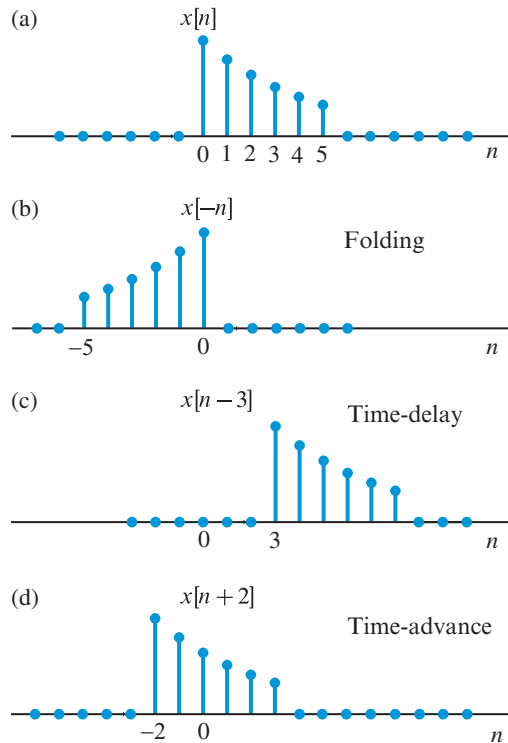
Figure 2.4 Folding and time-shifting operations.

Plotting To plot the sequence as a discrete-time signal (see Figure 2.3), we use the MATLAB function `stem` as follows

```
stem(n,x,'fill'); ylabel('x[n]'); xlabel('n');
```

When the number of samples is large the resulting stem plot becomes unintelligible. In such cases, we plot the envelope of the discrete-time signal using the function `plot` with a statement like `plot(n,x,'-')`. This function "connects" the dots of the sequence with a straight line segment. This process, which is known as linear interpolation, is discussed in Chapter 12.

Audio signals Although it is possible to plot audio (sound) signals, it is more informative to play and listen to these signals through a computer's built-in audio input/output devices using appropriate MATLAB functions. The `sound(x,Fs)` plays the signal `x` as an audio through speakers at `Fs` Hz rate. To read a wave file from disk into signal `x`, the `[x,Fs]=wavread('wavefile')` can be used. Similarly, the `wavwrite(x,Fs,'wavefle')` function is used to store `x` as a wave signal at `Fs` Hz rate. Additionally for Windows machines, the `wavrecord` and `wavplay` functions are available to record and play, respectively, audio signals from a computer's input/output devices. Tutorial Problem 6 discusses some of these functions.

Figure 2.5 Block diagram representation of a discrete-time system.

2.3 Discrete-time systems

A *discrete-time system* is a computational process or algorithm that transforms or maps a sequence $x[n]$, called the input signal, into another sequence $y[n]$, called the output signal. In practice, a discrete-time system is a numerical algorithm that processes an input sequence $x[n]$ of finite length, to produce a finite length output sequence $y[n]$. We shall denote a discrete-time system symbolically by

$$x[n] \xmapsto{\mathcal{H}} y[n] \quad \text{or} \quad y[n] = \mathcal{H}\{x[n]\}, \tag{2.15}$$

and graphically as shown in Figure 2.5. The symbol $\xmapsto{\mathcal{H}}$ stands for "maps to by operator $\mathcal{H}$." These representations, which hold for all n, are simply shorthand ways to say that there is a cause and effect relationship between $x[n]$ and $y[n]$. The term *filter* is often used interchangeably with the term system. However, strictly speaking, a filter is a special system designed to remove some components from the input signal or to modify some characteristics of the input signal in a certain way. In this sense, the term system is more general; however, in this book, we use both terms interchangeably.

A discrete-time system should be described by a mathematical formula or rule which unambiguously describes how to determine its output from the input. For example, the equation

$$y[n] = \frac{1}{3}\{x[n] + x[n-1] + x[n-2]\} \tag{2.16}$$

describes a three-point moving average filter, which is often used to smooth a signal corrupted by additive noise, for all values of n.

The five-point median filter, used to remove spikes from experimental data, is defined by

$$y[n] = \text{median}\{x[n-1], x[n-2], x[n], x[n+1], x[n+2]\}. \tag{2.17}$$

To determine the output, we sort the five indicated samples according to their value and then pick the middle sample.

The usefulness of general discrete-time systems is limited because their analysis and design are extremely difficult. To bypass this problem we focus on limited classes of discrete-time systems that satisfy some or all of the properties discussed in Sections 2.3.1 and 2.3.2. Unless otherwise stated, each of these properties is understood to hold for all input sequences.

2.3.1   Causality and stability

Definition 2.1 A system is called *causal* if the present value of the output does not depend on future values of the input, that is, $y[n_0]$ is determined by the values of $x[n]$ for $n \leq n_0$, only.

If the output of a system depends on future values of its input, the system is *noncausal*. Causality implies that if $x[n] = 0$ for $n < n_0$, then $y[n] = 0$ for $n < n_0$; that is, a causal system cannot produce an output before the input is applied. The discrete-time system (2.16) is causal. The system (2.17) is noncausal because the input samples $x[n+1]$ and $x[n+2]$ are not available when the output sample $y[n]$ needs to be computed. This noncausal system can be implemented in real-time if we delay the generation of its output by two sampling intervals, that is, compute $y[n]$ at time $t = (n+2)T$. Clearly, this problem does not exist if the entire input sequence is already stored in memory. Although causality is necessary for the real-time implementation of discrete-time systems, it is not really a problem in off-line applications where the input signal has been already recorded.

For any system to be useful, the input and output values should be always finite. In practical terms, this means that the implementation of the system does not lead to overflows. This leads to the concept of stability. In practical systems stability guarantees that, if the input signal is within the number range of the computer, there will not be any overflows in the output signal, that is, the system will not "blow-up." (If we require all internal variables of the system to be bounded we need the concept of *internal stability* or stability in the sense of Liapunov.) We now provide a formal definition of stability.

Definition 2.2 A system is said to be *stable*, in the Bounded-Input Bounded-Output (BIBO) sense, if every bounded input signal results in a bounded output signal, that is

$$|x[n]| \leq M_x < \infty \Rightarrow |y[n]| \leq M_y < \infty. \tag{2.18}$$

A signal $x[n]$ is bounded if there exists a positive finite constant M_x such that $|x[n]| \leq M_x$ for all n.

Example 2.1
The moving-average system (2.16) is stable. To prove this, we assume that the input is bounded, that is, $|x[n]| \leq M_x$ for all n. Using the inequality $|a + b| \leq |a| + |b|$, we have $|y[n]| \leq |x[n]| + |x[n-1]| + |x[n-2]| \leq 3M_x$. Therefore, we can choose $M_y = 3M_x$, and prove that the output is bounded. In contrast, to prove that a system is unstable, one counterexample is sufficient. Thus, the accumulator system defined by $y[n] = \sum_{k=0}^{\infty} x[n-k]$ is unstable because the bounded input $x[n] = u[n]$ produces the output $y[n] = (n+1)u[n]$, which becomes unbounded as $n \to \infty$. ∎

Since unstable systems generate unbounded output signals, that is overflows, they cannot be used in practical applications.

Linearity and time invariance

Stability is a property that should be satisfied by every practical system, whereas causality is required for systems that should operate in real-time. However, the properties that make the analysis of discrete-time systems mathematically tractable are linearity and time-invariance.

Definition 2.3 A system is called *linear* if and only if for every real or complex constant a_1, a_2 and every input signal $x_1[n]$ and $x_2[n]$

$$\mathcal{H}\{a_1 x_1[n] + a_2 x_2[n]\} = a_1 \mathcal{H}\{x_1[n]\} + a_2 \mathcal{H}\{x_2[n]\}, \tag{2.19}$$

for all values of n.

Equation (2.19), which is known as the *principle of superposition*, says that a linear combination of input signals produces the same linear combination of outputs. The superposition principle can be decomposed into two parts. If $a_2 = 0$ we have, $\mathcal{H}\{a_1 x_1[n]\} = a_1 \mathcal{H}\{x_1[n]\}$, which is the *homogeneity* or *scaling* property. Also, if $a_1 = a_2 = 1$, we have $\mathcal{H}\{x_1[n] + x_2[n]\} = \mathcal{H}\{x_1[n]\} + \mathcal{H}\{x_2[n]\}$, which is the *additivity* property. Linearity simplifies the analysis of discrete-time systems because we can decompose a complicated input signal into simpler components, determine the response to each individual component separately, and then compute the sum of all individual responses. Systems which do not satisfy the principle of superposition are said to be *nonlinear*.

An important consequence of linearity is that a linear system cannot produce an output without being excited. Indeed, since any zero input can be expressed as $x[n] = a_1 x_1[n] + a_2 x_2[n]$ with $a_1 = a_2 = 0$, it easily follows from (2.17) that for every linear system

$$x[n] = 0 \overset{\mathcal{H}}{\longmapsto} y[n] = 0. \tag{2.20}$$

In the following example, we illustrate the use of a linearity test based on Definition 2.3.

Example 2.2　Test for linearity

Test whether the following square-law system is linear or nonlinear:

$$y[n] = x^2[n]. \tag{2.21}$$

The test, which can be applied to any system, involves the following steps:

1. Apply the input $x_1[n]$ and use (2.21) to obtain the output $y_1[n] = x_1^2[n]$.
2. Apply the input $x_2[n]$ and use (2.21) to obtain the output $y_2[n] = x_2^2[n]$.
3. Apply the input $x[n] = a_1 x_1[n] + a_2 x_2[n]$ and use (2.21) to obtain the output $y[n]$. The result is $y[n] = a_1^2 x_1^2[n] + a_2^2 x_2^2[n] + 2a_1 a_2 x_1[n] x_2[n]$.

4. Form the signal $y_3[n] = a_1y_1[n] + a_2y_2[n] = a_1x_1^2[n] + a_2x_2^2[n]$ and check whether it is equal to $y[n]$. If the answer is yes, then the system is linear; otherwise it is nonlinear.

Since $y_3[n] \neq y[n]$, the system is nonlinear.   ∎

If the characteristics of a system do not change with time, the system is called *time-invariant*; otherwise it is called *time-varying*. This means that the shape of the output of a time-invariant system depends only on the shape of the input signal and not on the time instant the input was applied into the system. More precisely, we have the following definition.

Definition 2.4 A system is called *time-invariant* or *fixed* if and only if

$$y[n] = \mathcal{H}\{x[n]\} \Rightarrow y[n - n_0] = \mathcal{H}\{x[n - n_0]\}, \tag{2.22}$$

for every input $x[n]$ and every time shift n_0. That is, a time shift in the input results in a corresponding time shift in the output.

The following example illustrates how to test whether a system described by an input-output relationship is time-invariant.

Example 2.3   Test for time invariance

Test whether the following system is time-invariant or time-varying:

$$y[n] = x[n]\cos\omega_0 n. \tag{2.23}$$

Based on Definition 2.4 we perform a test that involves the following steps:

1. Apply an input $x_1[n] = x[n]$ and use (2.23) to compute the output $y_1[n] = x[n]\cos\omega_0 n$.
2. Apply the shifted input $x_2[n] = x[n - n_0]$ and use (2.23) to compute the output $y_2[n] = x[n - n_0]\cos\omega_0 n$.
3. Check whether the shifted sequence $y_1[n - n_0]$ is equal to $y_2[n]$. If the answer is yes the system is time-invariant; otherwise it is time-varying.

Since $y_1[n - n_0] = x[n - n_0]\cos\omega_0([n - n_0]) \neq y_2[n]$ the system is time-varying.   ∎

Example 2.4   Test for linearity and time invariance

A *downsampler* is a system,

$$y[n] = \mathcal{H}\{x[n]\} = x[nM], \tag{2.24}$$

that is used to sample a discrete-time signal $x[n]$ by a factor of M. Test the downsampler for linearity and time invariance.

To test linearity, let $y_1[n] = x_1[nM]$ and $y_2[n] = x_2[nM]$. Consider the downsampler output due to the input $x[n] = a_1 x_1[n] + a_2 x_2[n]$ given by

$$y[n] = \mathcal{H}\{x[n]\} = x[nM] = a_1 x_1[nM] + a_2 x_2[nM]$$
$$= a_1 y_1[n] + a_2 y_2[n].$$

Hence the downsampler is linear. To test time invariance, consider the output $y_2[n]$ due to the input $x[n - n_0]$

$$y_2[n] = \mathcal{H}\{x[n - n_0]\} = x[nM - n_0]$$
$$\neq y[n - n_0] = x[(n - n_0)M] = x[nM - n_0 M].$$

Thus the downsampler is time-varying.   ■

In summary, linearity means that the output due to a sum of input signals equals the sum of outputs due to each signal alone. Time-invariance means that the system does not change over time. The majority of analysis and design techniques presented in this book are for linear and time-invariant systems. Therefore, tests that can be used to determine whether a system is linear and time-invariant are essential.

| 2.3.3 | Block diagrams, signal flow graphs, and practical realizability |

Operations required in the implementation of a discrete-time system can be depicted in one of two ways: a block diagram or a signal flow graph. A block diagram provides a pictorial view of the overall operation of the system using simple interconnection of basic building blocks. A signal flow graph graphically defines the precise set of operations necessary for the system implementation. Elements of these two representations are shown in Figure 2.6.

Basic building blocks The implementation of discrete-time systems requires (1) the means to perform numerical computations, and (2) memory to save signal values and other parameters. The most widely used operations are provided by the four elementary discrete-time systems (or building blocks) shown on the left side in Figure 2.6. Arithmetic operations are performed using addition and multiplication. The *adder*, defined by $y[n] = x_1[n] + x_2[n]$, computes the sum of two sequences. The constant *multiplier*, defined by $y[n] = ax[n]$, produces the product of the input sequence by a constant. The basic memory element is the *unit delay* system defined by $y[n] = x[n - 1]$ and denoted by the z^{-1} operator which we shall study in Chapter 3. The unit delay is a memory location which can hold (store) the value of a sample for one sampling interval. Finally, the branching element is used to distribute a signal value to different branches.

We note that, if the output $y[n]$ for every n depends only on the input $x[n]$ at the same time, the system is said to be *memoryless*; otherwise it is said to be *dynamic*. However, we emphasize that the practical implementation of a memoryless system, like $y[n] = 2x^2[n]$, requires memory to store the multiplying factor 2 and the value of $x[n]$.

Figure 2.7 shows the block diagram of a system which computes the first difference $y[n] = x[n] - x[n - 1]$ of its input. For example, if the system is excited by the unit

Block Diagram Elements

Signal Flow Graph Elements

$x_2[n]$

$x_2[n]$

$x_1[n] \longrightarrow \boxed{+} \quad y[n] = x_1[n] + x_2[n]$

Adder

$x_1[n] \quad \longrightarrow \quad y[n] = x_1[n] + x_2[n]$

Summing node

a

$x[n] \longrightarrow \quad y[n] = ax[n]$

Multiplier

a

$x[n] \longrightarrow y[n] = ax[n]$

Gain branch

$x[n] \longrightarrow \boxed{z^{-1}} \quad y[n] = x[n-1]$

Unit delay

z^{-1}

$x[n] \longrightarrow y[n] = x[n-1]$

Unit delay branch

$w[n] \longrightarrow w[n]$

$w[n]$

Splitter

$w[n] \longrightarrow w[n]$

$w[n]$

Pick-off node
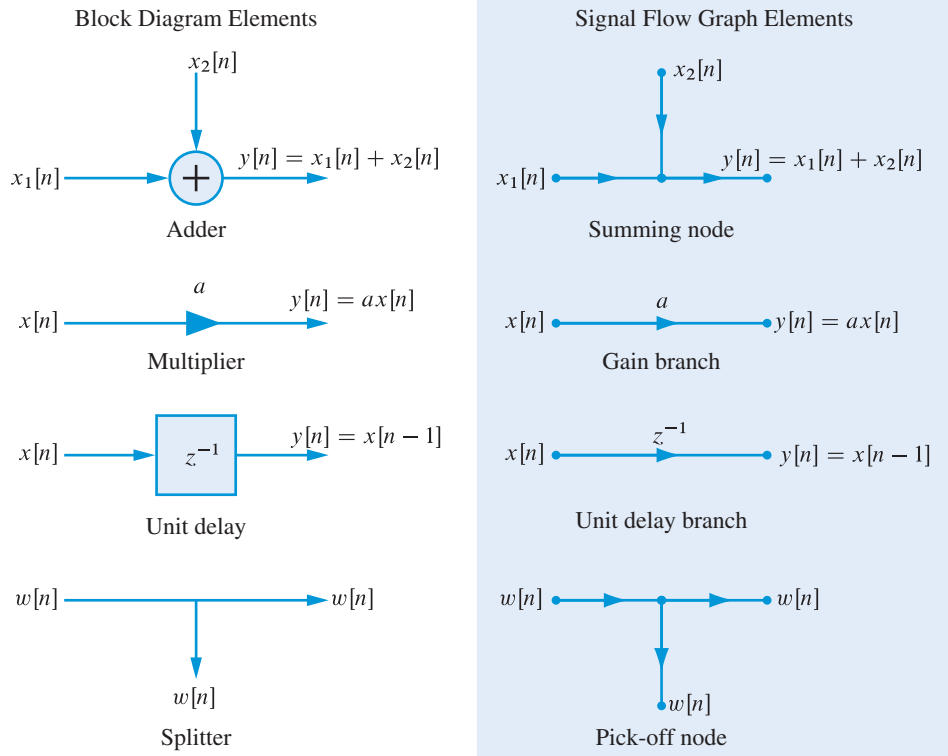
Figure 2.6 Basic building blocks and the corresponding signal flow graph elements for the implementation of discrete-time systems.

$x[n]$

$y[n]$

$\boxed{z^{-1}}$

-1
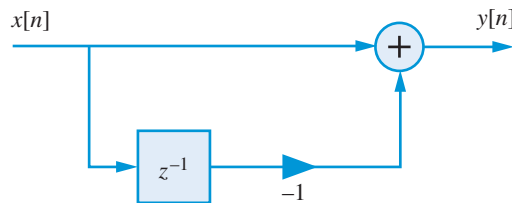
Figure 2.7 Discrete-time system whose output is the first difference of the input signal.

step sequence $u[n]$ the response is the unit sample sequence $\delta[n]$. Block diagrams provide a concise pictorial representation of the algorithm required to implement a discrete-time system and they can serve as a basis to develop software or hardware for its practical implementation.

Signal flow graphs This graphical representation is defined using branches and nodes. Operations such as gain and delay are specified using *directed* branches in which the gain or delay values are shown next to the branch arrow (unit gains are not explicitly shown). Nodes provide the connection points for branches as well as indicate signal values. The *summing* node is used to depict the addition operation while the *pick-off* node provides for
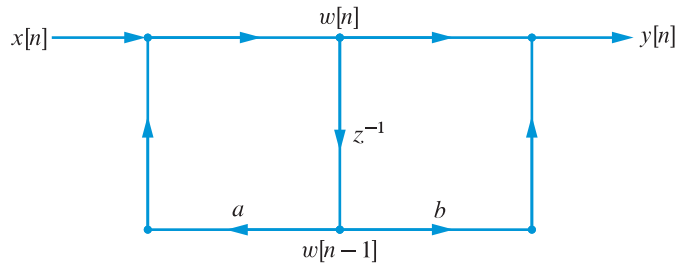
Figure 2.8 Signal flow graph of a first-order discrete-time system.

branch splitting. These signal flow graph elements are shown on the right side in Figure 2.6 and they correspond to the respective block-diagram elements. Signal entering a branch is taken as the signal value of the input node of the branch. There is at least one input branch where an external signal enters a system and at least one output branch where system output is obtained.

Figure 2.8 shows a signal flow graph representation of a discrete-time system in which the input branch applies signal $x[n]$ to the system and the output $y[n]$ is obtained at the rightmost node. Using an intermediate node signal $w[n]$ as shown, we can write down the following set of equations:

$$w[n] = x[n] + aw[n-1], \qquad \text{(input node)} \qquad (2.25a)$$

$$y[n] = w[n] + bw[n-1]. \qquad \text{(output node)} \qquad (2.25b)$$

After a simple manipulation to eliminate $w[n]$ in (2.25), we obtain

$$y[n] = x[n] + bx[n-1] + ay[n-1], \qquad (2.26)$$

which represents a general first-order discrete-time system.

Practical realizability A discrete-time system is called *practically realizable* if its practical implementation requires (1) a finite amount of memory for the storage of signal samples and system parameters, and (2) a finite number of arithmetic operations for the computation of each output sample. Clearly, any system which does not satisfy either of these conditions cannot be implemented in practice.

Most discrete-time systems discussed in this book will possess all the properties summarized in Table 2.2. We stress that all these properties are properties of systems and not properties of the input signals. Thus, to prove that a system possesses a certain property, we should show that the property holds for every input signal and for all n. However, one counterexample is sufficient to prove that a system does not have a given property.

2.4   Convolution description of linear time-invariant systems

Theoretical and practical applications require the ability to determine the effect of a system upon a class of input signals (e.g. speech), and design systems which can produce that

Table 2.2 Summary of discrete-time system properties.

Property	Input		Output
	$x[n]$	$\xmapsto{\mathcal{H}}$	$y[n]$
	$x_k[n]$	$\xmapsto{\mathcal{H}}$	$y_k[n]$
Linearity	$\sum_k c_k x_k[n]$	$\xmapsto{\mathcal{H}}$	$\sum_k c_k y_k[n]$
Time-invariance	$x[n - n_0]$	$\xmapsto{\mathcal{H}}$	$y[n - n_0]$
Stability	$|x[n]| \le M_x < \infty$	$\xmapsto{\mathcal{H}}$	$|y[n]| \le M_y < \infty$
Causality	$x[n] = 0$ for $n \le n_0$	$\xmapsto{\mathcal{H}}$	$y[n] = 0$ for $n \le n_0$
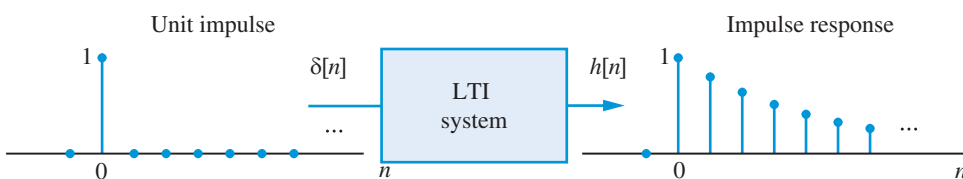
Figure 2.9 The impulse response of a linear time-invariant system.

effect and evaluate the performance of the system. The specification of the desired "effects" in precise mathematical terms requires a deep understanding of signal properties and is the subject of signal analysis. Understanding and predicting the effect of a general system upon the input signal is almost impossible. To develop a meaningful and feasible analysis, we limit our attention to systems that possess the properties of linearity and time-invariance.

The main premise of this section is that the response of a linear time-invariant (LTI) system to any input can be determined from its response $h[n]$ to the unit sample sequence $\delta[n]$ (see Figure 2.9), using a formula known as convolution summation. The sequence $h[n]$, which is known as *impulse response*, can also be used to infer all properties of a linear time-invariant system. Without linearity, we can only catalog the system output for each possible input.

A fundamental implication of linearity is that individual signals which have to be summed at the input are processed independently inside the system, that is, they super-impose and do not interact with each other. The superposition property greatly simplifies the analysis of linear systems, because if we express an input $x[n]$ as a sum of simpler sequences

$$x[n] = \sum_k a_k x_k[n] = a_1 x_1[n] + a_2 x_2[n] + a_3 x_3[n] + \cdots , \tag{2.27}$$

then the response $y[n]$ is given by

$$y[n] = \sum_k a_k y_k[n] = a_1 y_1[n] + a_2 y_2[n] + a_3 y_3[n] + \cdots , \tag{2.28}$$

where $y_k[n]$ is the response to an input $x_k[n]$ acting alone. There are three requirements for such an expansion to be useful:

1. The set of basic signals can be used to represent a very large class of useful signals.
2. There is a simple way to determine the coefficients a_k. It is also desirable to be able to compute the value of each coefficient without knowledge of the value of any other coefficient.
3. It should be easy to compute the response of a LTI system to each basic signal and synthesize the overall output from the individual responses.

We consider two sets of signals that satisfy these requirements. In this chapter we use the basic signals $x_k[n] = \delta[n - k]$; in Chapters 3, 4, and 5 we consider the decomposition of signals into complex exponential sequences.

Signal decomposition into impulses Let us define the sequence

$$x_k[n] = \begin{cases} x[k], & n = k \\ 0, & n \neq k \end{cases} \tag{2.29}$$

which consists of the sample $x[k]$ of $\{x[n]\}$ at $n = k$ and zero elsewhere. The sequence $x_k[n]$ can be obtained by multiplying the sequence $\{x[n]\}$ by the sequence

$$\delta[n - k] = \begin{cases} 1, & n = k \\ 0. & n \neq k \end{cases} \tag{2.30}$$

Hence, the sequence $\{x[n]\}$ can be expressed as

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k]. \quad -\infty < n < \infty \tag{2.31}$$

The left hand side of (2.31) represents the sequence $x[n]$ as a whole whereas the right hand side summations represent the sequence as a superposition of scaled and shifted unit sample sequences (see Figure 2.10). For example, the unit step can be written as a sum of delayed impulses
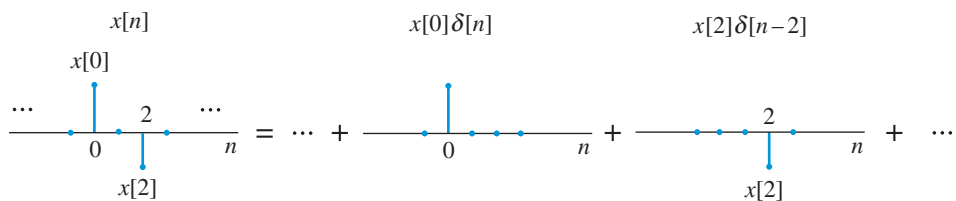
Figure 2.10 Decomposition of a discrete-time signal into a superposition of scaled and delayed unit sample sequences.

$$u[n] = \delta[n] + \delta[n-1] + \delta[n-2] + \cdots \tag{2.32}$$

$$= \sum_{k=0}^{\infty} \delta[n-k] = \sum_{k=-\infty}^{n} \delta[k]. \tag{2.33}$$

Clearly, any arbitrary sequence can be expressed as a sum of scaled and delayed impulses. The coefficient of the basic signal $\delta[n-k]$ is easily obtained as the value of the signal $x[n]$ at $n = k$.

Convolution sum We now illustrate how the properties of linearity and time-invariance restrict the form of discrete-time systems and simplify the understanding and analysis of their operation. More specifically, we show that we can determine the output of any LTI system if we know its impulse response.

We start by recalling that any sequence $x[n]$ can be decomposed into a superposition of scaled and shifted impulses as in (2.31). Consider next a linear (but possibly time-varying) system and denote by $h_k[n]$ its response to the basic signal $\delta[n-k]$. Then, from the superposition property for a linear system (see (2.27) and (2.28)), the response $y[n]$ to the input $x[n]$ is the same linear combination of the basic responses $h_k[n]$, that is,

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h_k[n], \tag{2.34}$$

which is known as the *superposition summation* formula. Equation (2.34) provides the response of a linear time-varying system in terms of the responses of the system to the impulses $\delta[n-k]$.

If we impose the additional constraint that the system is time-invariant, we have

$$\delta[n] \overset{\mathcal{H}}{\longmapsto} h[n] \Rightarrow \delta[n-k] \overset{\mathcal{H}}{\longmapsto} h_k[n] = h[n-k]. \tag{2.35}$$

Substitution of (2.35) into (2.34) gives the formula

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \quad -\infty < n < \infty \tag{2.36}$$

Equation (2.36), which is commonly called the *convolution sum* or simply *convolution* is denoted using the notation $y[n] = x[n] * h[n]$. Therefore, the response of a linear time-invariant system to any input signal $x[n]$ can be determined from its impulse response $h[n]$ using the convolution sum (2.36). If we know the impulse response of an LTI system, we can compute its response to any input without using the actual system. Furthermore, if we have no access to the internal implementation of the system (that is, we treat the system as a black box), we can try to "reverse-engineer" the system from its impulse response.

The operation described by the convolution sum takes two sequences $x[n]$ and $h[n]$ and generates a new sequence $y[n]$. We usually say that sequence $y[n]$ is the *convolution* of sequences $x[n]$ and $h[n]$ or that $y[n]$ is obtained by convolving $x[n]$ with $h[n]$. Convolution

describes how a linear time-invariant system modifies the input sequence to produce its output. Therefore, it is important to understand the mechanism portrayed by (2.36) and its interpretations.

Understanding the convolution sum To grasp the meaning of the convolution sum, we expand the summation in (2.36) and we explicitly write the resulting expressions for a few values of n, say $n = -1, 0, 1, 2, 3$. The result is

$$y[-1] = \cdots + x[-1]h[0] + x[0]h[-1] + x[1]h[-2] + x[2]h[-3] + \cdots$$
$$y[0] = \cdots + x[-1]h[1] + x[0]h[0] \quad + x[1]h[-1] + x[2]h[-2] + \cdots$$
$$y[1] = \cdots + x[-1]h[2] + x[0]h[1] \quad + x[1]h[0] \quad + x[2]h[-1] + \cdots$$
$$y[2] = \cdots + x[-1]h[3] + x[0]h[2] \quad + x[1]h[1] \quad + x[2]h[0] \quad + \cdots$$
$$y[3] = \cdots + x[-1]h[4] + x[0]h[3] \quad + x[1]h[2] \quad + x[2]h[1] \quad + \cdots \qquad (2.37)$$

There are two ways to look at (2.37): one equation at a time or all equations as a block. Each approach leads to a different interpretation and implementation of the convolution sum.

Convolution as a "scanning" operation Careful inspection of equation (2.36) leads to the following important observations:

- The samples of the sequence $x[k]$ are in natural order whereas the samples of the sequence $h[k]$ are in reverse order (flipping or time reversal).
- To determine the value of $y[n]$ for $n = n_0$, the flipped impulse response sequence is shifted so that the sample $h[0]$ is aligned to sample $x[n_0]$ of the input.

This process can be aided by writing the sequence of numbers $x[k]$ and $h[-k]$ on two separate strips of paper as shown in Figure 2.11 for the sequences

$$x[n] = \{1\ 2\ 3\ 4\ 5\}, \quad h[n] = \{-1\ 2\ 1\}. \qquad (2.38)$$

The index $n = 0$ is marked with a small arrow on both strips. We note that $h[-k]$ is $h[k]$ written in reverse order (backwards). To find the value of $y[n]$ for $n = n_0$, we slide the
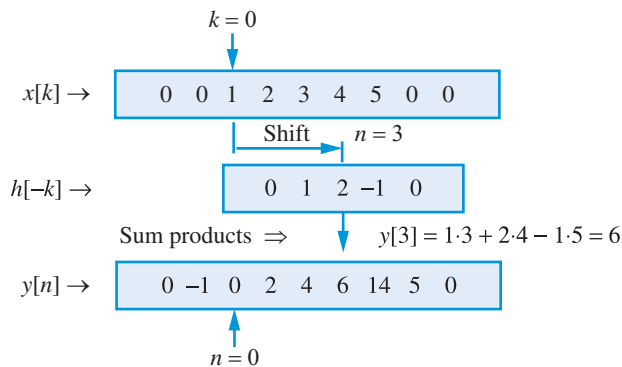
Figure 2.11 Graphical illustration of convolution using paper strips.

$h[-k]$ strip so that $h[0]$ is aligned to $x[n_0]$ and compute the sum of the products of adjacent numbers. The process, which should be repeated for all n, is illustrated in Figure 2.11 for $n = 3$.

We shall now provide a pictorial interpretation of the convolution operation by convolving the sequences

$$x[n] = \{\underset{\uparrow}{1}\ 1\ 1\ 1\ 1\ 1\}, \quad h[n] = \{\underset{\uparrow}{1}\ 0.5\ 0.25\ 0.125\}, \tag{2.39}$$

using the convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \tag{2.40}$$

The first crucial point to remember is that the summation is performed with respect to index k, so that n is just a parameter. Thus, we start by sketching the two sequences as a function of k, not of n, as shown in Figure 2.12. To obtain the sequence $h[n-k]$, we first reflect $h[k]$ about the vertical axis at $k = 0$. This yields the sequence

$$g[k] = h[-k]. \tag{2.41}$$

If we shift $g[k]$ by n samples, we have

$$g[k - n] = h[-(k - n)] = h[n - k]. \tag{2.42}$$

For positive n the shift is to the right; for negative n the shift is to the left (see also (2.37). The value of n is the index of the output sample, $y[n]$, we wish to compute.

We next multiply the sequences $x[k]$ and $h[n-k]$, to obtain the second sequence $z_n[k] = x[k]h[n-k]$ in the convolution sum. The sum of the samples of $z_n[k]$ provides the value of the output sample $y[n]$. To obtain another output sample, we shift $h[-k]$ to align $h[0]$ with the new output sample position, we multiply $x[k]$ by $h[n-k]$, and we sum the samples of the product. We stress that as the product sequence $x[k]h[n-k]$ changes with the amount of shift n so does the sum of its samples, that is, the output sample $y[n]$. Figure 2.12 illustrates this process for $n = 2$; however, the reader can repeat the process to derive the entire output sequence $y[n] = \left\{\underset{\uparrow}{1}, 1.5, 1.75, 1.875, 1.875, 1.875, 0.875, 0.375, 0.125\right\}$.

The process outlined in Figure 2.12 is repeated for every output sample we wish to compute, either by analytical or numerical means. The successive shifting of the sequence $h[-k]$ over the sequence $x[k]$ can be viewed as a "scanning" and "accumulate" process, where the samples of the input are "weighted" by the samples of the impulse response before summation. We will find this interpretation very useful when we study the operation and implementation of linear time-invariant systems.

In summary, the computation of convolution of two sequences involves the following steps:

1. Change the index of the sequences $h[n]$, $x[n]$ from n to k and plot them as a function of k.
2. Flip (or fold) the sequence $h[k]$ about $k = 0$ to obtain the sequence $h[-k]$.
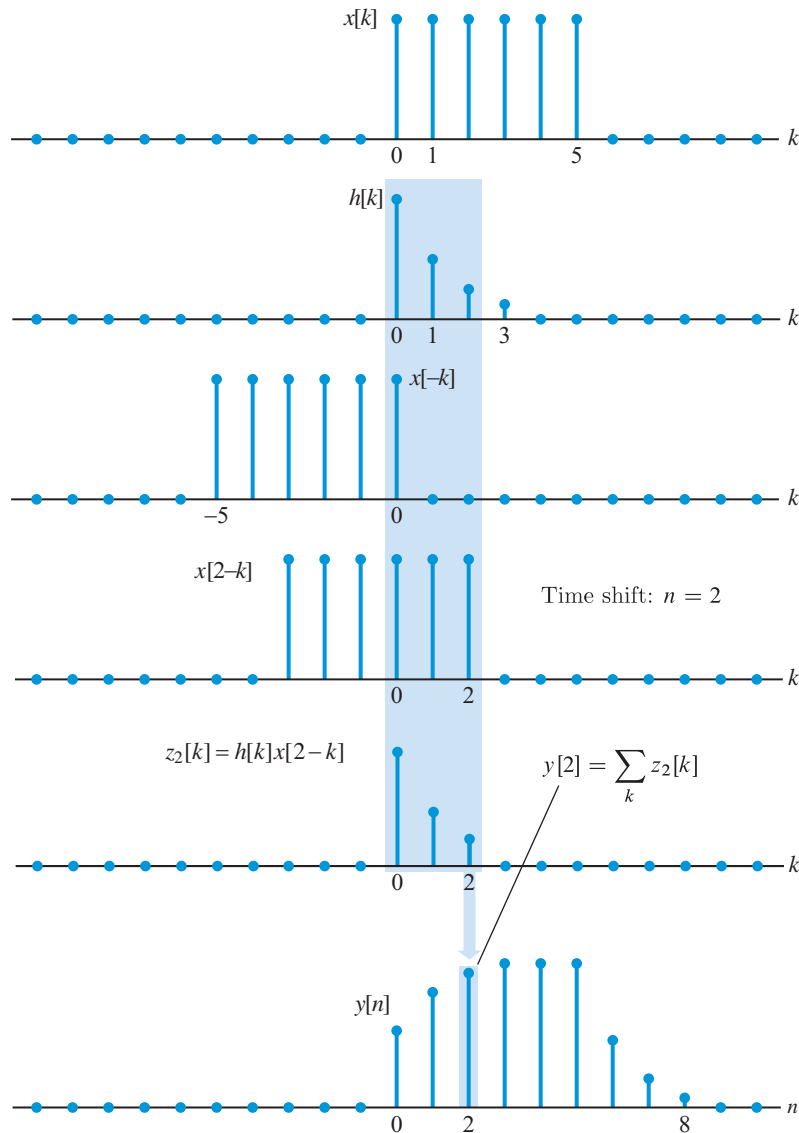
Figure 2.12 Graphical illustration of convolution as a scanning operation.

3. Shift the flipped sequence $h[-k]$ by n samples to the right, if $n > 0$, or to the left, if $n < 0$.
4. Multiply the sequences $x[k]$ and $h[n-k]$ to obtain the sequence $z_n[k] = x[k]h[n-k]$.
5. Sum all nonzero samples of $z_n[k]$ to determine the output sample at the given value of the shift n.
6. Repeat steps $3 - 5$ for all desired values of n.

Figure 2.13 illustrates this "scan," multiply, and add procedure in a tabular form, using the sequences given in (2.38). For example, if $n = 2$, we have $z_2[k] = x[k]h[2-k] = \{-2\ 6\ 4\}$ and $y[2] = -2 + 6 + 4 = 8$.

k	-3	-2	-1	0	1	2	3	4	5	6	7
$x[k]$				1	2	3	4	5			
$h[k]$			1	2	-1						
$h[-1-k]$		-1	2	1							
$h[-k]$			-1	2	1						
$h[1-k]$				-1	2	1					
$h[2-k]$					-1	2	1				
$h[3-k]$						-1	2	1			
$h[4-k]$							-1	2	1		
$h[5-k]$								-1	2	1	
$h[n]$			-1	0	2	4	6	14	5		
n	-3	-2	-1	0	1	2	3	4	5	6	7

Figure 2.13 The computation of convolution in tabular form.

If we interchange $h[n]$ and $x[n]$, that is we write the products in each equation of (2.37) in reverse order or we flip both strips in Figure 2.11, the result of convolution does not change. Hence,

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]. \tag{2.43}$$

Therefore, when we convolve two sequences we can flip the sequence which makes the computation of convolution sum easier.

Convolution as a superposition of scaled and shifted replicas If we now look at the columns of (2.37), we note that each column is a shifted impulse response sequence $h[n-k]$, $-\infty < n < \infty$, multiplied by the value $x[n]$ of the input at $n = k$. The sum of all these scaled and shifted sequences produces the output sequence $y[n]$. This viewpoint can be reinforced by considering an alternative derivation of the convolution sum outlined by the following steps:

$$\delta[n] \overset{\mathcal{H}}{\longmapsto} h[n] \qquad \text{(Impulse response)}$$

$$\delta[n-k] \overset{\mathcal{H}}{\longmapsto} h[n-k] \qquad \text{(Time-invariance)}$$

$$x[k]\delta[n-k] \overset{\mathcal{H}}{\longmapsto} x[k]h[n-k] \qquad \text{(Homogeneity)}$$

$$\underbrace{\sum_{k=-\infty}^{\infty} x[k]\delta[n-k]}_{x[n]} \overset{\mathcal{H}}{\longmapsto} \underbrace{\sum_{k=-\infty}^{\infty} x[k]h[n-k]}_{y[n]}. \qquad \text{(Additivity)}$$

The equations in the last line lead to the convolution sum formula (2.36). A pictorial illustration of this approach is given in Tutorial Problem 7.
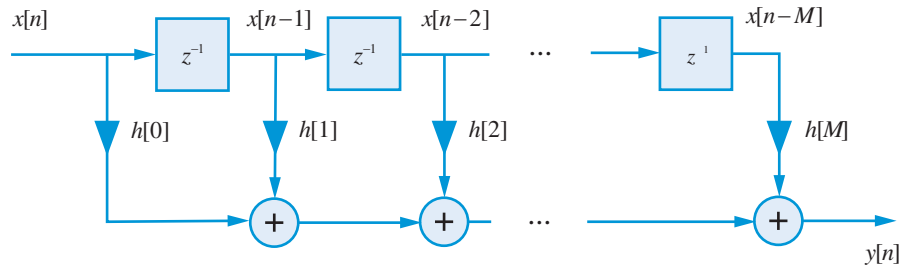
Figure 2.14 Block diagram representation of an FIR system.

FIR versus IIR systems The duration of the impulse response leads to two different types of linear time-invariant system. If the impulse response has a finite number of nonzero samples (finite support), we have a *finite (duration) impulse response (FIR)* system. Otherwise, we have a system with *infinite (duration) impulse response (IIR)*. Figure 2.14 illustrates the block diagram realization of an FIR system using the basic discrete-time system building blocks. Obviously, computing the convolution sum for IIR systems requires an infinite number of unit delays and arithmetic operations. However, as we will see in Section 2.10, there is a class of IIR systems that can be realized using a finite amount of memory and arithmetic operations.

2.5 Properties of linear time-invariant systems

From a mathematical viewpoint the roles of $h[n]$ and $x[n]$ in the convolution sum are equivalent; it is immaterial if we convolve $h[n]$ with $x[n]$ or vice versa. However, in the context of linear time-invariant systems, the roles played by the impulse response and the input are *not* equivalent. The nature of $h[n]$ determines the effect of the system on the input signal $x[n]$. Since all linear time-invariant systems are described by a convolution sum, we can use the properties of convolution to study their properties and determine the impulse response of interconnected systems.

2.5.1 Properties of convolution

If we consider a system with impulse response $h[n] = \delta[n]$, direct substitution into the convolution formula gives $y[n] = x[n]$ because only the term for $k = n$ is nonzero. Hence, the sequence $\delta[n]$ is the *identity element* of the convolution operation. Similarly, we can see that the output of the system with impulse response $h[n] = \delta[n - n_0]$ is $y[n] = x[n - n_0]$. This is an *ideal delay* system that delays the input signal by n_0 sampling intervals.

The convolution operation is *commutative*, that is

$$h[n] * x[n] = x[n] * h[n]. \tag{2.44}$$
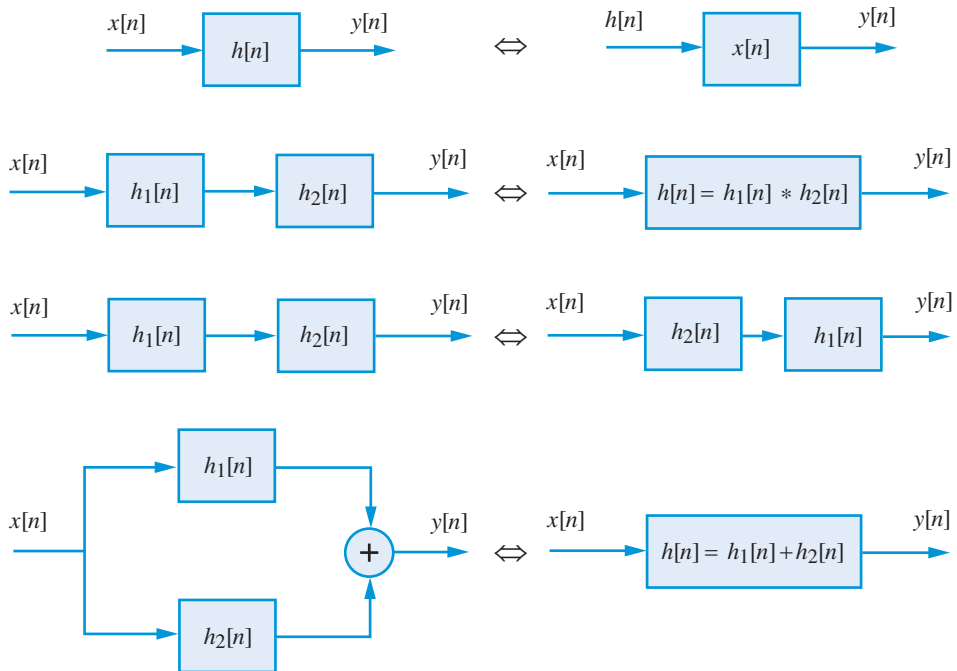
Figure 2.15 Convolution properties in the context of linear time-invariant systems. Systems on the same row are equivalent.

This can be shown by changing the summation variable k by $m = n - k$ in (2.36) as follows

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] = \sum_{m=-\infty}^{\infty} h[m]x[n - m] = h[n] * x[n]. \qquad (2.45)$$

Therefore, a linear time-invariant system with input $x[n]$ and impulse response $h[n]$ will have the same output as a system having impulse response $x[n]$ and input $h[n]$.

Now consider the *cascade interconnection* of two linear time-invariant systems, where the output of the first system is input to the second system (see Figure 2.15). The outputs of these systems are

$$v[n] = \sum_{k=-\infty}^{\infty} x[k]h_1[n - k] \quad \text{and} \quad y[n] = \sum_{m=-\infty}^{\infty} h_2[m]v[n - m]. \qquad (2.46)$$

Substituting the first equation into the second and interchanging the order of the summations, we have

$$y[n] = \sum_{k=-\infty}^{\infty} x[k] \sum_{m=-\infty}^{\infty} h_2[m]h_1[(n - k) - m]. \qquad (2.47)$$

Table 2.3 Summary of convolution properties.	
Property	**Formula**
Identity	$x[n] * \delta[n] = x[n]$
Delay	$x[n] * \delta[n - n_0] = x[n - n_0]$
Commutative	$x[n] * h[n] = h[n] * x[n]$
Associative	$(x[n] * h_1[n]) * h_2[n] = x[n] * (h_1[n] * h_2[n])$
Distributive	$x[n] * (h_1[n] + h_2[n]) = x[n] * h_1[n] + x[n] * h_2[n]$

We can easily see that the last summation is the convolution of $h_1[n]$ and $h_2[n]$ evaluated at $n - k$. If we define the sequence $h[n] \triangleq h_1[n] * h_2[n]$, then from (2.47) we obtain

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] = x[n] * h[n]. \tag{2.48}$$

Hence, the impulse response of two linear time-invariant systems connected in cascade is the convolution of the impulse responses of the individual systems.

If we consider the *parallel interconnection* of two linear time-invariant systems (see Figure 2.15) it is easy to show that

$$y[n] = h_1[n] * x[n] + h_2[n] * x[n] = (h_1[n] + h_2[n]) * x[n] \triangleq h[n] * x[n], \tag{2.49}$$

where $h[n] \triangleq h_1[n] + h_2[n]$. Therefore, the impulse response of two systems connected in parallel is the sum of the individual impulse responses.

The properties of convolution are summarized in Table 2.3 whereas their implications for system interconnections are illustrated in Figure 2.15.

2.5.2 Causality and stability

Since a linear time-invariant system is completely characterized by its impulse response sequence $h[n]$, we can use $h[n]$ to check whether the system is causal and stable.

Result 2.5.1 A linear time-invariant system with impulse response $h[n]$ is causal if

$$h[n] = 0 \quad \text{for} \quad n < 0. \tag{2.50}$$

Proof. If we write the convolution sum (2.36) in expanded form as

$$y[n] = \cdots + h[-1]x[n + 1] + h[0]x[n] + h[1]x[n - 1] + \cdots, \tag{2.51}$$

we see that the present output value $y[n]$ does not depend on the future input values $x[n + 1], x[n + 2], \ldots$, only if $h[n] = 0$ for $n < 0$. ∎

Due to (2.50), we often use the term *causal* for sequences with zero values for $n < 0$, because they can serve as impulse responses of causal systems.

Result 2.5.2 A linear time-invariant system with impulse response $h[n]$ is stable, in the bounded-input bounded-output sense, if and only if the impulse response is absolutely summable, that is, if

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty. \tag{2.52}$$

Proof. We shall first use Definition 2.2 to prove that condition (2.52) is sufficient, that is, if (2.52) holds the system is stable. If we assume that $x[n]$ is bounded, that is, $|x[n]| \leq M_x < \infty$ for all n, we have

$$|y[n]| = \left| \sum_{k=-\infty}^{\infty} h[k]x[n-k] \right| \leq \sum_{k=-\infty}^{\infty} |h[k]||x[n-k]| \leq M_x \sum_{k=-\infty}^{\infty} |h[k]|. \tag{2.53}$$

Since, by assumption, $S_h \triangleq \sum_{k=-\infty}^{\infty} |h[k]| = M_h < \infty$, we have $|y[n]| \leq M_h M_x \triangleq M_y < \infty$. Hence, $y[n]$ is bounded and the system is stable.

To prove that condition (2.52) is necessary, we shall show that there is a bounded sequence, which creates an unbounded response when (2.52) does not hold. Indeed, consider the input sequence

$$x[n] \triangleq \begin{cases} 1, & h[n] \geq 0 \\ -1, & h[n] < 0 \end{cases} \tag{2.54}$$

which is clearly bounded since $|x[n]| = 1$. The output of the system at $n = 0$ is

$$y[0] = \sum_{k=-\infty}^{\infty} h[k]x[-k] = \sum_{k=-\infty}^{\infty} |h[k]| = S_h, \tag{2.55}$$

and becomes infinity if $S_h = \infty$. Hence, if $S_h = \infty$, it is possible for a bounded input to produce an unbounded output. ∎

FIR systems are always stable. Indeed, we have

$$S_h = \sum_{k=-\infty}^{\infty} |h[k]| = \sum_{k=0}^{M} |h[k]| < \infty, \tag{2.56}$$

because M is finite. However, as the following example shows, IIR systems may or may not be stable.

Example 2.5

Consider the system with impulse response $h[n] = ba^n u[n]$. To test whether the system is stable, we check if the following sum is finite

$$S_h = \sum_{k=-\infty}^{\infty} |h[k]| = |b| \sum_{k=0}^{\infty} |a|^n. \tag{2.57}$$

If $|a| < 1$, the sum converges to $|b|/(1 - |a|)$ where we have used the sum of geometric series formula (see Tutorial Problem 9). Therefore, the system is stable only when $|a| < 1$. In this case, the impulse response decays asymptotically to zero. ∎

2.5.3   Convolution of periodic sequences

When one or both sequences to be convolved are periodic, the convolution sum may not always be finite. We can better understand the key issues if we view the convolution sum as the input-output relationship of a linear time-invariant system.

We first show that the response of a stable linear time-invariant system to a periodic input is periodic with the same period. Indeed, if we replace n by $n + N$ in (2.36), we obtain

$$y[n + N] = \sum_k h[k]x[n + N - k]. \tag{2.58}$$

Since the periodicity condition $x[n + N] = x[n]$, holds for all n, replacing n by $n - k$ gives $x[n + N - k] = x[n - k]$. Substitution of the last relation in (2.58) implies that $y[n+N] = y[n]$. Therefore, the convolution summation can be used for both aperiodic and periodic inputs as long as the linear time-invariant system is stable.

If $h[n]$ is periodic with period N then the system is unstable because the sum $\sum_{k=-\infty}^{\infty} |h[n]|$ is always infinite. If $\sum_{k=-\infty}^{\infty} |x[n]|$ is finite, then the convolution $y[n]$ exists and is periodic with period N. If $x[n]$ is periodic, say with period L, then the convolution sum cannot be finite. However, if N and L are commensurable $x[n]h[n]$ is periodic with period equal to the lowest common multiple of N and L. Then, we can define the so called "periodic convolution" by summing over one period. Periodic convolution has many important applications in digital signal processing (see Chapter 7).

2.5.4   Response to simple test sequences

To understand the behavior of a linear time-invariant system, we study its effect upon some simple test signals. Then, we can use the principle of superposition to understand its effect upon more complicated signals.

The simplest test signal is the unit sample sequence, which generates the impulse response $h[n]$. Then, by exploiting linearity and time invariance, we can use $h[n]$ to build the response to any other sequence.

The *step response*, that is, the response to the unit step sequence, helps to understand the "reaction" of a system to suddenly applied inputs. It is given by

$$s[n] = \sum_{k=-\infty}^{\infty} h[k]u[n-k] = \sum_{k=-\infty}^{n} h[k], \qquad (2.59)$$

because $u[n-k] = 0$ for $n-k < 0$ or $k > n$. Hence, the step response is the cumulative sum of the impulse response. Alternatively, the impulse response is the first-difference of the step response, that is, $h[n] = s[n] - s[n-1]$.

Consider now the response to the sequence $x[n] = a^n$, $-\infty < n < \infty$, where a can take real or complex values. Using the convolution sum, we have

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]a^{n-k} = \left(\sum_{k=-\infty}^{\infty} h[k]a^{-k} \right) a^n. \qquad (2.60)$$

The quantity inside the parentheses is a function $H(a)$ of the parameter a. The quantity $H(a)$ exists if $|H(a)| < \infty$. For example, if $h[n] = u[n] - u[n-M]$, we have

$$H(a) = \sum_{k=-\infty}^{\infty} h[k]a^{-k} = \sum_{k=0}^{M-1} a^{-k} = \frac{1-a^M}{1-a}, \qquad (2.61)$$

(see Tutorial Problem 7) which leads to

$$y[n] = H(a)a^n = \left(\frac{1-a^M}{1-a} \right) a^n. \qquad (2.62)$$

Therefore, the response of a stable linear time-invariant system to the exponential sequence $x[n] = a^n$, $-\infty < n < \infty$, is the same sequence multiplied by a system-dependent constant $H(a)$.

An important special case occurs for $a = e^{j\omega}$, that is for the complex exponential sequence $x[n] = e^{j\omega n}$. The response is just the input sequence scaled by a complex constant

$$y[n] = \left(\sum_{k=-\infty}^{\infty} h[k]e^{-j\omega k} \right) e^{j\omega n} = H(e^{j\omega})e^{j\omega n}. \qquad (2.63)$$

The quantity $H(e^{j\omega})$, known as a frequency response function, plays a prominent role in the analysis and design of linear time-invariant systems using frequency domain techniques. The responses to these basic test signals, which are part of the toolbox for linear system analysis, are summarized in Table 2.4.

2.6   Analytical evaluation of convolution

To compute the convolution $y[n]$ at a particular index n, we should sum all nonzero values of the product sequence $h[k]x[n-k]$, $-\infty < k < \infty$. In general, the range of the summation depends on the value of shift index n (see Figure 2.12). We next present a graphical procedure which illustrates how to determine the ranges of summation for the convolution sum and the support of the convolution sequence $y[n] = h[n] * x[n]$ of two, arbitrarily positioned, finite length sequences $\{x[n]\}_{N_1}^{N_2}$ and $\{h[n]\}_{M_1}^{M_2}$.

Table 2.4 Response of linear time-invariant systems to some test sequences.

Type of response	Input sequence		Output sequence
Impulse	$x[n] = \delta[n]$	$\stackrel{\mathcal{H}}{\longmapsto}$	$y[n] = h[n]$
Step	$x[n] = u[n]$	$\stackrel{\mathcal{H}}{\longmapsto}$	$y[n] = s[n] = \sum\limits_{k=-\infty}^{n} h[k]$
Exponential	$x[n] = a^n$, all n	$\stackrel{\mathcal{H}}{\longmapsto}$	$y[n] = H(a)a^n$, all n
Complex sinusoidal	$x[n] = \mathrm{e}^{\mathrm{j}\omega n}$, all n	$\stackrel{\mathcal{H}}{\longmapsto}$	$y[n] = H(\mathrm{e}^{\mathrm{j}\omega})\mathrm{e}^{\mathrm{j}\omega n}$, all n

$H(a) = \sum_{-\infty}^{\infty} h[n]a^{-n}$

We start by drawing, just for clarity, the envelopes of the two sequences; the shape of the envelopes is not important. Figure 2.16(a) shows the sequences $x[k]$ and $h[n-k]$ as a function of the summation index k. The sequence $h[n-k]$ is obtained by folding $h[k]$ to obtain $h[-k]$ and then shifting, by n samples, to get $h[n-k]$. Note that the sample $h[M_1]$ is now located at $k = n - M_1$ and the sample $h[M_2]$ at $k = n - M_2$. Since $M_1 \leq M_2$ this reflects the time-reversal (flipping) of the sequence $h[k]$. For illustration purposes, without loss of generality, we choose n to position $h[n-k]$ on the left of $x[k]$. Changing the parameter n will shift $h[n-k]$ to a different position along the k-axis. Careful inspection of Figure 2.16 shows that, depending on the overlap between the sequences $x[k]$ and $h[n-k]$, there are three distinct limits of summation for the convolution sum. These limits are indicated by the beginning and the end of the shaded intervals. Clearly, the convolution sum is zero when $n - M_1 < N_1$ or $n - M_2 > N_2$ because the sequences $x[k]$ and $h[n-k]$ do not overlap. Therefore, $y[n]$ is nonzero in the range $L_1 = M_1 + N_1 \leq n \leq L_2 = M_2 + N_2$. The three distinct ranges for the convolution sum are defined as follows.

Partial overlap (left) The range of summation, as shown in Figure 2.16(b), is from $k = N_1$ to $k = n - M_1$. This range is valid as long as $n - M_1 \geq N_1$ or $n \geq M_1 + N_1$ and $n - M_2 \leq N_1$ or $n \leq M_2 + N_1$. Hence, we have

$$y[n] = \sum_{k=N_1}^{n-M_1} x[k]h[n-k], \quad \text{for } N_1 + M_1 \leq n \leq N_1 + M_2.$$

Full overlap The range of summation, as shown in Figure 2.16(c), is from $k = n - M_2$ to $k = n - M_1$. This range is valid as long as $n - M_2 > N_1$ or $n > N_1 + M_2$ and $n - M_1 < N_2$ or $n < M_1 + N_2$. Hence,

$$y[n] = \sum_{k=n-M_2}^{n-M_1} x[k]h[n-k], \quad \text{for } N_1 + M_2 < n < M_1 + N_2.$$

Partial overlap (right) The range of summation, as shown in Figure 2.16(d), is from $k = n - M_2$ to $k = N_2$. This range is valid as long as $n - M_1 \geq N_2$ or $n \geq M_1 + N_2$ and $n - M_2 \leq N_2$ or $n \leq M_2 + N_2$. Hence,

(a)

$h[n-k]$ $x[k]$

$n-M_2$ $n-M_1$ N_1 N_2 k

(b)

$x[k]$

$h[n-k]$

$n-M_2$ N_1 $n-M_1$ N_2 k

(c)

$x[k]$

$h[n-k]$

N_1 $n-M_2$ $n-M_1$ N_2 k

(d)

$x[k]$

$h[n-k]$

N_1 $n-M_2$ N_2 $n-M_1$ k
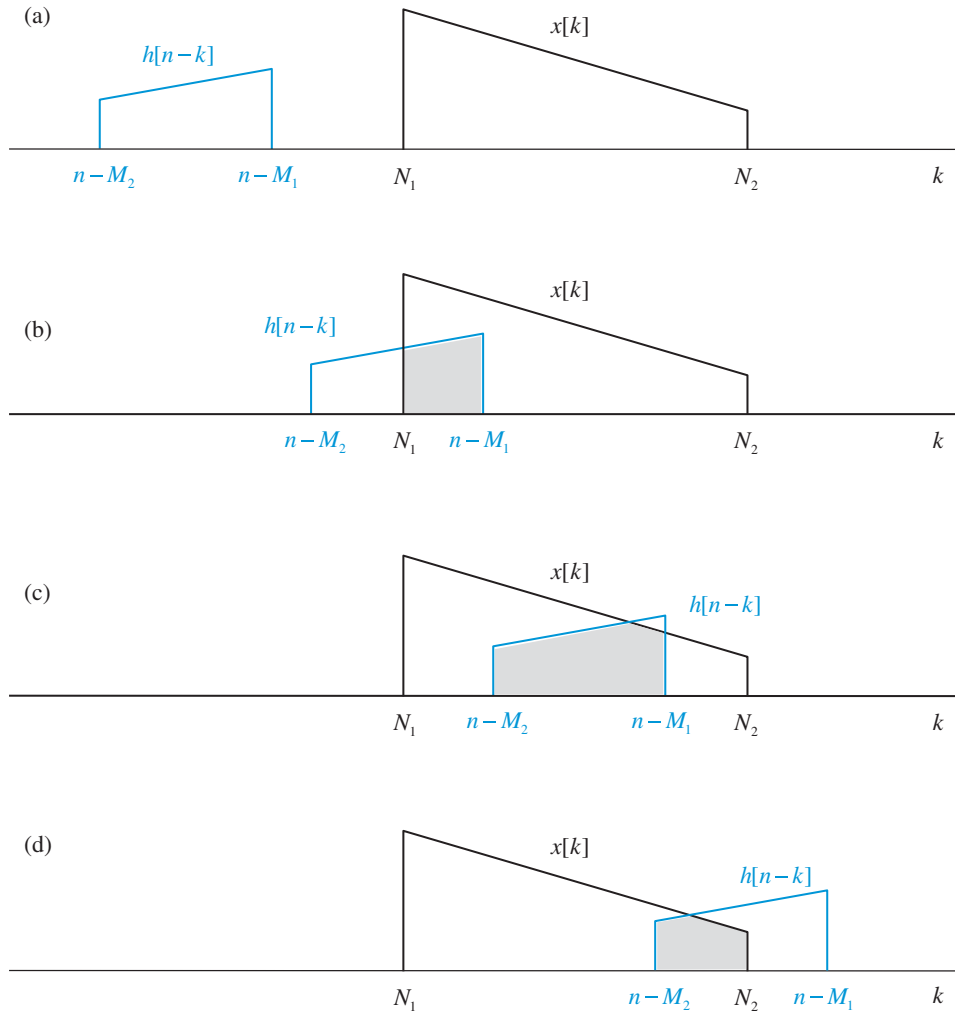
Figure 2.16 Visual aid to determine the limits for the convolution sum of finite duration sequences for $(N_2 - N_1) > (M_2 - M_1)$.

$$y[n] = \sum_{k=n-M_2}^{N_2} x[k]h[n-k], \quad \text{for } M_1 + N_2 < n < M_2 + N_2.$$

In conclusion, the convolution of $h[n]$, $n \in [M_1, M_2]$ and $x[n]$, $n \in [N_1, N_2]$ is a sequence $y[n]$, $n \in [M_1 + N_1, M_2 + N_2]$. This result holds for any values (positive or negative) of the limits.

When the impulse response and input sequences are given by simple formulas, we can determine the convolution sequence analytically. We illustrate this process with the following example.

Example 2.6

Compute the output $y[n]$ of a linear time-invariant system when the input $x[n]$ and the impulse response $h[n]$ are given by

$$x[n] = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad h[n] = \begin{cases} a^n, & 0 \le n \le M-1 \\ 0, & \text{otherwise} \end{cases} \tag{2.64}$$

respectively. We assume that $M < N$; the case $N > M$ is discussed in Problem 26.

The basic ideas underlying the computation of convolution are explained by working in detail through this example. However, the same principles can be used for different problems.

We start by plotting the sequences $h[k]$, $x[k]$, and $h[-k]$ as shown in Figure 2.17. Note that we have replaced the index n by the dummy index k to comply with formula (2.36). The location of the sample $h[0]$, when we shift the sequence $h[n-k]$, indicates the time shift n because $n - k = 0$ at $k = n$. Therefore, negative (positive) values of n correspond to shifting $h[n-k]$ to the left (right), that is moving $h[0]$ to the left (right) of $k = 0$. Shifting $h[n-k]$ for different values of n leads to five different ranges for the summation in the convolution formula.

No overlap When $n < 0$ (shifting $h[n-k]$ to the left), the two sequences do not overlap and the product sequence $x[k]h[n-k]$ is zero; hence

$$y[n] = 0 \text{ for } n < 0. \tag{2.65}$$

Partial overlap (left) The partial overlap of the two sequences starts at shift $n = 1$ and ends at shift $n = M - 2$. Therefore, when the shift is in the range $0 \le n \le M - 2$, the product $x[k]h[n-k]$ is nonzero in the range $0 \le k \le n$; hence

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=0}^{n} a^k 1 = \frac{1-a^{n+1}}{1-a}. \tag{2.66}$$

Full overlap The full overlap of the two sequences begins when the first sample $h[0]$ arrives at $n = M - 1$; it ends when the last sample $h[-N+1]$ arrives at $n = N - 1$. The range of summation, which has constant duration M (the length of the short sequence), is from $K_1 = 0$ to $K_2 = M - 1$. Hence

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = \sum_{k=0}^{M-1} a^k 1 = \frac{1-a^M}{1-a}. \tag{2.67}$$

Partial overlap (right) When $h[0]$ moves to $n = N$, we have the beginning of partial overlap, which lasts until $h[-N+1]$ comes to $n = M + N - 2$. Simple inspection of Figure 2.17 indicates that the upper limit of summation is fixed at $K_2 = M - 1$ and the lower limit is changing, $L_1 = n - N + 1$. Thus

$$y[n] = \sum_{k=n-N+1}^{M-1} a^k 1 = a^{n-N+1} \frac{1-a^{M+N-n-1}}{1-a}. \tag{2.68}$$
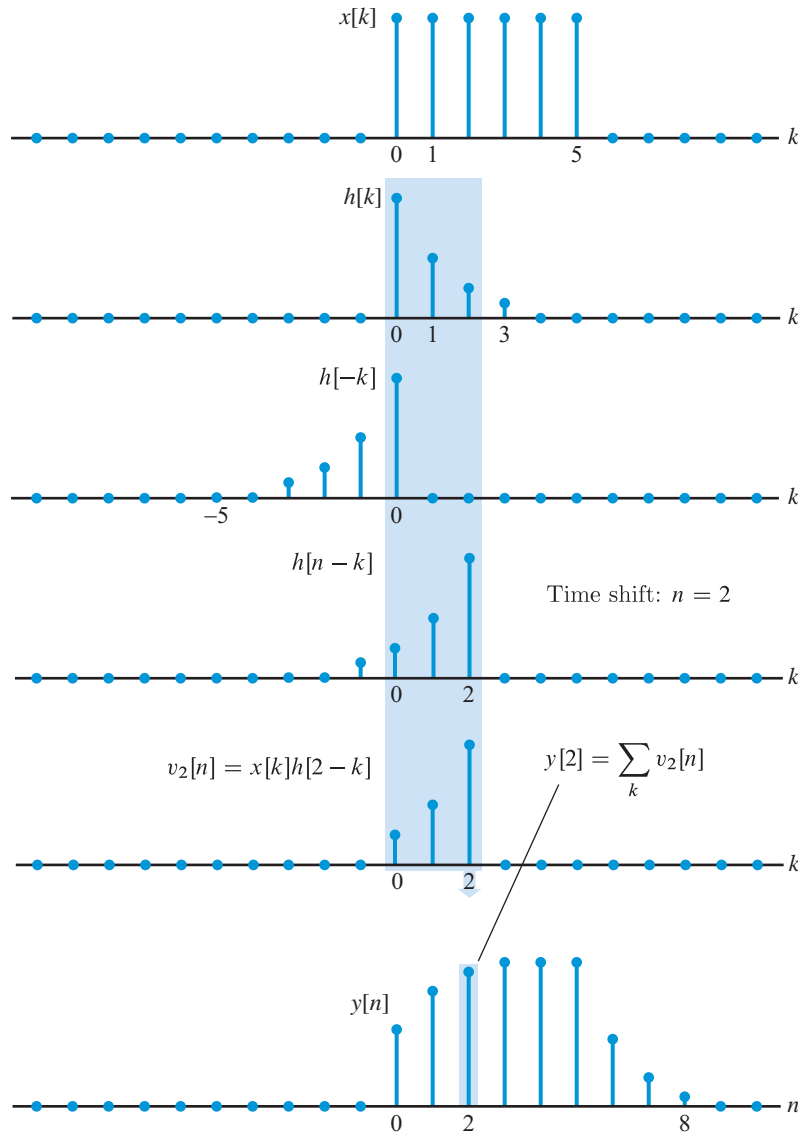
Figure 2.17 Graphical illustration to determine limits of summation in the computation of convolution.

No overlap When the last sample $h[n - N + 1]$ arrives at $k = M$, the two sequences cease to overlap; hence, there is no overlap after $n - N + 1 = M$ or $n = M + N - 1$. Therefore, we have

$$y[n] = 0 \text{ for } n \geq M + N - 1. \tag{2.69}$$

Equations (2.65)–(2.69) provide an analytical expression for the convolution $y[n]$ of the sequences $h[n]$ and $x[n]$. From the last plot in Figure 2.17 we can easily conclude that the length of the sequence $y[n]$ is $L = N + M - 1$.

Note also that Figures 2.12 and 2.17 depict the convolution of the same $x[n]$ and $h[n]$ signals. In Figure 2.12 signal $x[n]$ is folded and shifted against $h[n]$ while in Figure 2.17 signal $h[n]$ is folded and shifted. The resulting convolution $y[n]$ is exactly the same as expected using the commutation property of convolution.   ∎

2.7   Numerical computation of convolution

Suppose that we wish to compute the convolution $y[n]$ of the finite length sequences $\{h[n]\}_0^{M-1}$ and $\{x[n]\}_0^{N-1}$. For illustration assume that $M = 3$ and $N = 6$. Following the approach illustrated in Figure 2.16, we can see that $y[n] = 0$ for $n < 0$ and $n > L_y = M + N - 1 = 8$. Therefore, the nonzero values of the convolution sum are given by

$$
\begin{aligned}
y[-1] &= h[0]\,0 \quad + h[1]\,0 \quad + h[2]\,0 \qquad &\text{No overlap} \\
y[0] &= h[0]x[0] + h[1]\,0 \quad + h[2]\,0 \qquad &\text{Partial} \\
y[1] &= h[0]x[1] + h[1]x[0] + h[2]\,0 \qquad &\text{overlap} \\
y[2] &= h[0]x[2] + h[1]x[1] + h[2]x[0] \\
y[3] &= h[0]x[3] + h[1]x[2] + h[2]x[1] \qquad &\text{Full} \\
y[4] &= h[0]x[4] + h[1]x[3] + h[2]x[2] \qquad &\text{overlap} \\
y[5] &= h[0]x[5] + h[1]x[4] + h[2]x[3] \\
y[6] &= h[0]\,0 \quad + h[1]x[5] + h[2]x[4] \qquad &\text{Partial} \\
y[7] &= h[0]\,0 \quad + h[1]\,0 \quad + h[2]x[5] \qquad &\text{overlap} \\
y[8] &= h[0]\,0 \quad + h[1]\,0 \quad + h[2]\,0 \qquad &\text{No overlap.}
\end{aligned} \tag{2.70}
$$

This set of equations can be more concisely expressed as a matrix by vector multiplication, as follows

$$
\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \end{bmatrix} = \begin{bmatrix} x[0] & 0 & 0 \\ x[1] & x[0] & 0 \\ x[2] & x[1] & x[0] \\ x[3] & x[2] & x[1] \\ x[4] & x[3] & x[2] \\ x[5] & x[4] & x[3] \\ 0 & x[5] & x[4] \\ 0 & 0 & x[5] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ h[2] \end{bmatrix}. \tag{2.71}
$$

The matrix form of convolution involves a matrix known as *Toeplitz*, because the elements along each diagonal are the same. Computation of convolution as a matrix by vector multiplication is inefficient in terms of storage; however, we shall frequently use it to illustrate various concepts. Equation (2.71) is implemented in MATLAB by `y=convmtx(x,N+M-1)*h`. The convolution matrix is created by `convmtx` which is based on the MATLAB function `toeplitz`.

MATLAB computes the convolution (2.71) using the function

```
y=conv(h,x)
```

where

```
h=[h(1) h(2) ... h(M)]
x=[x(1) x(2) ... x(N)]
y=[y(1) y(2) ... y(M+N-1)].
```

Starting with either (2.70) or (2.71), we can develop two different types of algorithm to compute the convolution summation. The simpler approach, from a programming viewpoint, is to express (2.71) as a linear combination of column vectors:

$$
\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \end{bmatrix} = h[0] \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ 0 \\ 0 \end{bmatrix} + h[1] \begin{bmatrix} 0 \\ x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ 0 \end{bmatrix} + h[2] \begin{bmatrix} 0 \\ 0 \\ x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \end{bmatrix}. \tag{2.72}
$$

This formula expresses the convolution sequence as a superposition of scaled and delayed replicas of the input sequence. It can also be derived from the interpretation shown in Figure 2.17 if we interchange the role of the input and impulse response sequences. This approach can be very efficiently implemented in MATLAB using the vector-oriented function shown in Figure 2.18.

However, we can easily obtain a version with scalar computations by replacing the single loop in `convvec` with a double loop to obtain the function shown in Figure 2.19. This approach, which we use in function `y=convser(h,x)`, can be followed to implement convolution in FORTRAN or C. Functions `convvec` and `convser` provide identical functionality with the MATLAB function `y=conv(h,x)`.

The convolution of two arbitrarily positioned sequences $h[n]$, $n \in [M_1, M_2]$ and $x[n]$, $n \in [N_1, N_2]$, is a sequence $y[n]$, $n \in [M_1 + N_1, M_2 + N_2]$ (see Section 2.6). This result,

```
function y=convvec(h,x)
% Vector computation of y=h*x
M=length(h); N=length(x); h=h(:);
x=x(:); y=zeros(M+N-1,1);
for m=1:M
    y(m:m+N-1)=y(m:m+N-1)+h(m)*x;
end
```

Figure 2.18 Computation of convolution sum using vector operations.

```
function y=convser(h,x)
% Serial computation of y=h*x
M=length(h); N=length(x);
L=M+N-1; y=zeros(L,1);
for m=1:M
    for n=1:N
        k=n+m-1;
        y(k)=y(k)+h(m)*x(n);
    end
end
```

Figure 2.19 Computation of convolution sum using scalar operations.

```
function [y,ny]=conv0(h,nh,x,nx)
ny=[nh(1)+nx(1):nh(end)+nx(end)];
y=conv(h,x);
```

Figure 2.20 Computation of convolution sum along with index calculations.

which holds for any values (positive or negative) of the limits, is easily implemented in MATLAB by the function `[y,ny]=conv0(h,nh,x,nx)` shown in Figure 2.20, where `nh`, `nx`, and `ny` are the index vectors of the corresponding sequences.

From (2.72) it is clear that the computation of convolution requires *MN* multiplications and *MN* additions. All these convolution functions require that the entire sequences to be convolved are available and stored in memory before the processing takes place. The entire output sequence also becomes available after the processing has been completed. This type of processing is known as *block-processing* or *frame-processing*.

2.8 Real-time implementation of FIR filters

In most real-time applications, we wish to compute the output sample *y*[*n*] immediately after the arrival of the input sample *x*[*n*]. This approach, which proceeds on a sample-by-sample basis upon the input sequence, is known as *stream processing*. The computation should be completed before the next input sample comes. That is, the processor should have the processing power to complete all required computations within one sampling period. This is the essence of real-time operation in digital signal processing. The delay, $\tau < T$, between the arrival of an input sample and the generation of the corresponding output sample, is known as *latency*. With respect to convolution, stream processing amounts to computing (2.70) one row at a time, whereas block processing would involve the computation of a fixed-size block of rows, at a time. In block processing, real-time operation means that the processing of one block should be completed before the accumulation of the next block. Clearly, latency in block processing is larger than latency in stream processing.
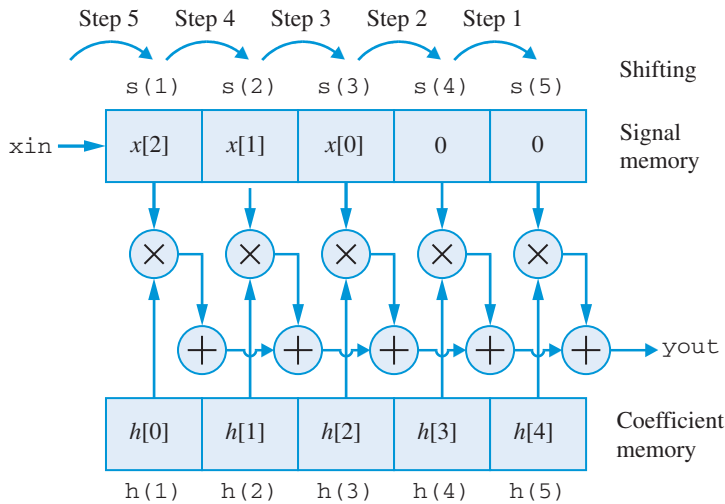
Figure 2.21 The operation of an FIR system.

To evaluate $y[n]$ on a sample-by-sample basis, we can compute each line of (2.70) in MATLAB as a dot product. In FORTRAN or C each dot product has to be determined as a sum of products. We will use this approach to illustrate how to implement an FIR system for stream operation. If we consider a system with impulse response $h[n]$, $0 \le n \le M - 1$, we need M memory locations to store the values $h[0],\dots,h[M - 1]$ and M locations to store the input samples $x[n], \dots, x[n - M + 1]$ required to compute the output sample $y[n]$. In MATLAB we use two vectors with elements `h(1),...,h(M)` and `s(1),...,s(M)`, respectively. The samples of the impulse response are stored before the processing starts, in the same order. However, careful inspection of (2.70) indicates that the samples of the input sequence should be entered into the signal array in reverse order. This is illustrated in Figure 2.21 for $M = 5$. We note that the signal array should be initialized with zeros before the system starts its operation. When the first input sample $x[0]$ arrives, it is stored at location `s(1)`, the sum of products `yout=s(1)*h(1)+...+s(5)*h(5)` is computed, and the value `yout` provides the output sample $y[0] = x[0]h[0]$. Then the contents of the signal memory are shifted to the right, starting with `s(4)`; otherwise, `s(1)` will fill every memory cell. The sample $x[0]$ moves to `s(2)` and $x[1]$ enters `s(1)`. The sum of products is computed and provides the sample $y[1] = x[1]h[0] + x[0]h[1]$. This process is repeated for each new input sample. Figure 2.21 shows the contents of the system memory for $n = 2$. The memory of the system is completely filled with signal samples at $n = M$. The signal memory remains completely filled until the last input sample $x[N - 1]$ enters the system. Thus, for $M \le n \le N - 1$ the output samples are computed exclusively from a weighted sum of input samples. Careful inspection of Figure 2.21 shows that if we start accumulating the products `h(i)*s(i)` from right to left, we can shift the contents of `s(i-1)` to `s(i)` after we have computed and accumulated this product. This single loop "multiply-accumulate-shift" approach is illustrated in the MATLAB script `firstream`, shown in Figure 2.22. The "multiply-accumulate-shift" operation is very important for the real-time implementation of digital filters. Thus, all special purpose digital signal processors perform this operation as a single instruction.

```
% Script file: firstream.m
% FIR filter implementation using stream processing
% Generate an input signal sequence
N=20; ni=(0:N-1); x=(3/4).^ni+0.1*rnd(size(ni));
% Store impulse response
M=5; h=ones(1,M)/M; % M-point Moving Average filter
% Initialize signal memory
s=zeros(1,M);
% Compute filter output sequence
for n=1:N     % Sampling-time index
    xin=x(n); % Get input sample from ADC or storage
    s(1)=xin;
    yout=h(1)*s(1);
    for m=M:-1:2
        yout=yout+h(m)*s(m); % Multiply, Accumulate
        s(m)=s(m-1);         % and Shift Operation
    end
    y(n)=yout; % Put output sample to DAC or storage
end
```

Figure 2.22 MATLAB script illustrating the real-time implementation of an FIR filter.

The MATLAB function `y=filter(h,1,x)` computes the convolution $y[n]$ of the sequences $h[n]$, $0 \leq n \leq M-1$ and $x[n]$, $0 \leq n \leq N-1$, in the same range $0 \leq n \leq N-1$ with the input sequence. In contrast, `y=conv(h,x)` computes the convolution in the full range $0 \leq n \leq N+M-2$.

2.9 FIR spatial filters

As explained in Chapter 1, a black-and-white picture is a signal that describes intensity variation over a spatial region. The sampled version of this picture over a rectangular grid is represented by a 2D *discrete-space* signal $x[m,n]$, $[m,n] \in \{(0, M-1) \times (0, N-1)\}$, which is also known as a *digital image*. Each sample of the digital image is a picture element and hence is called a *pixel*.

Spatial FIR filters are very popular and useful in the processing of digital images to implement visual effects like noise filtering, edge detection, etc. Although digital image processing is not the topic of this book, we will use FIR spatial filters as a visual demonstration of the convolution operation, albeit in two dimensions.

Let us consider the task of smoothing sharp image features, like edges. Images have sharp edges when the local intensity rises or drops sharply and have blurred or
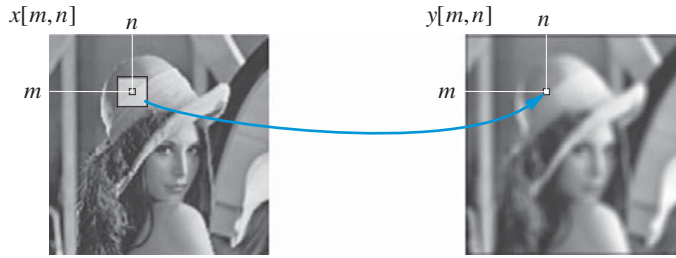
Figure 2.23 The FIR spatial filtering operation.

fuzzy perception when local intensity is smooth. A simple smoothing operation involves replacing each pixel by its average over a local region as shown in Figure 2.23.

Consider a 3×3 region around the pixel $x[m, n]$. Then the smoothed pixel value $y[m, n]$ can be computed as an arithmetic mean of the nine pixels in the local region

$$
\begin{aligned}
y[m, n] = \tfrac{1}{9}(x[m-1, n-1] \quad &+ x[m-1, n] \quad + x[m-1, n+1] \\
+ x[m, n-1] \quad &+ x[m, n] \quad + x[m, n+1] \\
+ x[m+1, n-1] \quad &+ x[m+1, n] \quad + x[m+1, n+1]),
\end{aligned}
\tag{2.73}
$$

which can be written in a compact form as

$$
y[m, n] = \sum_{k=-1}^{1} \sum_{\ell=-1}^{1} \left(\frac{1}{9}\right) x[m-k, n-\ell].
\tag{2.74}
$$

We next define a 2D sequence $h[m, n]$

$$
h[m, n] = \begin{cases} \frac{1}{9}, & -1 \le m, n \le 1 \\ 0, & \text{otherwise} \end{cases}
\tag{2.75}
$$

which can be seen as an FIR spatial filter impulse response. Then we can write (2.74) as

$$
y[m, n] = \sum_{k=-1}^{1} \sum_{\ell=-1}^{1} h[k, \ell] x[m-k, n-\ell],
\tag{2.76}
$$

which is a 2D convolution of image $x[m, n]$ with an FIR spatial filter $h[m, n]$. A general expression for 2D convolution, when the FIR spatial filter has finite symmetric support $(2K + 1) \times (2L + 1)$, is given by

$$
y[m, n] = \sum_{k=-K}^{K} \sum_{\ell=-L}^{L} h[k, \ell] x[m-k, n-\ell].
\tag{2.77}
$$

Figure 2.23 shows the result of a 5×5 smoothing filter operation on the image Lena.
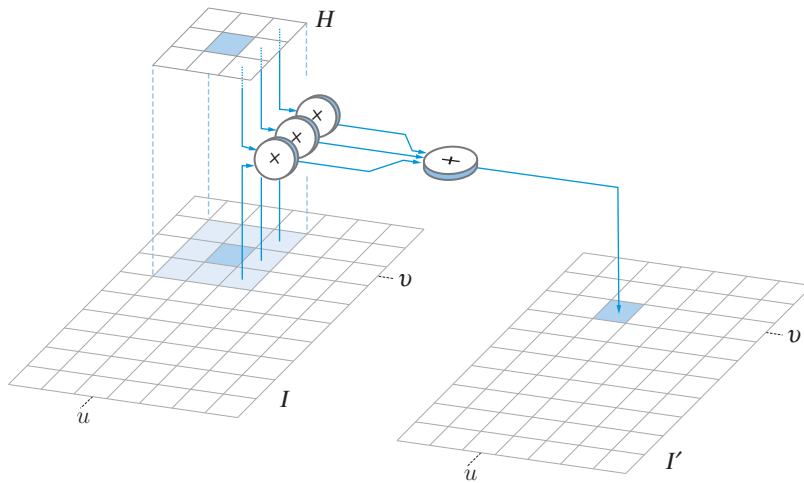
Figure 2.24 FIR spatial filter implementation.

Filter implementation Note that (2.77) can also be written as

$$y[m, n] = \sum_{k=m-K}^{m+K} \sum_{\ell=n-L}^{n+L} x[k, \ell] h[m - k, n - \ell]. \tag{2.78}$$

This suggests the following steps for the computation of convolution at each pixel $[m, n]$:

1. The filter array $h[k, \ell]$ is rotated by $180°$ to obtain $h[-k, -\ell]$ array.
2. The rotated array is moved over the image so that the origin $h[0, 0]$ coincides with the current image pixel $x[m, n]$.
3. All filter coefficients are multiplied with the corresponding image pixels and the results are added.
4. The resulting sum is stored at the current pixel $[m, n]$ in the filtered image $y[m, n]$.

These steps are shown in Figure 2.24 which pictorially illustrates the convolution operation. The MATLAB function `y=conv2(h,x)` implements the 2D convolution operation in (2.78). However, the more suitable function for FIR spatial filtering is `y=filter2(h,x)` which uses the `conv2` function but provides the output sequence `y` with the same size as that of the input sequence `x`. Using different shapes and values for the FIR filter support, various visual effects like motion-blur, edge detection, edge enhancement, etc. can be obtained. These and other issues are examined in Problems 15, 16, and 46.

2.10 Systems described by linear constant-coefficient difference equations

We have shown in Section 2.4 that every linear time-invariant system (1) is uniquely characterized by its impulse response sequence, and (2) its output can be determined

by the convolution of impulse response and input sequences. Unfortunately, the convolution sum of IIR systems cannot be used in practice because it requires an infinite number of arithmetic operations and memory locations. In this section, we introduce a *subclass* of practically realizable IIR linear time-invariant systems, where the output and input sequences are related by a linear constant-coefficient difference equation.

Consider a causal and stable linear time-invariant system with an exponential impulse response sequence

$$h[n] = ba^n u[n]. \quad -1 < a < 1 \tag{2.79}$$

The response to an input sequence $x[n]$, applied at $n = -\infty$, can be written as

$$y[n] = \sum_{k=-\infty}^{n} x[k]h[n-k] = \sum_{k=0}^{\infty} h[k]x[n-k]$$

$$= bx[n] + bax[n-1] + ba^2 x[n-2] + \cdots$$

$$= bx[n] + a(bx[n-1] + bax[n-2] + \cdots).$$

If we recognize that the expression enclosed inside the parentheses is the output value $y[n-1]$, we obtain

$$y[n] = ay[n-1] + bx[n]. \tag{2.80}$$

This equation shows that we can easily compute each output value of the IIR system (2.79) using previously computed output values. This representation is known as a *recursive* implementation of the system (see Figure 2.25). At any time n_0 the value $y[n_0-1]$ contains all the relevant information concerning the past history of the system, which is required to determine the response to any input for $n \geq n_0$. We say that $y[n_0-1]$ constitutes the *state* of the system, and provides the "memory" which separates the future from the past.

Zero-input and zero-state responses The response of the system to an input $x[n]$ applied at $n = 0$ can be obtained either from the convolution sum or the recursive equation (2.80). The response of a causal system to a causal input is given by

$$y[n] = \left(\sum_{k=0}^{n} h[k]x[n-k] \right) u[n]. \tag{2.81}$$
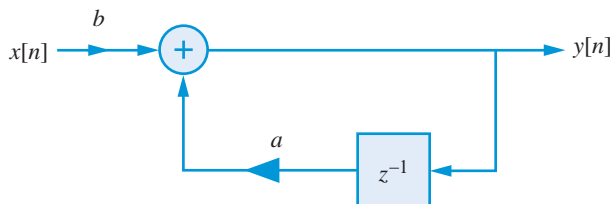
Figure 2.25 Block diagram representation of a simple recursive IIR system.

If we iterate (2.80), starting with $n = 0$, we obtain the output for $n \geq 0$ as follows:

$$y[0] = ay[-1] + bx[0]$$

$$y[1] = ay[0] + bx[1]$$

$$= a^2 y[-1] + bax[0] + bx[1]$$

$$y[2] = ay[1] + bx[2]$$

$$= a^3 y[-1] + ba^2 x[0] + bax[1] + bx[2]$$

$$\vdots$$

$$y[n] = ay[n-1] + bx[n]$$

$$= a^{n+1} y[-1] + ba^n x[0] + ba^{n-1} x[1] + \cdots + bx[n].$$

Using (2.79), the last equation can be written as

$$y[n] = a^{n+1} y[-1] + h[n]x[0] + h[n-1]x[1] + \cdots + h[0]x[n]. \tag{2.82}$$

We see that the output $y[n]$ for $n \geq 0$, depends both on the input $x[n]$ for $n \geq 0$ and the initial condition $y[-1]$. The value of $y[-1]$ summarizes the response of the system to past inputs applied for $n < 0$.

If we set $x[n] = 0$ for $n \geq 0$, we obtain

$$y_{\text{zi}}[n] = a^{n+1} y[-1], \quad n \geq 0 \tag{2.83}$$

which is known as the *zero-input response* of the system. If we assume that $y[-1] = 0$, that is the system is initially at rest or at zero-state, the output is given by

$$y_{\text{zs}}[n] = \sum_{k=0}^{n} h[k]x[n-k], \tag{2.84}$$

which is called the *zero-state response* of the system. Therefore, the total response of the recursive system is given by

$$y[n] = \underbrace{a^{n+1} y[-1]}_{\substack{\text{zero-input} \\ \text{response}}} + \underbrace{\sum_{k=0}^{n} h[k]x[n-k]}_{\substack{\text{zero-state} \\ \text{response}}} = y_{\text{zi}}[n] + y_{\text{zs}}[n]. \tag{2.85}$$

Comparing (2.81) and (2.85) shows that the convolution representation (2.81) and the recursive representation (2.80) of the system (2.79) are identical only if $y[-1] = 0$. Then, the recursive system (2.80) is linear and time-invariant. If $y[-1] \neq 0$, the system is linear in a more general sense that involves linearity with respect to both input and initial conditions.

Steady-state and transient step responses To obtain the step response of the system we set $x[n] = u[n]$ in (2.82). The result is

$$y[n] = \sum_{k=0}^{n} ba^k + a^{n+1}y[-1] = b\frac{1-a^{n+1}}{1-a} + a^{n+1}y[-1], \quad n \geq 0 \tag{2.86}$$

where we have used the geometric summation formula to compute the sum. For a stable system, that is, when $|a| < 1$, we have

$$y_{ss}[n] = \lim_{n\to\infty} y[n] = b\frac{1}{1-a}, \quad n \geq 0 \tag{2.87}$$

which is known as the *steady-state response*. The remaining component

$$y_{tr}[n] = b\frac{-a^{n+1}}{1-a} + a^{n+1}y[-1], \quad n \geq 0 \tag{2.88}$$

which becomes zero as $n \to \infty$ is called the *transient response*. This suggests that the step response of a linear time-invariant system can be decomposed in two different ways as follows:

$$y[n] = \underbrace{\frac{b}{1-a}}_{y_{ss}[n]} + \underbrace{b\frac{-a^{n+1}}{1-a} + a^{n+1}y[-1]}_{y_{tr}[n]} = \underbrace{\frac{b}{1-a} + b\frac{-a^{n+1}}{1-a}}_{y_{zs}[n]} + \underbrace{a^{n+1}y[-1]}_{y_{zi}[n]}. \tag{2.89}$$

In general, we have

$$y_{zi}[n] \neq y_{tr}[n], \tag{2.90}$$

$$y_{ss}[n] \neq y_{zs}[n]. \tag{2.91}$$

If the system is stable $y_{ss}[n] = \lim_{n\to\infty} y_{zs}[n]$. This is illustrated in Figure 2.26 for $0 < a < 1$ and $-1 < a < 0$, respectively. It is important to note how the oscillation of the impulse response, when $-1 < a < 0$, is inflicted on the transient response. We have chosen $b = 1 - a$, so that the output has the same "level" with the input when the system reaches steady state (see Problem 44).

Response to a suddenly applied complex exponential sequence If we set in (2.82) $x[n] = e^{j\omega_0 n}$ and use the geometric summation formula, we obtain

$$y[n] = a^{n+1}y[-1] + e^{j\omega_0 n} \sum_{k=0}^{n} \left(ae^{-j\omega_0}\right)^k$$

$$= a^{n+1}y[-1] + \frac{1 - a^{n+1}e^{-j\omega_0(n+1)}}{1 - ae^{-j\omega_0}} e^{j\omega_0 n}. \tag{2.92}$$

This can be split in two different ways as

$$y[n] = \overbrace{\underbrace{a^{n+1}y[-1]}_{}}^{y_{zi}[n]} + \overbrace{\underbrace{\frac{-a^{n+1}e^{-j\omega_0(n+1)}}{1 - ae^{-j\omega_0}} e^{j\omega_0 n}}_{y_{tr}[n]}}^{y_{zs}[n]} + \overbrace{\underbrace{\frac{1}{1 - ae^{-j\omega_0}} e^{j\omega_0 n}}_{y_{ss}[n]}}^{y_{zs}[n]}. \tag{2.93}$$
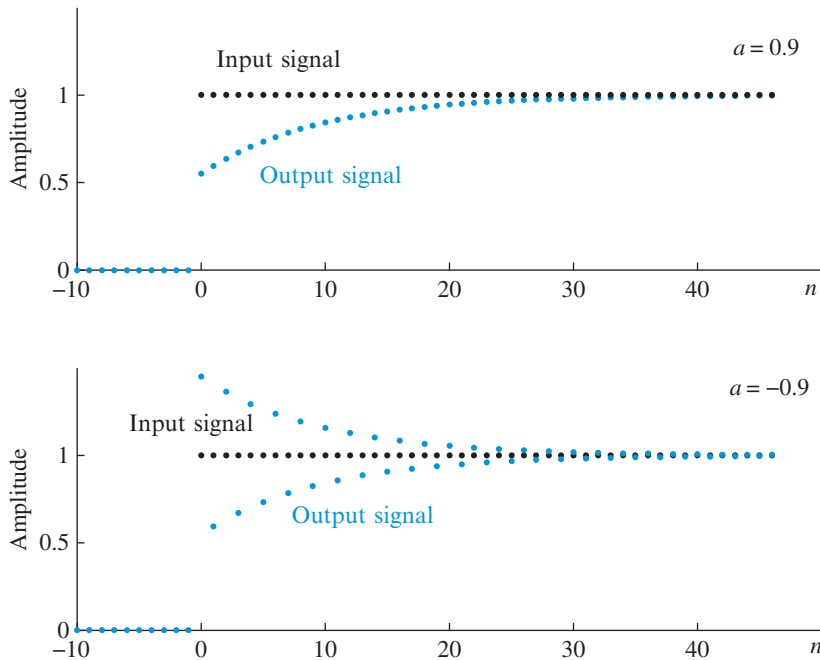
Figure 2.26 Step response of a recursive linear time-invariant system. (For clarity, samples are shown without their stems.)

For a stable system ($|a| < 1$) the transient response decays to zero, with a rate that depends on the value of a (the rate of decay increases as $|a|$ approaches 0). Because the input and output sequences take complex values, we can invoke the principle of superposition and plot separately their real and imaginary parts. The results are shown in Figure 2.27, where we indicate the transient and steady-state intervals of the response.

Careful inspection of Figures 2.26 and 2.27 shows that:

- The steady-state response tracks the input signal.
- The transient-response reveals properties of the system, but eventually dies out if the system is stable.

Linear time-invariant systems used in signal processing applications run for long time periods. The transient effects resulting from nonzero initial conditions and the sudden application of the input are not as important as the steady-state response because they "die-out" quickly. *In this book, unless otherwise stated, we always assume that the initial conditions are zero or equivalently that the system is "initially at rest."* This is easily achieved by initializing, that is, "filling-up," all memory locations of the system with zeros.

General recursive systems These systems are described by the difference equation

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k], \tag{2.94}$$
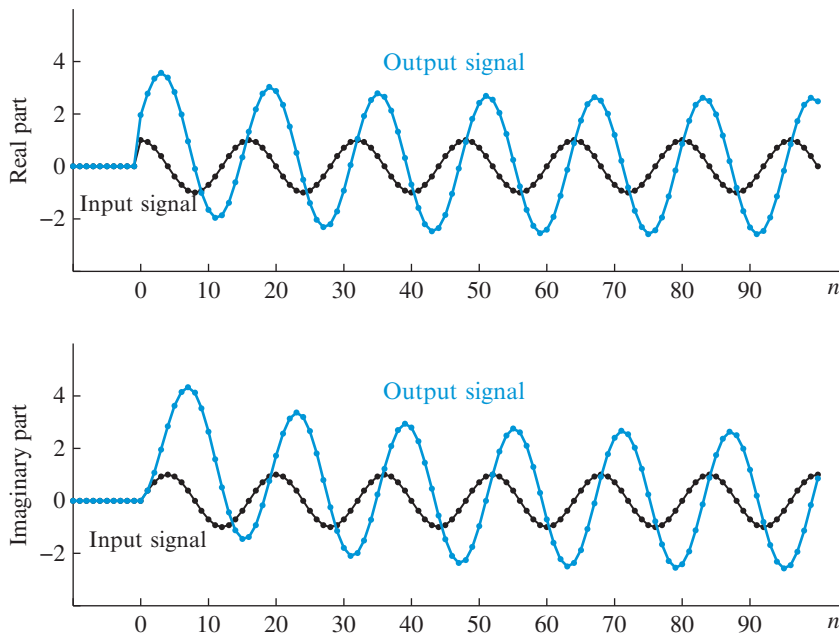
Figure 2.27 Transient and steady-state responses to a suddenly applied complex exponential sequence.

which is known as a *linear constant-coefficient difference equation* (LCCDE). If the *feedback* coefficients a_k and the *feedforward* coefficients b_k are fixed, the system is time-invariant; if they depend on n the system is time-varying. The number N is known as the *order* of the system. For $N = 0$ we have

$$y[n] = \sum_{k=0}^{M} b_k x[n-k],$$ (2.95)

which is a nonrecursive system with finite duration impulse response $h[n] = b_n$ for $0 \le n \le M$ and zero elsewhere. This system is linear and time-invariant. Nonrecursive systems are FIR but there are FIR systems which can be implemented recursively. This is illustrated in the following example.

Example 2.7 Recursive FIR system
The moving average filter

$$y[n] = \frac{1}{M+1} \sum_{k=0}^{M} x[n-k]$$ (2.96)

is FIR with impulse response

$$h[n] = \begin{cases} \dfrac{1}{M+1}, & 0 \le n \le M \\ 0. & \text{elsewhere} \end{cases} \tag{2.97}$$

This system can be implemented nonrecursively using equation (2.96). However, a simple algebraic manipulation gives

$$y[n] = y[n-1] + \frac{1}{M+1}\{x[n] - x[n-1-M]\}, \tag{2.98}$$

which leads to a recursive implementation. In a nonrecursive system we can skip the computation of a sample, say $y[n_0]$, and still be able to compute $y[n]$ for $n > n_0$. This is not possible for recursive systems. ∎

Computation of difference equations The implementation of systems described by the LCCDE (2.94) can be done using the stream processing approach explained in Section 2.7. This requires two structures of the form shown in Figure 2.21; one for the feedforward part and one for the feedback part of the difference equation (see Problem 47). The MATLAB function `filter`, provides an efficient implementation of (2.94). The input and output arguments of this function are specified as follows

$$\texttt{y=filter(b,a,x),}$$

where

$$\texttt{b = [b0 b1 ... bM]} = [b_0\ b_1 \ldots b_M]$$

$$\texttt{a = [1 a1 ... aN]} = [1\ a_1 \ldots a_N]$$

$$\texttt{x = [x(1) x(2) ... x(L)]} = [x[0]\ x[1] \ldots x[L-1]]$$

$$\texttt{y = [y(1) y(2) ... y(L)]} = [y[0]\ y[1] \ldots y[L-1]].$$

For example, the statement `y=filter(1,[1 -0.9],ones(50,1))` computes the first 50 samples of the zero-state response of the filter $y[n] = 0.9y[n-1] + u[n]$. There are two important observations concerning the function `y=filter(b,a,x)`:

- First, the feedback coefficients enter in the parameter vector $\texttt{a} = [1\ a_1 \ldots a_N]$ with their sign reversed. This is because MATLAB assumes that all feedback terms in (2.94) have been moved on the left hand side as follows:

$$y[n] + \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]. \tag{2.99}$$

- Second, the output sequence is computed at the same time interval as the input sequence.
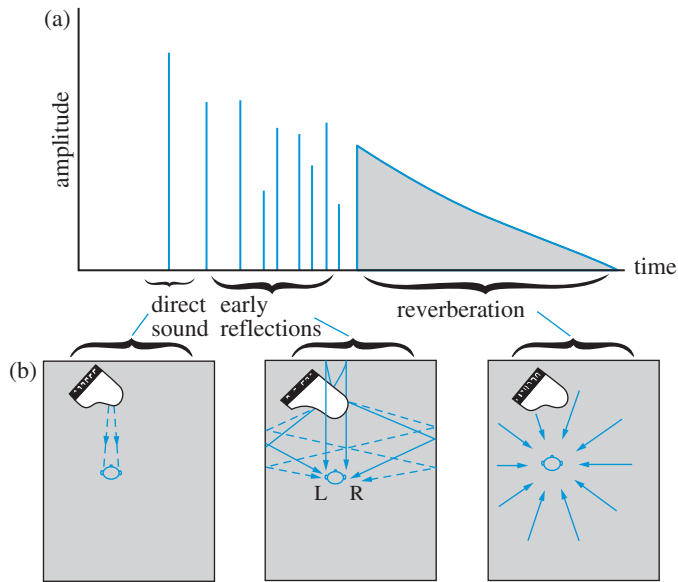
Figure 2.28 (a) Simplified separation of a room's impulse response into perceptually relevant sections. (b) Sound propagation paths, from an instrument to a listener, responsible for each section of the impulse response. Adapted from Bloom (1985).

The algorithm used in the implementation of (2.94) in function `y=filter(b,a,x)` is described in Chapter 9. Additionally, the functions `impz` and `stepz`, based on the `filter` function, can be used to compute the impulse response and the step response of an LTI system, respectively.

Example 2.8   Echo generation and reverberation

When music is performed in a concert hall, a torrent of echoes from the various surfaces in the room strikes the ear, producing the impression of space to the listener. More specifically, the sound reaching the listener consists of several components: direct sound, early reflections, and reverberations. The early reflections correspond to the first few reflections off the wall, whereas the reverberation is composed of densely packed late reflections (see Figure 2.28). Music recorded in an almost anechoic studio, using microphones placed close to the instruments, and played at home or in a car does not sound natural. The typical solution to this problem is to create and add some amount of artificial reverberation to the original recording before distribution.

A single echo is easily generated using the FIR filter

$$y[n] = x[n] + ax[n - D], \quad -1 < a < 1 \tag{2.100}$$

where $x[n]$ is the original signal, D is the round-trip delay in number of sampling intervals, and a is the attenuation factor due to propagation and reflection. If the delay $\tau = D/F_s$ is greater than approximately 40 ms, an echo will be heard. A second echo will be given by

$a^2x[n-2D]$, a third by $a^3x[n-3D]$, and so on. Therefore, a multiple echo generating FIR filter is given by

$$y[n] = x[n] + ax[n-D] + a^2x[n-2D] + a^3x[n-3D] + \cdots, \qquad (2.101)$$

which has impulse response

$$h[n] = \delta[n] + a\delta[n-D] + a^2\delta[n-2D] + a^3\delta[n-3D] + \cdots \qquad (2.102)$$

This filter generates an infinite sequence of echoes having exponentially decaying amplitudes and spaced D sampling periods apart. A more efficient recursive implementation is given by (see derivation of (2.69))

$$y[n] = ay[n-D] + x[n]. \quad -1 < a < 1 \qquad (2.103)$$

The condition $-1 < a < 1$ assures the stability of the system. The implementation of (2.103) using MATLAB and its effects on speech signals is the subject of Tutorial Problem 19. Such simple filters provide the basic building blocks of more sophisticated digital reverberators (see Sections 5.7 and 5.9). ∎

For the first-order recursive system with zero-initial conditions we were able to determine analytically its impulse response and to show that the system can be described by a convolution sum. Then, we used the impulse response to find for what values of the parameter a the system is stable (the parameter b does not affect the stability of the system).

In general, given a system described by the LCCDE (2.94), we wish to be able to address the following issues:

1. Prove that the system is linear time-invariant.
2. Determine analytically the impulse response of the system.
3. Given an analytical expression for the input $x[n]$ find an analytical expression for the output $y[n]$.
4. Given the coefficients $\{a_k, b_k\}$ determine if the system is stable.

The z-transform, to be discussed in Chapter 3, provides an elegant and powerful mathematical tool to deal with these issues.

2.11   Continuous-time LTI systems

In this section we provide a concise introduction to continuous-time LTI systems. The results obtained show that the nature of the time variable (continuous or discrete) is a less fundamental characteristic of a system than the properties of linearity and time-invariance. The adopted approach parallels and builds upon the material developed for discrete-time systems.

We have shown that every discrete-time system that satisfies the linearity and time-invariance constraints can be represented by the convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]. \qquad (2.104)$$

For continuous-time signals, the natural equivalent of the convolution sum is the *convolution integral*

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\mathrm{d}\tau, \qquad (2.105)$$

where $y(t)$, $x(t)$, and $h(t)$ are continuous-time signals.

Graphical interpretation of convolution   We shall now explain the nature of the convolution operation (2.105) using the signals $h(t)$ and $x(t)$, shown in Figure 2.29. The mechanism is similar to the one described in Section 2.4 for discrete-time signals. The most crucial point to keep in mind is that integration is performed with respect to τ; hence, τ is "washed-out" and the result $y(t)$ of convolution is a function of t. The graphical computation of the convolution integral involves the following steps:

1. Replace t by τ and plot the functions $h(t)$ and $x(t)$ as a function of τ, not t.
2. Keep the function $x(\tau)$ fixed.
3. Think of $h(\tau)$ as a rigid wire frame, and flip this frame about the vertical axis ($\tau = 0$) to obtain $h(-\tau)$.
4. Shift the flipped frame by t_0 seconds to obtain the function $h(t_0 - \tau)$. The value $h(0)$ is located at $\tau = t_0$, that is, at the point where the argument of $h(t-\tau)$ equals zero. Therefore, $h(-\tau)$ is shifted to the right when $t_0 > 0$ and to the left when $t_0 < 0$.
5. The area under the product of $x(\tau)$ and $h(t_0 - \tau)$ is $y(t_0)$, the value of the convolution at $t = t_0$.
6. Repeat this procedure, shifting the frame $h(-\tau)$ by different amounts (positive or negative) to evaluate $y(t)$ for all values of t.

These steps are illustrated in Figure 2.29, where we show the product of the integrand $x(\tau)h(t-\tau)$ for three values of the parameter t. The value of the convolution integral is the area under this curve. Since for different values of t, the curve $h(t-\tau)$ takes various positions along the τ-axis, the shape of the function $x(\tau)h(t-\tau)$ and the area under its curve change as a continuous function of t. The result is the convolution function $y(t)$.

The unit impulse function   The convolution sum (2.104) completely characterizes a discrete-time system through its impulse response $h[n]$. The sequence $h[n]$ is the response of the system to the unit impulse sequence

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0. & n \neq 0 \end{cases} \qquad (2.106)$$
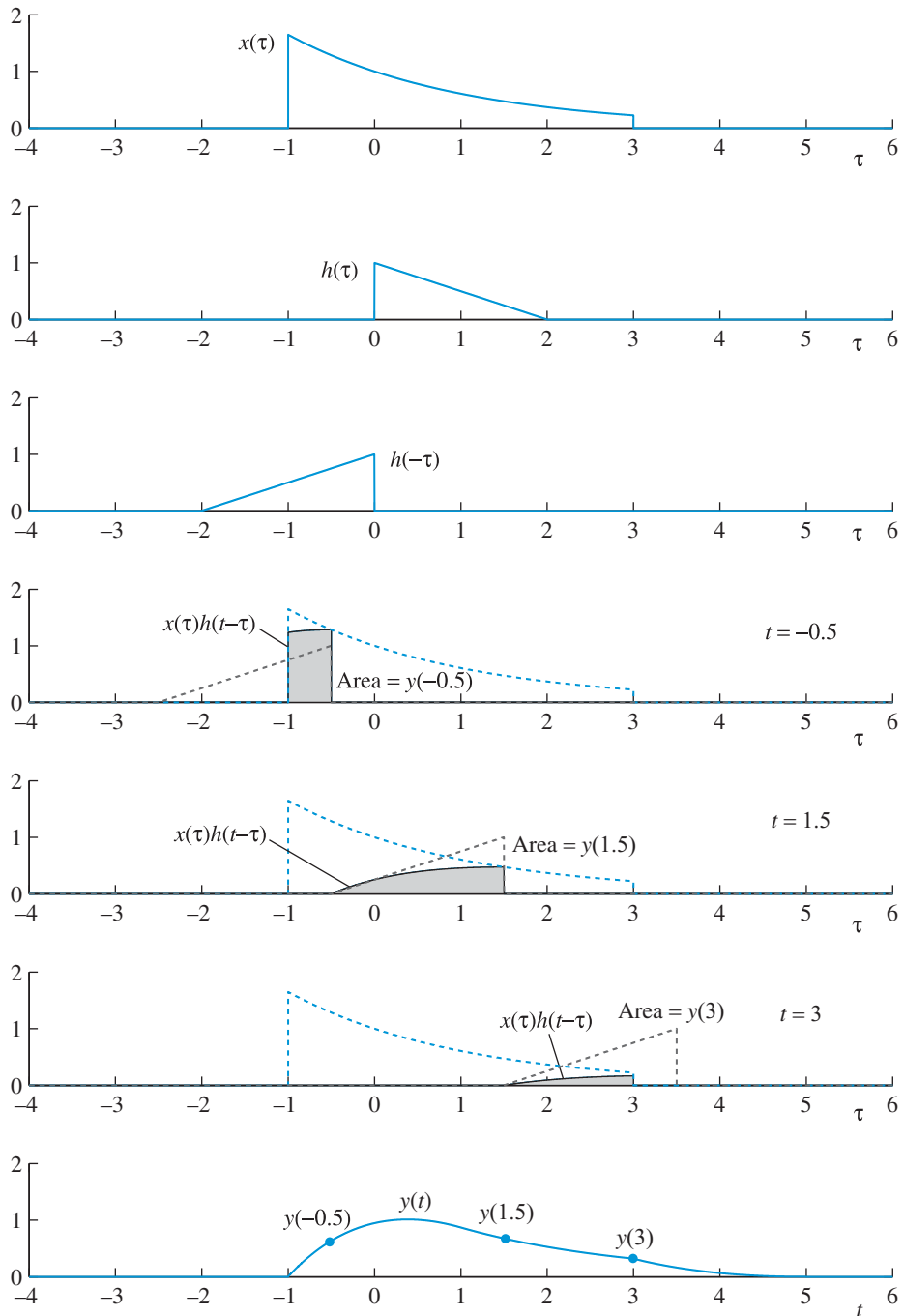
Figure 2.29 Graphical illustration of the convolution operation steps: folding, shifting, multiplication, and integration.

To develop a similar representation for continuous-time LTI systems, we need to define a "continuous-time impulse." Unfortunately, a definition like

$$\delta(t) = \begin{cases} 1, & t = 0 \\ 0, & t \neq 0 \end{cases} \tag{2.107}$$

would not work because the the signal $\delta(t)$ has zero energy. It turns out that the definition of a continuous-time impulse function is a difficult mathematical problem. In fact, $\delta(t)$ belongs to a family of functions known as *distributions* or *generalized functions*.

While ordinary functions are defined by assigning values to the independent variable, generalized functions are defined by their effect, that is, in terms of what they "do," to a test signal. To develop such an operational definition of $\delta(t)$, we consider the convolution of an arbitrary signal $x(t)$ and a narrow rectangular pulse

$$\delta_\Delta(t) = \begin{cases} 1/\Delta, & -\Delta/2 < t < \Delta/2 \\ 0, & \text{otherwise} \end{cases} \tag{2.108}$$

with unit area, that is, $\int \delta_\Delta(t)\mathrm{d}t = 1$. We note that as $\Delta \to 0$, the pulse becomes narrower but taller; however, the area always remains equal to one. To evaluate the convolution integral

$$y(t) = \int_{-\infty}^{\infty} x(\tau)\delta_\Delta(t - \tau)\mathrm{d}\tau \tag{2.109}$$

we center the folded pulse at $\tau = t$ to create $\delta_\Delta(t - \tau)$ and we multiply by $x(\tau)$ to create the integrand $x(\tau)\delta_\Delta(t - \tau)$ (see Figure 2.30). The product $x(\tau)\delta_\Delta(t - \tau)$ is zero except in the interval $t - \Delta/2 < \tau < t + \Delta/2$. If the pulse is narrow enough and $x(\tau)$ is smooth enough within this interval, we have approximately

$$x(\tau)\delta_\Delta(t - \tau) \approx x(t)\delta_\Delta(t - \tau). \tag{2.110}$$
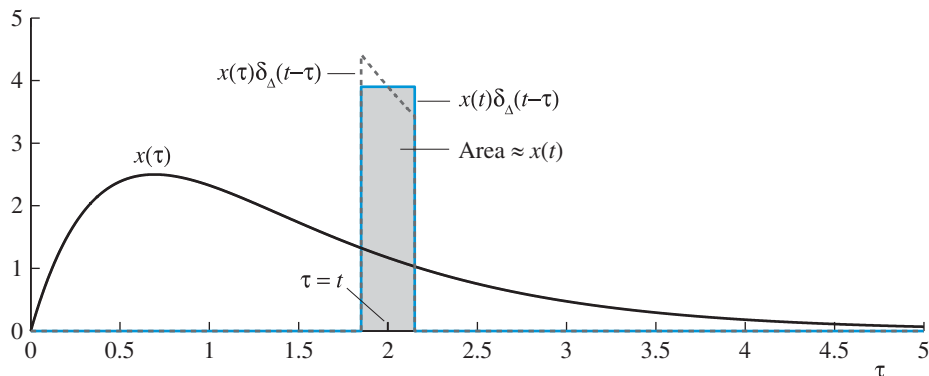
Figure 2.30 Interpretation of convolution by a narrow pulse as a scanning operation.

Substituting into the convolution integral (2.109), we obtain

$$y(t) = \int_{-\infty}^{\infty} x(\tau)\delta_\Delta(t-\tau)\mathrm{d}\tau \approx x(t)\int_{-\infty}^{\infty} \delta_\Delta(t-\tau)\mathrm{d}\tau = x(t). \qquad (2.111)$$

This approximation improves as the duration Δ of the pulse decreases.

Suppose now that there is an ideal impulse function $\delta(t)$ that makes the approximation (2.111) exact, that is

$$\int_{-\infty}^{\infty} x(\tau)\delta(t-\tau)\mathrm{d}\tau = x(t). \qquad (2.112)$$

We define the *unit impulse* $\delta(t)$ as the signal which, for any $x(t)$, satisfies

$$x(t) * \delta(t) = x(t). \qquad (2.113)$$

Thus, the unit impulse convolved with any function reproduces that function. In this sense, $\delta(t)$ is the identity element of the convolution operation. The *operational definition* (2.113) can be used to derive all properties of $\delta(t)$ in a consistent manner.

If we multiply a signal $x(t)$ with a narrow pulse $\delta_\Delta(t-t_0)$, centered at $t = t_0$, we obtain the approximate relation

$$x(t)\delta_\Delta(t-t_0) \approx x(t_0)\delta_\Delta(t-t_0). \qquad (2.114)$$

The area under $x(t_0)\delta_\Delta(t-t_0)$ is equal to $x(t_0)$. The exact version of (2.114) is

$$x(t)\delta(t-t_0) = x(t_0)\delta(t-t_0). \qquad (2.115)$$

The actual value of $x(t)$ at $t = t_0$ is provided by the area of $x(t)\delta(t-t_0)$. Indeed

$$\int_{-\infty}^{\infty} x(t)\delta(t-t_0)\mathrm{d}t = x(t_0)\int_{-\infty}^{\infty} \delta(t-t_0)\mathrm{d}t = x(t_0). \qquad (2.116)$$

Equation (2.116) is known as the sampling property of $\delta(t)$ because the impulse picks the value of $x(t)$ at $t = t_0$.

Impulse response and convolution If we interpret the definition (2.112) as a decomposition of $x(t)$ into a chain of scaled and shifted impulses, we can use the linearity and time-invariance properties to obtain a convolution description of continuous-time LTI systems. Indeed, if we denote by $h(t)$ the response of the system to the impulse $\delta(t)$ (see Figure 2.31), we have
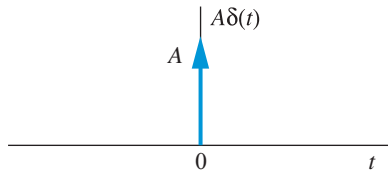
Figure 2.31 Symbolic representation of the continuous-time unit impulse.

$$\delta(t) \xmapsto{\mathcal{H}} h(t) \qquad \text{(Impulse Response)}$$

$$\delta(t - \tau) \xmapsto{\mathcal{H}} h(t - \tau) \qquad \text{(Time} - \text{Invariance)}$$

$$x(\tau)\delta(t - \tau) \xmapsto{\mathcal{H}} x(\tau)h(t - \tau) \qquad \text{(Homogeneity)}$$

$$\underbrace{\int_{-\infty}^{\infty} x(\tau)\delta(t - \tau)\mathrm{d}\tau}_{x(t)} \xmapsto{\mathcal{H}} \underbrace{\int_{-\infty}^{\infty} x(\tau)h(t - \tau)\mathrm{d}\tau}_{y(t)}, \quad \text{(Additivity)}$$

which leads to (2.105). This is similar to the derivation for discrete-time systems; the only difference is that, due to the continuity of time, summation has been replaced by integration in the additivity step of the linearity property.

As in the discrete-time case, a continuous-time LTI system is completely characterized by its impulse response $h(t)$. Indeed, it can be easily shown that an LTI system is causal if its impulse response satisfies the condition

$$h(t) = 0, \quad t < 0 \qquad (2.117)$$

and stable if its impulse response is absolutely integrable, that is, if

$$\int_{-\infty}^{\infty} |h(t)|\mathrm{d}t < \infty. \qquad (2.118)$$

The main implication of the time variable continuity is the replacement of the summation operation by integration in (2.50) and (2.52).

Continuous-time systems in MATLAB The description of continuous-time systems requires the use of continuous functions. Therefore, their analysis in MATLAB can be done *only* approximately. The convolution integral (2.105) is usually approximated using numerical integration techniques (see Problem 36). Symbolic computation is also possible if $x(t)$ and $h(t)$ are specified by simple equations.

Learning summary

- Linearity makes it possible to characterize a system in terms of the responses $h_k[n]$ to the shifted impulses $\delta[n-k]$ for all k, whereas time-invariance implies that $h_k[n] = h[n-k]$. The combination of linearity and time-invariance allows the complete characterization of a system by its impulse response $h[n]$.

- The impulse response $h[n]$ of an LTI system can be used to compute the output of the system for any input via the convolution sum and check whether the system is causal and stable.

  - Input-output description: $\quad y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$

  - Stability: $\quad \sum_{n=-\infty}^{\infty} |h[k]| < \infty$

  - Causality: $\quad h[n] = 0$ for $n < 0$.

- The subclass of linear time-invariant systems, which are realizable in practice, is described by linear constant-coefficient difference equations

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k].$$

If all feedback coefficients a_k are zero, we have a system with a finite duration impulse response (FIR), which is usually implemented nonrecursively. If at least one of a_k are nonzero, we have a recursive system with an infinite duration impulse response (IIR). In most signal processing applications, we assume that systems described by difference equations are initially at rest, that is, the initial conditions $y[-1], \ldots, y[-N]$ are set to zero. In the next chapter, we introduce a new tool, the z-transform, and use it to analyze linear time-invariant systems.

- Continuous-time LTI systems are completely characterized, like discrete-time systems, by their impulse response $h(t)$. Simply, the convolution sum is replaced by the convolution integral and the conditions for stability and causality are modified in an obvious manner. Practically realizable continuous-time LTI systems are described by linear constant coefficient differential equations.

TERMS AND CONCEPTS

Additivity property A property of a system in which a sum of input produces the corresponding sum of outputs, that is, $\mathcal{H}\{x_1[n] + x_2[n]\} = \mathcal{H}\{x_1[n]\} + \mathcal{H}\{x_2[n]\}$.

Bounded signal A signal $x[n]$ is bounded if there exists a positive constant M such that $|x[n]| \leq M$ for all n.

Causal system A system whose present value of its output does not depend on future values

of its input. An LTI system is causal if its impulse response is zero for $n < 0$.

Convolution An operation that produces the output of an LTI system to any arbitrary input using system impulse response. For discrete-time systems, it is given by a summation operation and for continuous-time systems, it is given by an integral operation.

Discrete-time signal A signal whose value $x[n]$ is defined for every value of the integer variable n, also called a sequence.

Discrete-time system A system which transforms a discrete-time input signal $x[n]$ into a discrete-time output signal $y[n]$. Mathematically, it is described by $y[n] = \mathcal{H}\{x[n]\}$.

Dynamic system A system whose output $y[n]$ for every n depends on its inputs and outputs at other times.

Elementary signals Simple signals like unit sample, unit step, etc., that are useful in representation and analysis.

Energy of a signal The quantity $\sum_{-\infty}^{\infty} |x[n]|^2$ is defined as the signal energy and denoted by $\mathcal{E}_x$.

FIR system An LTI system characterized by a finite (duration) impulse response.

Fundamental period The smallest value N with respect to which a periodic signals repeats itself.

Homogeneity property A property of a system in which a scaled input produces the corresponding scaled output, that is, $\mathcal{H}\{a\,x[n]\} = a\mathcal{H}\{x[n]\}$.

Impulse response Response of an LTI system to the unit sample signal. It is denoted by $h[n]$.

IIR system An LTI system characterized by an infinite (duration) impulse response.

LCCDE A linear constant-coefficient difference equation relating a linear combination of the present and past outputs to a linear combination of the present and past inputs. An LTI system can be described as an LCCDE.

Linear system A system that satisfies the properties of homogeneity and additivity, that is, the principle of superposition.

LTI system A system that is both linear and time invariant. It is completely characterized by its impulse response.

Memoryless system A system whose output $y[n]$ for every n depends only on its input $x[n]$ at the same time.

Noncausal system A system whose output depends on future values of its input.

Nonrecursive system A system whose output at each n cannot be computed from its previously computed output values. Nonrecursive systems are FIR systems.

Periodic signal A signal $x[n] = x[n + N]$ that repeats every $N > 0$ samples for all n.

Power of a signal The quantity $\lim_{L\to\infty} \frac{\mathcal{E}_x}{L}$ is defined as the signal power and denoted by $\mathcal{P}_x$.

Principle of superposition A property of a system in which a linear combination of inputs produces a corresponding linear combination of outputs, that is, $\mathcal{H}\{a_1x_1[n] + a_2x_2[n]\} = a_1\mathcal{H}\{x_1[n]\} + a_2\mathcal{H}\{x_2[n]\}$.

Practically realizable system A discrete-time system is practically realizable if its practical implementation requires a finite amount of memory and a finite number of arithmetic operations.

Recursive system A system whose output at each n can be computed from its previously computed output values. Recursive systems are IIR systems.

Sampling period or interval The time interval between consecutive samples of a discrete-time signal.

Sampling rate or frequency The number of samples per second obtained during periodic sampling.

(BIBO) Stable system A system that produces bounded output for every bounded input. An LTI system is BIBO stable if its impulse response is absolutely summable.

State of a system The relevant information at $n = n_0$, concerning the past history of the system, which is required to determine the output to any input for $n \geq n_0$.

Steady-state response A response of a stable LTI system that continues or persists as $n \to \infty$. It is either a constant or sinusoidal in nature.

Step response Response of an LTI system to the unit step signal.

Terms and concepts

Time invariant (or fixed) system A system whose input/output pairs are invariant to a shift in time, that is, a time-shifted input produces a corresponding time-shifted output.

Transient response A response of an LTI system that decays to zero as $n \to \infty$.

Zero-input response A response of an LTI system due to initial conditions when no input has been applied.

Zero-state response A response of an LTI system due to an applied input when no initial conditions are present.

MATLAB functions and scripts

Name	Description	Page
conv	Computation of convolution sequence	56
conv0*	Compute convolution and its support	57
conv2	Convolution of 2D sequences	61
convmtx	Convolution matrix	55
convser	Serial computation of convolution	57
convvec	Vector computation of convolution	56
delta*	Generate unit sample sequence	28
filter	Implementation of a difference equation	67
filter2	Implementation of 2D FIR spatial filter	61
firstream*	Real-time FIR filter simulation	59
fold*	Fold or flip a sequence	29
impz	Computation of impulse response	68
persegen*	Generate periodic sequence	28
plot	General plotting function	30
pulsetrain	Generate a pulse train	80
shift*	Shift a sequence by n_0 samples	29
stem	Plot a sequence	30
stepz	Computation of step response	68
sound	Playing of audio signals	30
timealign*	Create sequences with the same support	29
unitpulse*	Generate unit pulse sequence	28
unitstep*	Generate unit step sequence	28
wavread	Read a wave audio file	30
wavwrite	Write a wave audio file	30

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

- Oppenheim *et al.* (1997) and Haykin and Van Veen (2003) have a parallel treatment of discrete-time and continuous-time signals and systems at a level comparable to that of this text.

- Proakis and Manolakis (2007) provides a more detailed discussion of discrete-time signals and systems from a digital signal processing perspective; in contrast Oppenheim and Schafer (2010) has a more concise treatment.

- A thorough and well organized introduction to MATLAB is provided in Hanselman and Littlefield (2005). Van Loan (2000) includes a nice introduction to MATLAB with emphasis on numerical computation and related graphics.

- The real-time implementation of linear constant-coefficient difference equations using floating-point or fixed-point digital signal processors is discussed in Kuo and Gan (2005). The software development is based on C, C++, or assembly language programming.

- A thorough treatment of continuous-time LTI systems and the delta function is provided in Oppenheim *et al.* (1997). Bracewell (2000) provides an illuminating discussion of linearity, time-invariance and convolution.

Review questions

1. Describe various ways discrete-time signals can be specified.
2. Define energy and power of discrete-time signals.
3. Why are elementary signals useful? Describe a few of these signals.
4. Can sinusoidal sequences be always periodic? Explain.
5. Describe signal operations used in signal manipulations.
6. Define a causal and stable system. Why are these properties needed?
7. What are the two basic properties of a linear system.
8. Define time-invariance and explain its usefulness.
9. How many building blocks are needed to implement linear, time-invariant systems? Describe these blocks.
10. When is a system practically realizable?
11. Every signal can be described by a linear combination of scaled and shifted unit samples. True or false? Explain.
12. A linear, time-invariant system can be completely characterized by its response to a particular elementary signal. What is this signal and what is the resulting response called?
13. A linear, time-invariant system can be completely characterized by its response to a unit impulse sequence. Why is this true? How do we obtain response to any arbitrary sequence?
14. Explain "convolution as a scanning" operation and "convolution as a superposition of scaled and shifted replicas" operation.
15. What are FIR systems? IIR systems?

16. Explain the difference between recursive and nonrecursive systems.
17. IIR systems are always implemented as recursive systems. True or false? Explain.
18. FIR systems can only be implemented as nonrecursive systems. True or false? Explain.
19. Describe the important properties of convolution.
20. Define causality and stability of an LTI system in terms of its impulse response.
21. Response of a stable LTI system to a periodic input is periodic. Explain.
22. Response of an LTI system to a complex exponential signal is also a complex exponential. True or false. Explain.
23. Describe the difference equation representation of an LTI system. Can every LTI system be described this way?
24. Explain the difference between zero-input and zero-state responses.
25. Explain the difference between steady-state and transient step responses.
26. Explain the difference between zero-input and transient responses. Between zero-state and steady-state responses.

Problems

Tutorial problems

1. Write a MATLAB script to generate and plot the following signals described in Section 2.1, for $-20 \leq n \leq 40$.
   (a) unit sample $\delta[n]$,
   (b) unit step $u[n]$,
   (c) real exponential signal $x_1[n] = (0.80)^n$,
   (d) complex exponential signal
       $x_2[n] = (0.9e^{j\pi/10})^n$, and
   (e) sinusoidal sequence
       $x_3[n] = 2\cos[2\pi(0.3)n + \pi/3]$.
   Since $x_2[n]$ is complex-valued, plot the real part, imaginary part, magnitude, and phase using the function `subplot`.

2. Let $x[n] = \{5, 4, 3, 2, 1\}$. This problem examines the commutativity of the folding and shifting operations. Consider a new sequence $x[2 - n] = x[-(n - 2)]$.
   (a) Let $y_1[n]$ be obtained by first folding $x[n]$ and then shifting the result to the right by two samples. Determine and plot $y_1[n]$.
   (b) Let $y_2[n]$ be obtained by first shifting $x[n]$ to the right by two samples and then folding the result. Determine and plot $y_2[n]$.
   (c) From your plots are $y_1[n]$ and $y_2[n]$ the same signals? Which signal represents the correct $x[2 - n]$ signal?

3. Let $x[n] = \{-1, 0, 1, 2, 3, 4, 4, 4, 4, 4\}$.
   (a) Determine the sequences $x[-n]$, $x[n - 3]$, and $x[n + 2]$ by hand.
   (b) Determine the sequences in (a) using the `fold` and `shift` functions.
   (c) Plot the sequences $x[n], x[-n], x[n - 3]$, and $x[n + 2]$ using the function `stem`.

4. Use function `repmat` to generate 5 periods of a periodic sequence with a single period defined by (a) $\{1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\}$ (b) $\cos(0.1\pi n), 0 \leq n \leq 9$ (c) $0.8^n, 0 \leq n \leq 9$. Repeat with functions `presegen` and `pulsetrain`.

5. The sinusoidal signal $\cos(\omega_0 n + \theta_0)$ is periodic in n if the normalized frequency $f_0 \triangleq \frac{\omega_0}{2\pi}$ is a rational number, that is, $f_0 = \frac{M}{N}$, where M and N are integers.

   (a) Prove the above result.

   (b) Generate and plot (use the `stem` function) $\cos(0.1n - \pi/5), -20 \leq n \leq 20$. Is this sequence periodic? Can you conclude periodicity from the plot?

   (c) Generate and plot (use the `stem` function) $\cos(0.1\pi n - \pi/5), -10 \leq n \leq 20$. Is this sequence periodic? If it is, what is the fundamental period. What interpretation can you give to the integers M and N?

6. This problem uses the sound file "handel" available in MATLAB. This sound is sampled at $F_s = 8192$ samples per second using 8-bits per sample.

   (a) Load the sound waveform "handel" in an array `x` and listen to it using the `sound` function at the full sampling rate.

   (b) Select every other sample in `x` which reduces the sampling rate by a factor of two. Now listen to the new sound array using the `sound` function at half the sampling rate.

   (c) Select every fourth sample in `x` which reduces the sampling rate by a factor of four. Listen to the resulting sound array using the `sound` function at quarter the sampling rate.

   (d) Save the generated sound in part (c) using the `wavwrite` function.

7. Compute and plot the response of the following systems:

$$y[n] = \frac{n}{n+1}y[n-1] + x[n], \quad y[-1] = 0$$

$$y[n] = 0.9y[n-1] + x[n], \quad y[-1] = 0$$

   to the inputs $x[n] = \delta[n]$ and $x[n] = \delta[n-5]$, for $0 \leq n \leq 20$, and comment upon the obtained results.

8. A 5-point moving average filter computes a simple average over five input samples at each n.

   (a) Determine the difference equation for this filter.

   (b) Determine and plot the impulse response $h[n]$.

   (c) Draw the system block diagram.

9. A one-sided exponential sequence of the form $a^n u[n]$, where a is an arbitrary (real- or complex-valued) constant, is called a geometric sequence.

   (a) Show that the sum of the samples of the geometric sequence is given by

$$\sum_{n=0}^{\infty} a^n = \frac{1}{1-a}. \quad \text{for } |a| < 1 \tag{2.119}$$

   (b) Show that the finite sum of its first N terms is given by N if $a = 1$ and by

$$\sum_{n=0}^{N-1} a^n = \frac{1-a^N}{1-a}. \quad a \neq 1 \tag{2.120}$$

10. The input $x[n] = \{1, 3, 2, -1\}$ is applied to the LTI system described by the impulse
response $h[n] = 2(0.8)^n$, $0 \le n \le 6$.

    (a) Using the convolution as a superposition of scaled and shifted replicas, determine
$y[3]$.

    (b) Illustrate the above calculation graphically.

11. A system is implemented by the statements

```
y1=conv(ones(1,5),x);
y2=conv([1 -1 -1 -1 1],x);
y=conv(ones(1,3),y1+y2);
```

    (a) Determine the impulse response of the equivalent system `y=conv(h,x)`.

    (b) Compute and compare the step responses of the two equivalent system representations.

12. Use the function `convmtx` to compute the convolution of the finite length sequences
in (2.38) and (2.39) using a matrix by vector multiplication.

13. Show that the response of a stable linear time-invariant system tends asymptotically to
zero after the input is "turned-off," that is, when $x[n] = 0$ for $n \ge n_0$.

14. Explain how to use the function `y=conv(h,x)` to compute the response of a noncausal
system to an input applied at $n = 0$. Assume that the system becomes causal if we
delay the impulse response by n_0 samples.

15. In this problem use the Lena image shown in Figure 2.23 which is available in the
book toolbox.

    (a) Load the Lena image in MATLAB and display using the `imshow` function.

    (b) Consider the 3×3 impulse response $h[m, n]$ given in (2.75). Filter the Lena image
using (2.78) and display the resulting image and verify that it looks similar to the
corresponding one in Figure 2.23. Assume zero boundary conditions.

    (c) Repeat part (b) using the impulse response

$$h[m, n] = \begin{cases} \frac{1}{25}, & -2 \le m, n \le 2 \\ 0, & \text{otherwise} \end{cases}$$

and comment on the result.

16. In this problem use the Lena image shown in Figure 2.23 which is available in the
book toolbox.

    (a) Load the Lena image in MATLAB and display using the `imshow` function.

    (b) Consider the 1D impulse response $h[n] = \frac{1}{5}\{1, 1, 1, 1, 1\}$. Using it perform 1D
convolution along each row of the Lena image and display the resulting blurred
image. Comment on the result.

    (c) Using the above impulse response now perform comvolution along each column
of the Lena image and display the resulting blurred image. Compare this image
with the one in part (b).

    (d) Using $h[n]$ perform convolution along each column of the result image of part (b)
and display the resulting image. How does it compare with the above two result
images as well as the one in part (c) in Problem 15?

17. A discrete-time system is described by the following difference equation

$$y[n] = 1.15y[n-1] - 1.5y[n-2] + 0.7y[n-3] - 0.25y[n-4] + 0.18x[n]$$
$$+ 0.1x[n-1] + 0.3x[n-2] + 0.1x[n-3] + 0.18x[n-4]$$

with zero initial conditions.

(a) Compute and plot the impulse response $h[n]$, $0 \le n \le 100$ using the function `h=impz(b,a,N)`.

(b) Compute and plot the output $y[n]$, if $x[n] = u[n]$, $0 \le n \le 100$ using the function `y=filter(b,a,x)`.

(c) Compute and plot the output $y[n]$, if $x[n] = u[n]$, $0 \le n \le 100$ using the function `y=conv(h,x)`.

(d) Compute and plot the output $y[n]$, if $x[n] = u[n]$, $0 \le n \le 100$ using the function `y=filter(h,1,x)`.

Compare and explain the obtained results.

18. Consider the nonrecursive (2.96) and recursive (2.98) implementations of the moving average filter discussed in Example 2.7.

(a) Draw block diagram representations of the nonrecursive and recursive representations for $M = 5$.

(b) Compute the step response of the system for $M = 5$ using MATLAB function `filter` to implement (i) the nonrecursive implementation and (ii) the recursive implementation.

19. A recursive implementation of reverberation is given by (2.103) which is given below

$$y[n] = x[n] + ay[n-D],$$

where $D = \tau F_s$ is the delay in sampling interval given the delay τ in seconds and sampling rate F_s and a is an attenuation factor. To generate digital reverberation we will use the sound file `handel` which is recorded at $F_s = 8192$ samples per second. (See Problem 6 for using this file.)

(a) For $\tau = 50$ ms and $a = 0.7$, obtain a difference equation for the digital reverberation and process the sound in `handel`. Comment on its audio quality.

(b) Repeat (a) for $\tau = 100$ ms.

(c) Repeat (a) for $\tau = 500$ ms.

(d) Which implementation sounds natural?

20. A continuous-time LTI system has impulse response $h(t) = e^{-t/2}u(t)$.

(a) Determine the responses $y_1(t)$ and $y_2(t)$ to the input signals $x_1(t) = u(t)$ and $x_2(t) = 2, 0 \le t \le 3$ and zero elsewhere.

(b) Using the properties of linearity and time-invariance, show that $y_2(t)$ can be obtained from $y_1(t)$.

Basic problems

21. Run and carefully study MATLAB script `dtsas.m` to familiarize yourself with the generation, plotting, and manipulation of discrete-time signals.

22. A downsampler system is defined in (2.24). Consider the sequence $x[n] = \cos(0.1\pi n)$ for $-30 \leq n \leq 30$. Using the `stem` function plot
    (a) $x[n]$ versus n.
    (b) A down sampled signal $y[n]$ for $M = 5$.
    (c) A down sampled signal $y[n]$ for $M = 20$.
    (d) How does the downsampled signal appear? Compressed or expanded.

23. Test which of the following systems are linear, time-invariant, causal, and stable.
    (a) $y[n] = x[-n]$ (Time-flip)
    (b) $y[n] = \log(|x[n]|)$ (Log-magnitude)
    (c) $y[n] = x[n] - x[n - 1]$ (First-difference)
    (d) $y[n] = \text{round}\{x[n]\}$ (Quantizer)

24. The file `djw6576.txt` contains the weekly opening value $x[n]$, $0 \leq n \leq N - 1$, of the Dow Jones Industrial Average for $N = 600$ weeks starting in January 1965.
    (a) Write a MATLAB script to compute the following moving averages

    $$y_1[n] = \frac{1}{51} \sum_{k=0}^{50} x[n - k] \quad \text{and} \quad y_2[n] = \frac{1}{51} \sum_{k=-25}^{25} x[n - k].$$

    Use the MATLAB functions `filter` or `conv` only to check your results.
    (b) Plot the sequences $x[n]$, $y_1[n]$, and $y_2[n]$ for $0 \leq n \leq N - 1$ on the same plot and comment upon the results. Use function `plot` and represent each sequence with a dot of different color.

25. Consider the finite duration sequences $x[n] = u[n] - u[n - N]$ and $h[n] = n(u[n] - u[n - M])$, $M \leq N$.
    (a) Find an analytical expression for the sequence $y[n] = h[n] * x[n]$.
    (b) Verify the result in (a) for $N = 10$ and $M = 5$ using function `y=conv(h,x)`.

26. Repeat Example 2.6 assuming that $M > N$. Hint: See Figure 2.16.

27. Determine the convolution $y[n] = h[n] * x[n]$ of the following sequences

    $$x[n] = a^n u[n], \quad h[n] = b^n u[n]$$

    for $a \neq b$ and verify the result with MATLAB using $a = 1/4$ and $b = 1/3$.

28. Let $x[n] = h[n] = (0.9)^n u[n]$ and $y[n] = x[n] * h[n]$.
    (a) Determine $y[n]$ analytically and plot using MATLAB.
    (b) Take first 50 samples of $x[n]$ and $h[n]$. Compute and plot $y[n]$ using the `conv` function.
    (c) Using the filter function, determine and plot the first 99 samples of $y[n]$.
    (d) Which of the outputs in (b) and (c) come close to that in (a)? Explain.

29. The properties of convolution are given in Table 2.3. Verify these properties using MATLAB on the following signals:

    $$x[n] = n\{u[n - 10] - u[n + 15]\}$$
    $$h[n] = (0.5)^n \{u[n] - u[n - 10]\}$$
    $$h_1[n] = \cos(0.05\pi n)\{u[n] - u[n - 21]\}$$
    $$h_2[n] = 2\delta[n + 3] + \delta[n - 1] - 3\delta[n - 5].$$

30. Write a MATLAB function

    `[y,L1,L2]=convol(h,M1,M2,x,N1,N2)`

    to compute the convolution of two arbitrarily positioned finite length sequences using the procedure illustrated in Figure 2.16. Also, show how to simplify the implementation using a sequence $x_{zp}[n]$ created by padding the beginning and the end of $x[n]$ with $M_2 - M_1$ zeros.

31. Consider the system $y[n] = y[n-1] + y[n-2] + x[n]$, $y[-1] = y[-2] = 0$.
    (a) Compute and plot the impulse response, for $0 \leq n \leq 100$, using function `filter`.
    (b) Can you draw any conclusions about the stability of this system from the results in (a)?
    (c) Determine the output $y[n]$, if the input is $x[n] = a^n$, $-\infty < n < \infty$, and comment upon the result.

32. Use the function `filter` to compute and plot the first 60 samples of the impulse response and step response of the system

    $$y[n] = 1.15y[n-1] - 1.5y[n-2] + 0.7y[n-3] - 0.25y[n-4] + 0.18x[n]$$
    $$+ 0.1x[n-1] + 0.3x[n-2] + 0.1x[n-3] + 0.18x[n-4].$$

33. A first-order digital differentiator is given by $y[n] = x[n] - x[n-1]$. Implement this filter on the following signals and plot the results.
    (a) $x[n] = 10\{u[n+10] - u[n-20]\}$.
    (b) $x[n] = n\{u[n] - u[n-10]\} + (20-n)\{u[n-10] - u[n-20]\}$.
    (c) $x[n] = \cos(0.2\pi n - \pi/2)\{u[n] - u[n-40]\}$.

34. A system is described by the difference equation

    $$y[n] = x[n] - 0.9y[n-1] + 0.81y[n-2]. \tag{2.121}$$

    Using MATLAB determine and plot
    (a) Impulse response of the system.
    (b) Step response of the system.
    (c) Identify the transient response and the steady-state response in (b).

35. Check whether the following systems are linear, time-invariant, causal, and stable.
    (a) $y(t) = x(t-1) + x(2-t)$
    (b) $y(t) = dx(t)/dt$
    (c) $y(t) = \int_{-\infty}^{3t} x(\tau)d\tau$
    (d) $y(t) = 2x(t) + 5$.

36. Consider two continuous-time signals defined by

    $$h(t) = \begin{cases} 1, & -1 \leq t \leq 1 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad x(t) = \begin{cases} (1/3)t, & 0 \leq t \leq 3 \\ 0. & \text{otherwise} \end{cases}$$

    (a) Determine the convolution function $y(t) = h(t) * x(t)$.
    (b) If $h[n] = h(nT)$ and $x[n] = x(nT)$, show that $y(t)$ can be approximated at any value of t by

    $$\hat{y}(t) = T \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

(c) Evaluate $\hat{y}(nT)$ and compare it with $y(nT)$ by plotting both sequences on the same graph for $T = 0.1$ and $T = 0.01$. Also compare the mean square error $E = (1/N)\sum_{n=0}^{N-1}(y[n] - \hat{y}[n])^2$ in the two cases and determine for what value of T this error becomes negligible.

Assessment problems

37. Let $x[n] = \{\underset{\uparrow}{0}, 1, 2, 3, 4, 5\}$. Consider a new sequence $x[-4 - n] = x[-(n+4)]$.

   (a) Let $y_1[n]$ be obtained by first folding $x[n]$ and then shifting the result to the left by four samples. Determine and plot $y_1[n]$.
   (b) Let $y_2[n]$ be obtained by first shifting $x[n]$ to the left by four samples and then folding the result. Determine and plot $y_2[n]$.
   (c) From your plots are $y_1[n]$ and $y_2[n]$ the same signals? Which signal represents the correct $x[-4 - n]$ signal?

38. Generate and plot (using the `stem` function) samples of the following signals:
   (a) $x_1[n] = 5\delta[n+1] + n^2(u[n+5] - u[n-4]) + 10(0.5)^n(u[n-4] - u[n-8])$.
   (b) $x_2[n] = (0.8)^n \cos(0.2\pi n + \pi/4), 0 \le n \le 20$.
   (c) $x_3[n] = \sum_{m=0}^{4}(m+1)\{\delta[n-m] - \delta[n-2m]\}, 0 \le n \le 20$.

39. Let the sequence $x[n]$ be

$$x[n] = \begin{cases} 0, & n < 0 \\ n, & 0 \le n \le 10 \\ 0, & 11 \le n. \end{cases} \qquad (2.122)$$

   Generate and plot using the `stem` function,
   (a) $2x[n-4]$.
   (b) $3x[n-5]$.
   (c) $x[3-n]$.

40. Consider the following discrete-time system

$$y[n] = 10x[n]\cos(0.25\pi n + \theta), \qquad (2.123)$$

   where θ is a constant. Check if the system is
   (a) Linear.
   (b) Time invariant.
   (c) Causal.
   (d) Stable.

41. Write a MATLAB function to compute and plot the output of the discrete-time system

$$y[n] = 5y[n-1] + x[n], \quad y[-1] = 0$$

   for $x[n] = u[n], 0 \le n \le 1000$. Based on these results can you make a statement regarding the stability of the system? Hint: Check the value $y[600]$.

42. A discrete-time LTI system with input $x[n]$ and output $y[n]$ is implemented by the MATLAB function `y=agnosto(x)`. Using this function *only once*, compute the response $y_i[n]$ of the system to the input sequences $x_1[n] = u[n]$, $x_2[n] = (1/2)^n$, and $x_3[n] = \cos(2\pi n/20)$, for $0 \le n \le 100$. Hint: After completing the problem, you may use `y=agnosto(x)` to check your results.

43. The sum $A_x \triangleq \sum_n x[n]$ can be thought of as a measure of the "area" under a sequence $x[n]$.

    (a) Starting with the convolution sum (2.36), show that $A_y = A_x A_h$.

    (b) Given the sequences

    ```
    x=sin(2*pi*0.01*(0:100))+···0.05*randn(1,101);   h=ones(1,5);
    ```

    compute $y[n] = h[n] * x[n]$, check whether $A_y = A_x A_h$, and plot $x[n]$ and $y[n]$ on the same graph.

    (c) Normalize $h[n]$ so that $A_h = 1$ and repeat part (b).

    (d) If $A_h = 1$, then $A_y = A_x$. Use this result to explain the difference between the plots obtained in parts (b) and (c).

44. Compute the step response of a system with impulse response $h[n] = ba^n u[n]$. Choose b so that $s[n]$ approaches the level of $u[n]$ for large values of n. Hint: Use the results in Problem 43.

45. Repeat the procedure described in Example 2.6, by folding the sequence $x[k]$ instead of $h[k]$. Based on (2.44) you should obtain the same output sequence.

46. In this problem use the Lena image available in the book toolbox.

    (a) Load the Lena image in MATLAB and display using the `imshow` function.

    (b) Consider the 3×3 impulse response $h[m, n]$ (known as a Sobel filter) given below

    $$h[m, n] = \begin{array}{|ccc|} \hline 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \\ \hline \end{array}$$

    in which the $[0, 0]$ term is in the center. Filter the Lena image using this impulse response and display the resulting image. Comment on the result. (Study Tutorial Problem 15.)

    (c) Repeat part (b) using the impulse response

    $$h[m, n] = \begin{array}{|ccc|} \hline 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \\ \hline \end{array}$$

    and comment on the result.

47. Based on the MATLAB script in Figure 2.22 write a script `lccde.m` to compute the linear constant-coefficient difference equation (2.94). Test the code by computing the impulse response of the system given in Problem 32.

48. Compute the convolution $y(t) = h(t) * x(t)$ for $h(t) = u(t) - u(t - 3)$ and $x(t) = u(t) - u(t - 2)$.

49. Repeat Problem 36 using the signals $h(t)$ and $x(t) = x_2(t)$ in Problem 20.

Review problems

50. Consider the following reverberator: $y[n] = ay[n - D] + x[n - D]$
    (a) Explain the key difference between this reverberator and the one in (2.103).
    (b) Write a MATLAB function for the implementation of this reverberator that requires D delays only.
    (c) Compute and plot the impulse response for $a = 0.7$ and compare with the impulse response of (2.103).
    (d) Experiment as in Problem 19 and compare the two reverberators.

51. The digital echo system described in Example 2.8 can be represented by a general impulse response

$$h[n] = \sum_{k=0}^{\infty} a_k \delta[n - kD].$$

To remove these echoes, an inverse system is needed and one implementation of such a system is given by

$$g[n] = \sum_{k=0}^{\infty} b_k \delta[n - kD],$$

such that $h[n] * g[n] = \delta[n]$.
    (a) Determine the algebraic equations that the successive b_k must satisfy.
    (b) Solve these equations for b_0, b_1, and b_2 in terms of a_k.
    (c) For $a_0 = 1$, $a_1 = 0.5$, $a_2 = 0.25$, and all other a_ks zero, determine $g[n]$.

52. Determine, with justification, whether each of the following statements is true or false regarding discrete-time LTI systems.
    (a) A system is causal if the step response $s[n]$ is zero for $n < 0$.
    (b) If the impulse response $h[n] \neq 0$ is periodic, then the output is always periodic.
    (c) A cascade connection of a stable and an unstable system is always unstable.
    (d) The inverse of a causal system is a noncausal system.
    (e) A system with infinite-duration impulse response is unstable.
    (f) If $|h[n]|$ is finite at each n, then the system is stable.

53. The second derivative operation $y = \frac{dx}{dt}$ is approximated by the difference equation

$$y[n] = x[n + 1] - 2x[n] + x[n - 1], \tag{2.124}$$

which is a noncausal LTI system and is used as an edge detector in image processing.
    (a) Determine the impulse response of this edge detector.
    (b) Load the Lena image in MATLAB and process it row-by-row using the above impulse response. Display the resulting image and comment on its appearance.
    (c) Now process the Lena image column-by-column using the impulse response in (a). Display the resulting image and comment on its appearance.

54. The 2D second derivative or *Laplacian* can be approximated by the noncausal impulse response

$$h[m, n] = \begin{vmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{vmatrix} \qquad (2.125)$$

in which the $[0, 0]$ term is in the center. It is also used as an edge detector in image processing.

(a) Load the Lena image in MATLAB and process it using the impulse response (2.125). Display the resulting image and comment on its appearance.

(b) An edge-enhanced image is obtained by subtracting the Laplacian of the image from the original. Determine the impulse response of this operation.

(c) Now process the Lena image using the impulse response in (b). Display the resulting image and comment on its appearance.

3

The *z*-transform

We have seen that discrete-time signals or sequences are defined, generated, and processed by systems in the *n-domain* or *time-domain*, that is, as functions of the discrete index *n*. In this sense, we also say that the implementation of discrete-time systems takes places in the time-domain. The purpose of this chapter is threefold. First, we introduce a new representation of sequences, known as the *z*-transform. Second, we study how the properties of a sequence are related to the properties of its *z*-transform. Finally, we use the *z*-transform to study LTI systems described by a convolution sum or a linear constant-coefficient difference equation.

Study objectives

After studying this chapter you should be able to:

- Understand how to represent a sequence of numbers with a function of a complex variable called the *z*-transform.
- Change a sequence by manipulating its *z*-transform and vice versa.
- Possess a basic understanding of the concept of system function and use it to investigate the properties of discrete-time LTI systems.
- Determine the output of systems described by linear constant-coefficient difference equations using the *z*-transform.

3.1 Motivation

In Section 2.4, we exploited the decomposition of an arbitrary sequence into a linear combination of scaled and shifted impulses,

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k], \tag{3.1}$$

to show that every LTI system can be represented by the convolution sum

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n - k] = \sum_{k=-\infty}^{\infty} h[k]x[n - k]. \tag{3.2}$$

The impulse response sequence $h[n]$ specifies completely the behavior and the properties of the associated LTI system.

In general, any sequence that passes through a LTI system changes shape. We now ask: is there any sequence that retains its shape when it passes through an LTI system? To answer this question, we consider the complex exponential sequence

$$x[n] = z^n, \quad \text{for all } n \tag{3.3}$$

where $z = \mathcal{R}e(z) + j\mathcal{I}m(z)$ is a complex variable defined everywhere on the complex plane. We emphasize that the condition "for all n" in (3.3), is extremely important for the validity of subsequent results. The response of the LTI system (3.2) to the input sequence (3.3) is

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]z^{n-k} = \left(\sum_{k=-\infty}^{\infty} h[k]z^{-k} \right) z^n, \quad \text{for all } n. \tag{3.4}$$

If the summation inside the parentheses converges, the result is a function of z, denoted by

$$H(z) = \sum_{k=-\infty}^{\infty} h[k]z^{-k}. \tag{3.5}$$

Then the output sequence is given by

$$y[n] = H(z)z^n, \quad \text{for all } n. \tag{3.6}$$

Thus, the output sequence is the same complex exponential as the input sequence, multiplied by a constant $H(z)$ that depends on the value of z. The quantity $H(z)$, as a function of the complex variable z, is known as the *system function* or *transfer function* of the system.

We say that the complex exponential sequences (3.3) are eigenfunctions of LTI systems. The constant $H(z)$, for a specified value of the complex variable z, is the eigenvalue associated with the eigenfunction z^n. We note that, in contrast to impulse sequences,

whose shape changes when they pass through LTI systems, complex exponential sequences *retain* their shape. This property, which is an exclusivity of LTI systems and complex exponential sequences, provides the basis for the analysis of LTI systems using the z-transform.

If the input to a LTI system can be expressed as a linear combination of complex exponentials, that is,

$$x[n] = \sum_k c_k z_k^n, \quad \text{for all } n \tag{3.7}$$

then, due to the linearity property, the output will be

$$y[n] = \sum_k c_k H(z_k) z_k^n, \quad \text{for all } n. \tag{3.8}$$

If $H(z_k) = 0$, for some z_k, the corresponding complex exponential sequence does *not* pass through the system. This observation provides the basis for the design of systems that selectively "filter out" certain complex exponential components of the input signal. However, for this to be useful, we should be able to express any sequence as a linear combination of complex exponentials. This decomposition is made possible using the *z*-transform of discrete-time signals.

The *z*-transform is a powerful tool, that can be used to understand, analyze, and design LTI systems and provide insight into their effect on the input signals.

3.2   The *z*-transform

The *z*-transform of a sequence $x[n]$ is a function $X(z)$ defined by

$$X(z) = \sum_{n=-\infty}^{\infty} x[n] z^{-n}, \tag{3.9}$$

where the independent variable z can represent any complex number. Since z is a complex variable, it is convenient to interpret the *z*-transform using the correspondence between complex numbers and points in the plane. This correspondence is illustrated in Figure 3.1(a). The unit circle, shown in Figure 3.1(b), is defined by $|z| = 1$ and shows the geometric loci of all points at distance one from the origin.

The infinite sum in (3.9) may or may not be finite for all sequences or all values of z. For any given sequence, the set of values of z for which the series (3.9) converges is known as the *region of convergence* (ROC) of the z-transform. The values of z for which $X(z) = 0$ are called *zeros* of $X(z)$, and the values of z for which $X(z)$ is infinite are known as *poles*. By definition, the ROC *cannot* include any poles. A graphical illustration of *z*-transform is provided in Figure 3.2. As we illustrate in the following examples, the ROC is an essential part of the *z*-transform.
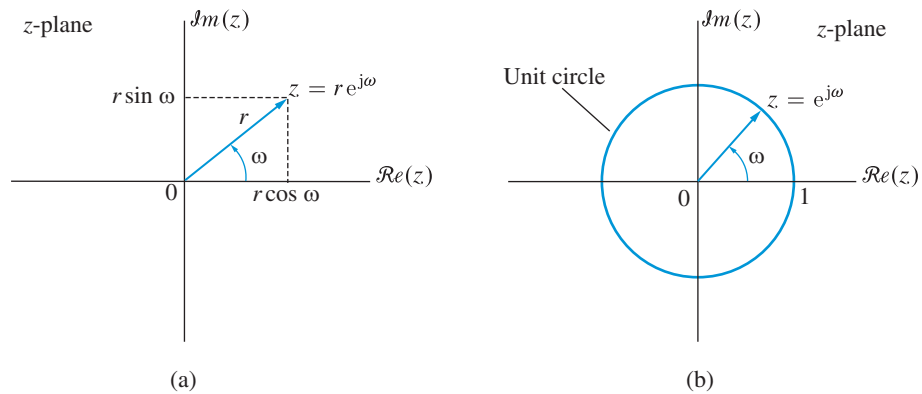
Figure 3.1 (a) A point $z = re^{j\omega}$ in the complex plane can be specified by the distance r from the origin and the angle ω with the positive real axis (polar coordinates) or the rectangular coordinates $r\cos(\omega)$ and $r\sin(\omega)$. (b) The unit circle, $|z| = 1$, in the complex plane.
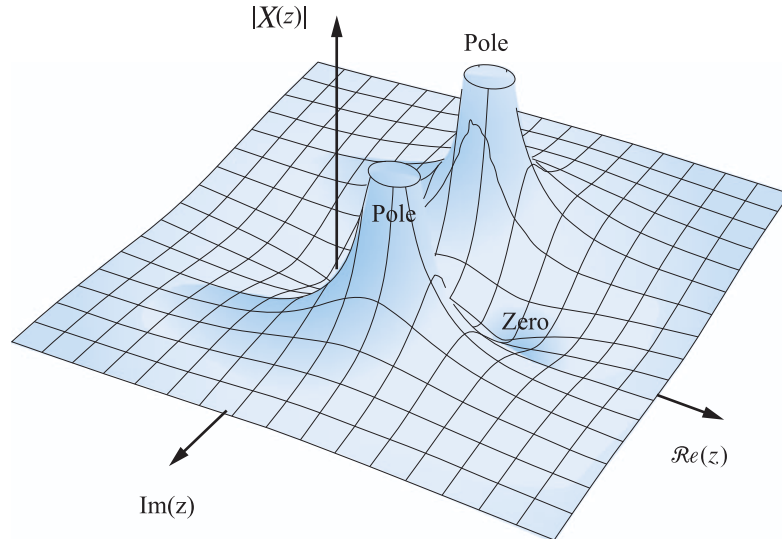
Figure 3.2 The magnitude $|X(z)|$ of the *z*-transform represents a surface in the *z*-plane. There are two zeros at $z_1 = 0$, $z_2 = 1$ and two poles at $p_{1,2} = 0.9e^{\pm j\pi/4}$.

Example 3.1   Unit sample sequence

The *z*-transform of the unit sample sequence is given by

$$X(z) = \sum_{n=-\infty}^{\infty} \delta[n]z^{-n} = z^0 = 1. \quad \text{ROC: All } z \tag{3.10}$$

Example 3.2   Square-pulse sequence

The *z*-transform of the square-pulse sequence

$$x[n] = \begin{cases} 1, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

is given by

$$X(z) = \sum_{n=0}^{M} 1 z^{-n} = \frac{1 - z^{-(M+1)}}{1 - z^{-1}}. \quad \text{ROC: } z \ne 0 \tag{3.12}$$

∎

Example 3.3   Exponential-pulse sequence

The *z*-transform of the exponential-pulse sequence

$$x[n] = \begin{cases} a^n, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases} \tag{3.13}$$

is given by

$$X(z) = \sum_{n=0}^{M} a^n z^{-n} = \sum_{n=0}^{M} (az^{-1})^n = \frac{1 - a^{M+1} z^{-(M+1)}}{1 - az^{-1}}. \quad \text{ROC: } z \ne 0 \tag{3.14}$$

∎

Example 3.4   Causal exponential sequence

The *z*-transform of the causal exponential sequence $x[n] = a^n u[n]$ is given by

$$X(z) = \sum_{n=0}^{\infty} (az^{-1})^n = \frac{1}{1 - az^{-1}} = \frac{z}{z - a}. \quad \text{ROC: } |z| > |a| \tag{3.15}$$

The infinite geometric series converges if $|az^{-1}| < 1$ or $|z| > |a|$. Since $X(z) = 1/(1 - az^{-1}) = z/(z - a)$, there is a zero at $z = 0$ and a pole at $p = a$. For $a = 1$ we obtain the *z*-transform of the unit step sequence

$$X(z) = \frac{1}{1 - z^{-1}}. \quad \text{ROC: } |z| > 1 \tag{3.16}$$
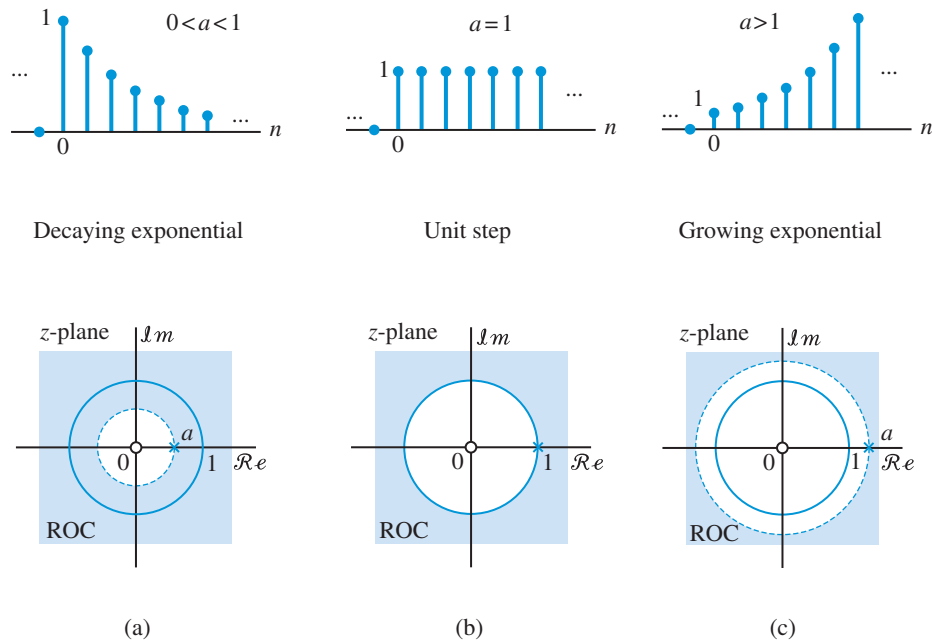
Figure 3.3 Pole-zero plot and region of convergence of a causal exponential sequence $x[n] = a^n u[n]$ with (a) decaying amplitude ($0 < a < 1$), (b) fixed amplitude (unit step sequence), and (c) growing amplitude ($a > 1$).

Figure 3.3 shows the sequence $x[n] = a^n u[n]$ and the ROC of its z-transform for $a < 1$, $a = 1$, and $a > 1$. Note that a can be real or complex. ∎

Example 3.5   Anticausal exponential sequence

The z-transform of the anticausal exponential sequence

$$y[n] = -b^n u[-n-1] = \begin{cases} 0, & n \geq 0 \\ -b^n, & n < 0 \end{cases} \tag{3.17}$$

is given by

$$Y(z) = -\sum_{n=-\infty}^{-1} b^n z^{-n} = -b^{-1}z(1 + b^{-1}z + b^{-2}z^2 + \cdots).$$

The infinite geometric series inside the parenthesis converges if $|b^{-1}z| < 1$ or $|z| < |b|$. Thus

$$Y(z) = \frac{-bz^{-1}}{1 - b^{-1}z} = \frac{1}{1 - bz^{-1}} = \frac{z}{z - b}. \quad \text{ROC: } |z| < |b| \tag{3.18}$$

The z-transform function $Y(z)$ has a zero at $z = 0$ and a pole at $p = b$. ∎

For $b = a$ we have $Y(z) = X(z)$, even if $x[n] \neq y[n]$! Hence, the unique specification of a sequence $x[n]$ requires both the function $X(z)$ and its ROC.

Example 3.6   Two-sided exponential sequence

The *z*-transform of the two-sided exponential sequence

$$x[n] = \begin{cases} a^n, & n \geq 0 \\ -b^n, & n < 0 \end{cases} \tag{3.19}$$

is obtained by substituting (3.19) into (3.9) and by splitting the summation into two parts as follows:

$$X(z) = -\sum_{n=-\infty}^{-1} b^n z^{-n} + \sum_{n=0}^{\infty} a^n z^{-n}. \tag{3.20}$$

The first sum, see (3.15), converges to $1/(1 - bz^{-1})$ for $|z| < |b|$. The second sum, see (3.18), converges to $1/(1 - az^{-1})$ for $|z| > |a|$. For $X(z)$ to exist, both sums must converge for a set of common values of z. This requires that both $|z| < |b|$ and $|z| > |a|$. The two sets overlap only when $|b| > |a|$, in which case the ROC is the annular region $|a| < |z| < |b|$ (see Figure 3.4). The *z*-transform does *not* exist when $|b| < |a|$. ∎

Example 3.7   Exponentially oscillating sequence

Consider a causal sinusoidal sequence with exponentially varying amplitude:

$$x[n] = r^n(\cos \omega_0 n)u[n]. \quad r > 0, \ 0 \leq \omega_0 < 2\pi \tag{3.21}$$

The constant ω_0 determines the number of samples per period of oscillation in the sequence $\cos \omega_0 n$. Periodicity requires that $\cos \omega_0 n = \cos[\omega_0(n+N)]$ for all n or equivalently $\omega_0 N = k2\pi$. A period of 2π radians contains N samples, where $N = 2\pi/\omega_0$. For example, if $\omega_0 = \pi/4$ radians, the sinusoid is periodic with fundamental period $N = 8$ samples.

Using the identity $\cos \theta = \frac{1}{2}e^{j\theta} + \frac{1}{2}e^{-j\theta}$, we have

$$X(z) = \sum_{n=0}^{\infty} r^n(\cos \omega_0 n)z^{-n} = \frac{1}{2}\sum_{n=0}^{\infty}(re^{j\omega_0}z^{-1})^n + \frac{1}{2}\sum_{n=0}^{\infty}(re^{-j\omega_0}z^{-1})^n. \tag{3.22}$$

Since $|e^{\pm j\omega_0}| = 1$, both sums converge if $|rz^{-1}| < 1$, or, equivalently, $|z| > r$. Hence,

$$X(z) = \frac{1}{2}\frac{1}{1 - re^{j\omega_0}z^{-1}} + \frac{1}{2}\frac{1}{1 - re^{-j\omega_0}z^{-1}}, \quad \text{ROC: } |z| > r \tag{3.23}$$
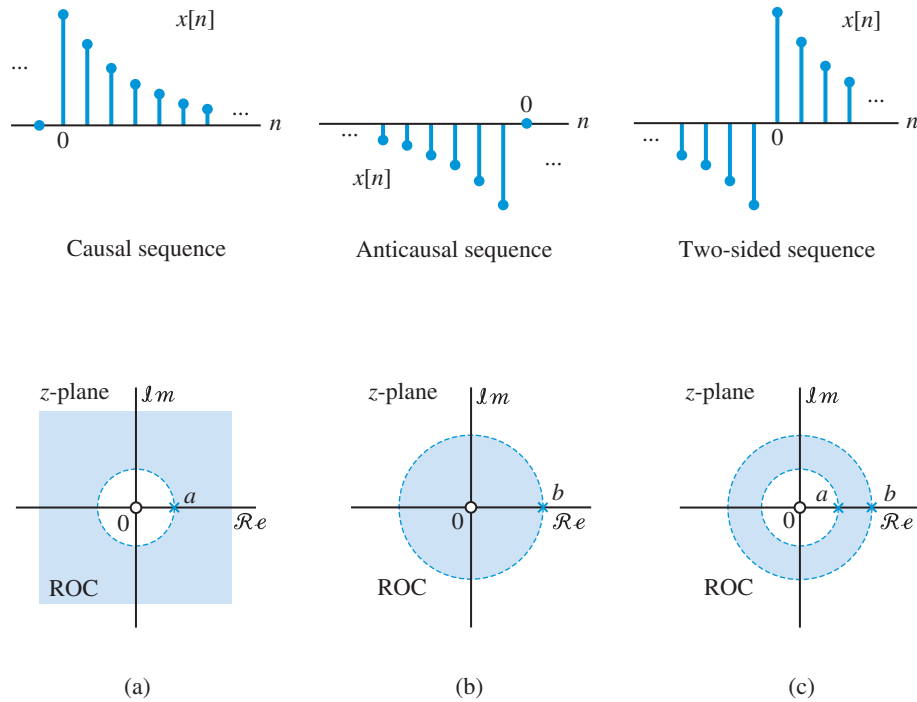
Figure 3.4 Pole-zero plot and region of convergence for the (a) causal, (b) anticausal, and (c) two-sided exponential sequences discussed in Example 3.6.

or by combining the two terms

$$X(z) = \frac{1 - (r\cos\omega_0)z^{-1}}{(1 - re^{j\omega_0}z^{-1})(1 - re^{-j\omega_0}z^{-1})} = \frac{1 - (r\cos\omega_0)z^{-1}}{1 - 2(r\cos\omega_0)z^{-1} + r^2 z^{-2}}. \tag{3.24}$$

Multiplying both numerator and denominator by z^2, we have

$$X(z) = \frac{z(z - r\cos\omega_0)}{(z - re^{j\omega_0})(z - re^{-j\omega_0})}. \tag{3.25}$$

Thus, $X(z)$ has two zeros at $z_1 = 0, z_2 = r\cos\omega_0$ and two complex-conjugate poles at $p_1 = re^{j\omega_0}, p_2 = re^{-j\omega_0}$. The pole-zero plot and ROC are shown in Figure 3.5. ■

For easy reference, a summary of z-transform pairs is provided in Table 3.1. These transform pairs will be sufficient to deal with the z-transforms of most sequences encountered in practice. From the previous examples and Table 3.1 we see that the ROC of any z-transform has the following properties:

3.2 The z-transform

z-plane, $\mathcal{I}m$, p_1, z_1, z_2, 0, $\mathcal{R}e$, p_2, ROC
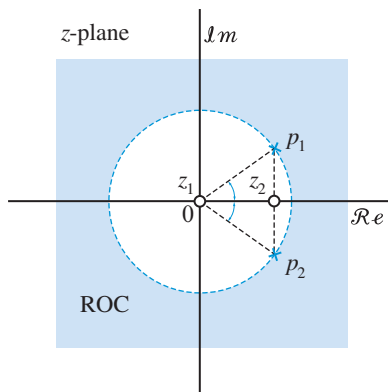
Figure 3.5 Pole-zero plot and region of convergence for Example 3.7.

Table 3.1 Some common z-transform pairs

	Sequence $x[n]$	z-Transform $X(z)$	ROC
1.	$\delta[n]$	1	All z
2.	$u[n]$	$\dfrac{1}{1 - z^{-1}}$	$\lvert z \rvert > 1$
3.	$a^n u[n]$	$\dfrac{1}{1 - az^{-1}}$	$\lvert z \rvert > \lvert a \rvert$
4.	$-a^n u[-n-1]$	$\dfrac{1}{1 - az^{-1}}$	$\lvert z \rvert < \lvert a \rvert$
5.	$na^n u[n]$	$\dfrac{az^{-1}}{(1 - az^{-1})^2}$	$\lvert z \rvert > \lvert a \rvert$
6.	$-na^n u[-n-1]$	$\dfrac{az^{-1}}{(1 - az^{-1})^2}$	$\lvert z \rvert < \lvert a \rvert$
7.	$(\cos \omega_0 n)u[n]$	$\dfrac{1 - (\cos \omega_0)z^{-1}}{1 - 2(\cos \omega_0)z^{-1} + z^{-2}}$	$\lvert z \rvert > 1$
8.	$(\sin \omega_0 n)u[n]$	$\dfrac{(\sin \omega_0)z^{-1}}{1 - 2(\cos \omega_0)z^{-1} + z^{-2}}$	$\lvert z \rvert > 1$
9.	$(r^n \cos \omega_0 n)u[n]$	$\dfrac{1 - (r\cos \omega_0)z^{-1}}{1 - 2(r\cos \omega_0)z^{-1} + r^2 z^{-2}}$	$\lvert z \rvert > r$
10.	$(r^n \sin \omega_0 n)u[n]$	$\dfrac{(\sin \omega_0)z^{-1}}{1 - 2(r\cos \omega_0)z^{-1} + r^2 z^{-2}}$	$\lvert z \rvert > r$

- The ROC *cannot* include any poles.
- The ROC is a connected (that is, a single contiguous) region.
- For finite duration sequences the ROC is the entire z-plane, with the possible exception of $z = 0$ or $z = \infty$.

- For infinite duration sequences the ROC can have one of the following shapes:

Type of sequence	ROC
Right-sided ($x[n] = 0, n < n_0$) $\Rightarrow$	ROC: $|z| > r$
Left-sided ($x[n] = 0, n > n_0$) $\Rightarrow$	ROC: $|z| < r$
Two-sided $\Rightarrow$	ROC: $a < |z| < b$

Finally, it is important to emphasize the following important points:

- The z-transform of a sequence consists of an algebraic formula and its associated ROC. Thus, to uniquely specify a sequence $x[n]$ we need both $X(z)$ and its ROC.
- The function $X(z)$ is legitimate only for z within its ROC. We stress that $X(z)$ is *not* defined when z is outside the ROC, even if the formula for $X(z)$ yields meaningful results for these values.

Representation of polynomials in MATLAB   Since most practical z-transforms are a ratio of polynomials, we start by explaining how MATLAB handles polynomials. In MATLAB polynomials are represented by *row* vectors containing the coefficients of the polynomial in decreasing order. For example, the polynomial

$$B(z) = 1 + 2z^{-1} + 3z^{-3}$$

is entered as b=[1,2,0,3]. We stress that even though the coefficient of the z^{-2} term is zero, it is included in the coefficient vector. The function z=roots(b) computes and returns the roots of a polynomial as a column vector. If z is a column vector containing the roots of a polynomial, the function b=poly(z) returns a row vector with the polynomial coefficients. The use of these functions is illustrated in the following script:

```
>> b=[1,1.5,2]; z=roots(b)
z =
  -0.7500 + 1.1990i
  -0.7500 - 1.1990i
>> b=poly(z)
b =
    1.0000    1.5000    2.0000
```

Some extra caution is required if we wish to compute the value of $B(z)$ at a given value of z. The reason is that MATLAB assumes polynomials with positive exponents, that is,

$$B(z) = 1 + 2z^{-1} + 3z^{-3} = z^{-3}(z^3 + 2z^2 + 3) = z^{-3}\tilde{B}(z); \tag{3.26}$$

to evaluate the value of $B(z)$ at $z = 2$ we use the command

```
>> polyval(b,2)/2^3
ans =
    2.3750.
```

Additional functions for polynomial manipulation will be introduced as needed.

3.3 The inverse z-transform

The recovery of a sequence $x[n]$ from its z-transform ($X(z)$ and ROC) can be formally done using the formula

$$x[n] = \frac{1}{2\pi j} \oint_C X(z)z^{n-1}\mathrm{d}z, \tag{3.27}$$

which involves complex integration using the method of residues. However, the following simpler procedures are sufficient for most sequences and z-transforms encountered in the analysis of LTI systems:

- Expansion into a series of terms in the variables z and z^{-1} and picking their coefficients. For rational functions this is done using long division. This is implemented in MATLAB using the function `deconv` and is mostly used to compute a few values for checking purposes.
- Partial fraction expansion and table look-up, which is implemented in MATLAB using the function `residuez`. This is the method used in most practical applications.

Using the definition (3.9) we can show that the z-transform of a linear combination of distinct exponentials (that is, $p_k \neq p_m, k \neq m$) is given by

$$x[n] = \sum_{k=1}^{N} A_k(p_k)^n \overset{\mathcal{Z}}{\longleftrightarrow} X(z) = \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}, \tag{3.28}$$

with ROC the intersection of the ROCs of the individual exponential sequences. If we combine the terms of the summation, we have

$$X(z) = \frac{\displaystyle\sum_{k=1}^{N} A_k \prod_{\substack{m=1 \\ m \neq k}}^{N} (1 - p_m z^{-1})}{\displaystyle\prod_{k=1}^{N}(1 - p_k z^{-1})} = \frac{b_0 + b_1 z^{-1} + \cdots + b_{N-1} z^{-(N-1)}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \tag{3.29}$$

which is a *proper* rational function because the degree of the numerator is less than the degree of the denominator. This suggests a procedure for the inversion of proper rational z-transforms with distinct poles, which is illustrated in the following two examples.

Example 3.8   Real and distinct poles

Consider a sequence $x[n]$ with z-transform

$$X(z) = \frac{1 + z^{-1}}{(1 - z^{-1})(1 - 0.5z^{-1})}. \tag{3.30}$$
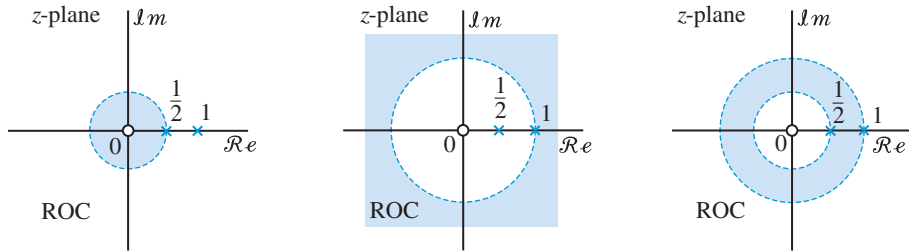
Figure 3.6 Pole-zero plot and possible regions of convergence for the z-transform in Example 3.8.

Since this is a proper rational fraction with distinct poles $p_1 = 1$ and $p_2 = 0.5$, it can be expressed in the form (3.28) as

$$X(z) = \frac{1 + z^{-1}}{(1 - z^{-1})(1 - 0.5z^{-1})} = \frac{A_1}{1 - z^{-1}} + \frac{A_2}{1 - 0.5z^{-1}}. \tag{3.31}$$

If we multiply both sides, first by $(1 - z^{-1})(1 - 0.5z^{-1})$, and then by z, we obtain

$$z + 1 = A_1(z - 0.5) + A_2(z - 1), \tag{3.32}$$

which must hold for all z. If we set $z = 1$, we find that $A_1 = 0.4$, whereas for $z = 0.5$ we find that $A_2 = -3$.

To find the sequences corresponding to the partial fractions, we need to know their ROC. The pole-zero plot of $X(z)$ is given in Figure 3.6. Since a ROC cannot include any poles, there are three possible choices for valid ROCs.

If ROC: $|z| > 1$, both fractions are the z-transform of causal sequences. Hence

$$x[n] = 4u[n] - 3\left(\frac{1}{2}\right)^n u[n]. \quad \text{(causal)} \tag{3.33}$$

If ROC: $|z| < 0.5$, both fractions are the z-transform of anticausal sequences. Hence

$$x[n] = -4u[-n - 1] + 3\left(\frac{1}{2}\right)^n u[-n - 1]. \quad \text{(anticausal)} \tag{3.34}$$

If ROC: $0.5 < |z| < 1$, this can be obtained as the intersection of ROC: $|z| < 1$ and ROC:$|z| > 0.5$. Hence, using z-transform pairs 3 and 4 in Table 3.1, we obtain

$$x[n] = -4u[-n - 1] - 3\left(\frac{1}{2}\right)^n u[n]. \quad \text{(two-sided)} \tag{3.35}$$

■

Example 3.9   Complex conjugate distinct poles

To illustrate this case consider a causal sequence $x[n]$ with z-transform

$$X(z) = \frac{1 + z^{-1}}{1 - z^{-1} + 0.5z^{-2}}. \tag{3.36}$$

The poles, obtained by solving the quadratic equation $z^2 - z + 0.5 = 0$, are

$$p_1 = \frac{1}{2}(1 + j) = \frac{1}{\sqrt{2}}e^{j\pi/4} \quad \text{and} \quad p_2 = \frac{1}{2}(1 - j) = \frac{1}{\sqrt{2}}e^{-j\pi/4}.$$

Since $p_1 \neq p_2$, we have

$$X(z) = \frac{1 + z^{-1}}{1 - z^{-1} + 0.5z^{-2}} = \frac{A_1}{1 - p_1 z^{-1}} + \frac{A_2}{1 - p_2 z^{-1}}. \tag{3.37}$$

If we multiply both sides, first by $(1 - p_1 z^{-1})(1 - p_2 z^{-1})$, and then by z, we obtain

$$z + 1 = A_1(z - p_2) + A_2(z - p_1). \tag{3.38}$$

Setting $z = p_1$ and $z = p_2$, we solve for A_1 and A_2, respectively, in (3.38)

$$A_1 = \frac{1}{2} - j\frac{3}{2} = \frac{\sqrt{10}}{2}e^{-j71.56°} \quad \text{and} \quad A_2 = \frac{1}{2} + j\frac{3}{2} = \frac{\sqrt{10}}{2}e^{j71.56°}.$$

Note that, because the polynomial has real coefficients, $p_1 = p_2^*$ and $A_1 = A_2^*$. Since $x[n]$ is causal, each term in (3.37) results in a causal sequence. Hence,

$$x[n] = A_1(p_1)^n u[n] + A_1^*(p_1^*)^n u[n]. \tag{3.39}$$

Using the polar expressions $A_1 = Ae^{j\theta}$, $p_1 = re^{j\omega_0}$ and Euler's identity, we obtain

$$x[n] = Ar^n\left(e^{j\omega_0 n}e^{j\theta} + e^{-j\omega_0 n}e^{-j\theta}\right)u[n] = 2Ar^n\cos(\omega_0 n + \theta)u[n], \tag{3.40}$$

where $r = 1/\sqrt{2}$, $\omega_0 = \pi/4$, $A = \sqrt{10}/2$, and $\theta = -71.56°$. ■

If we have a rational function with distinct poles

$$X(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \tag{3.41}$$

the complete partial fraction expansion takes the form

$$X(z) = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}, \tag{3.42}$$

where

$$A_k = (1 - p_k z^{-1})X(z)|_{z=p_k}, \tag{3.43}$$

and $C_k = 0$ when $M < N$, that is, when the rational function is proper.

The parameters $\{C_k, p_k, A_k\}$ in expansion (3.42) can be computed using the MATLAB function

$$[A,p,C]=residuez(b,a) \tag{3.44}$$

whose use is illustrated in the next example. Function `residuez` can handle multiple (that is, repeated) poles; however, this case is not encountered often in practical applications.

Example 3.10 Partial fraction expansion using `residuez`

The following expansion:

$$X(z) = \frac{6 - 10z^{-1} + 2z^{-2}}{1 - 3z^{-1} + 2z^{-2}} = 1 + \frac{2}{1 - z^{-1}} + \frac{3}{1 - 2z^{-1}}, \tag{3.45}$$

is obtained by calling `residuez` with b=[6,-10,2] and a=[1,-3,2]. The reverse operation can be done using the same function as: [b,a]=residuez(A,p,C).

```
>> b=[6 -10 2];
>> a=[1 -3 2];
>> [A,p,C]=residuez(b,a)
A =
      3
      2
p =
      2
      1
C =
      1
>> [b,a]=residuez(A,p,C)
b =
      6    -10     2
a =
      1     -3     2
```

Working as in Example 3.8 we obtain the following sequences

$$x[n] = \delta[n] + (2 + 3 \times 2^n)u[n], \qquad\qquad \text{ROC: } |z| > 2$$

$$x[n] = \delta[n] - (2 + 3 \times 2^n)u[-n-1], \qquad\qquad \text{ROC: } |z| < 1$$

$$x[n] = \delta[n] + 2u[n] - 3 \times 2^n u[-n-1]. \qquad \text{ROC: } 1 < |z| < 2$$

■

3.4   Properties of the *z*-transform

Since there is a unique correspondence between a sequence $x[n]$ and its z-transform $X(z)$, we can change a sequence by manipulating its z-transform. In this section we discuss some key properties of z-transforms that are useful in the analysis and design of LTI systems. Additional properties will be introduced as needed.

Linearity The z-transform is a linear operator, that is,

$$a_1 x_1[n] + a_2 x_2[n] \overset{\mathcal{Z}}{\longleftrightarrow} a_1 X_1(z) + a_2 X_2(z), \quad \text{ROC contains } R_{x_1} \cap R_{x_2} \qquad (3.46)$$

which follows directly from the definition (3.9). We emphasize that when we combine z-transforms the resulting z-transform exists only if the ROCs of all individual transforms overlap (see Example 3.6). Hence, the ROC of the linear combination in (3.46) is at least the intersection of R_{x_1} and R_{x_2}.

Time shifting Consider

$$x[n-k] \overset{\mathcal{Z}}{\longleftrightarrow} z^{-k}X(z). \quad \text{ROC} = R_x(\text{except } z = 0 \text{ or } z = \infty) \qquad (3.47)$$

If $k > 0$ ($k < 0$), the original sequence $x[n]$ is shifted right (left) and the shifting introduces a pole at $z = 0$ ($z = \infty$). Clearly, this pole should be excluded from the ROC of the resulting z-transform.

To prove (3.47), we set $y[n] = x[n-k]$, substitute into the definition (3.9), and change the index of summation from n to $m = n - k$. More specifically,

$$Y(z) = \sum_{n=-\infty}^{\infty} x[n-k]z^{-n} = \sum_{m=-\infty}^{\infty} x[m]z^{-(m+k)} = z^{-k} \sum_{m=-\infty}^{\infty} x[m]z^{-m}, \qquad (3.48)$$

which leads to (3.47).

Example 3.11

Consider the sequence

$$x[n] = \begin{cases} 1, & 0 \le n \le N - 1 \\ 0. & \text{otherwise} \end{cases} \qquad (3.49)$$

Using the definition (3.9) we have

$$X(z) = \sum_{n=0}^{N-1} 1z^{-n} = 1 + z^{-1} + \cdots + z^{-(N-1)} = \begin{cases} N, & z = 1 \\ \dfrac{1 - z^{-N}}{1 - z^{-1}}, & z \neq 1. \end{cases} \tag{3.50}$$

The ROC is the entire z-plane, except $z = 0$.

The sequence $x[n]$ can be written, using the unit step sequence $u[n]$, as $x[n] = u[n] - u[n-N]$. Applying the linearity and time-shifting properties, we obtain

$$X(z) = U(z) - z^{-N}U(z) = (1 - z^{-N})U(z) = \frac{1 - z^{-N}}{1 - z^{-1}}. \tag{3.51}$$

■

Convolution of sequences Convolving two sequences is equivalent to multiplying their z-transforms:

$$x_1[n] * x_2[n] \overset{\mathcal{Z}}{\longleftrightarrow} X_1(z)X_2(z). \quad \text{ROC contains } R_{x_1} \cap R_{x_2} \tag{3.52}$$

This property is a consequence of linearity and time shifting properties. Indeed, applying successively the linearity and time shifting properties to the convolution summation

$$y[n] = \sum_{k=-\infty}^{\infty} x_1[k]x_2[n-k], \tag{3.53}$$

we obtain $Y(z) = \sum_k x_1[k]z\{x_2[n-k]\}$ and

$$Y(z) = \sum_{k=-\infty}^{\infty} x_1[k]z^{-k}X_2(z) = \left(\sum_{k=-\infty}^{\infty} x_1[k]z^{-k} \right) X_2(z) = X_1(z)X_2(z). \tag{3.54}$$

Since $Y(z)$ is obtained by multiplying $X_1(z)$ and $X_2(z)$, its ROC should include the intersection of the ROCs of $X_1(z)$ and $X_2(z)$. The convolution property plays a very important role in the analysis of LTI systems (see Section 3.5).

Polynomial multiplication in MATLAB The convolution theorem (3.52) shows that polynomial multiplication is equivalent to convolution. Therefore, to compute the product

$$B(z) = (1 + 2z^{-2})(1 + 4z^{-1} + 2z^{-2} + 3z^{-3})$$
$$= 1 + 4z^{-1} + 4z^{-2} + 11z^{-3} + 4z^{-4} + 6z^{-5},$$

we use the function

```
>> b=conv([1 0 2],[1 4 2 3])
b =
    1    4    4    11    4    6
```

to find the coefficients of $B(z)$.

Multiplication by an exponential sequence According to this property

$$a^n x[n] \xleftrightarrow{\mathcal{Z}} X(z/a). \quad \text{ROC} = |a|R_x \tag{3.55}$$

Indeed, if $y[n] = a^n x[n]$, we have

$$Y(z) = \sum_{n=-\infty}^{\infty} a^n x[n] z^{-n} = \sum_{n=-\infty}^{\infty} x[n](z/a)^{-n} = X(z/a). \tag{3.56}$$

The value $X(z_1)$, taken by $X(z)$ at $z = z_1$, is taken by $Y(z)$ at $Y(az_1) = X(az_1/a) = X(z_1)$. Hence, we have a mapping from $z \to az$. Since a and z take complex values, the result is scaling (expansion or shrinking) and rotation of the *z*-plane, the ROC, and the pole-zero pattern.

Differentiation of the *z*-transform $X(z)$ Multiplying the value of each sample $x[n]$ by its index n, is equivalent to differentiating $X(z)$. More specifically,

$$nx[n] \xleftrightarrow{\mathcal{Z}} -z\frac{\mathrm{d}X(z)}{\mathrm{d}z}, \quad \text{ROC} = R_x \tag{3.57}$$

which is obtained by differentiating both sides of (3.9). Indeed, we have

$$\frac{\mathrm{d}X(z)}{\mathrm{d}z} = \sum_{n=-\infty}^{\infty} x[n](-n)z^{-n-1} = -z^{-1} \sum_{n=-\infty}^{\infty} nx[n]z^{-n}, \tag{3.58}$$

which leads to (3.57). Note that both sequences have the same ROC.

Example 3.12 Second-order pole
We shall compute the *z*-transform of the sequence

$$x[n] = na^n u[n] = n(a^n u[n]),$$

using the differentiation property. From *z*-transform pair 3 in Table 3.1 and the differentiation property (3.57), we have

$$X(z) = -z\frac{\mathrm{d}}{\mathrm{d}z}\left(\frac{1}{1 - az^{-1}}\right) = \frac{az^{-1}}{(1 - az^{-1})^2}, \quad |z| > |a|$$

which is *z*-transform pair 5 in Table 3.1. ∎

Conjugation of a complex sequence By the conjugation property,

$$x^*[n] \xleftrightarrow{\mathcal{Z}} X^*(z^*). \quad \text{ROC} = R_x \tag{3.59}$$

The proof is easily obtained by conjugating both sides of the definition of the *z*-transform (3.9).

Table 3.2 Some z-transform properties.

	Property	Sequence	Transform	ROC
		$x[n]$	$X(z)$	R_x
		$x_1[n]$	$X_1(z)$	R_{x_1}
		$x_2[n]$	$X_2(z)$	R_{x_2}
1.	Linearity	$a_1x_1[n] + a_2x_2[n]$	$a_1X_1(z) + a_2X_2(z)$	At least $R_{x_1} \bigcap R_{x_2}$
2.	Time shifting	$x[n - k]$	$z^{-k}X(z)$	R_x except $z = 0$ or ∞
3.	Scaling	$a^nx[n]$	$X(a^{-1}z)$	$\lvert a\rvert R_x$
4.	Differentation	$nx[n]$	$-z\dfrac{dX(z)}{dz}$	R_x
5.	Conjugation	$x^*[n]$	$X^*(z^*)$	R_x
6.	Real-part	$\mathrm{Re}\{x[n]\}$	$\frac{1}{2}[X(z) + X^*(z^*)]$	At least R_x
7.	Imaginary part	$\mathrm{Im}\{x[n]\}$	$\frac{1}{2}[X(z) - X^*(z^*)]$	At least R_x
8.	Folding	$x[-n]$	$X(1/z)$	$1/R_x$
9.	Convolution	$x_1[n] * x_2[n]$	$X_1(z)X_2(z)$	At least $R_{x_1} \bigcap R_{x_2}$
10.	Initial-value theorem	$x[n] = 0$ for $n < 0$	$x[0] = \lim\limits_{z\to\infty} X(z)$	

Time reversal The time reversal or folding property is expressed as

$$x[-n] \overset{\mathcal{Z}}{\longleftrightarrow} X(1/z). \quad \text{ROC} = \frac{1}{R_x} \tag{3.60}$$

The proof is easily obtained by conjugating both sides of the definition of the z-transform (3.9). The notation ROC $= 1/R_x$ means that R_x is inverted; that is, if $R_x = \{r_1 < \lvert z\rvert < r_2\}$, then $1/R_x = \{1/r_2 < \lvert z\rvert < 1/r_1\}$.

Initial-value theorem If $x[n]$ is a causal sequence, that is, $x[n] = 0$ for $n < 0$, then

$$x[0] = \lim_{z\to\infty} X(z), \tag{3.61}$$

which is obtained by considering the limit of each term in the z-transform summation.

**Summary of properties** For convenience, the properties of the z-transforms are summarized in Table 3.2.

3.5   System function of LTI systems

In Section 2.4 we showed that every LTI can be completely characterized in the time domain by its impulse response $h[n]$. In this respect, using the impulse response $h[n]$, we can compute the output of the system for any input via the convolution summation
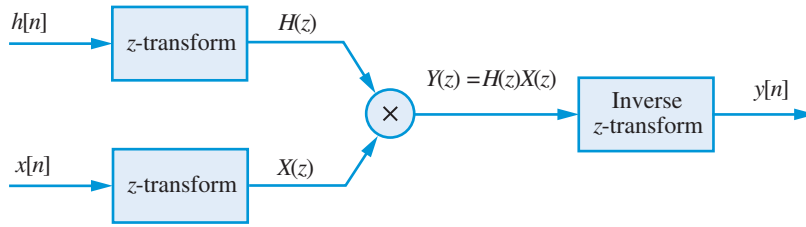
Figure 3.7 Procedure for the analytical computation of the output of an LTI system using the convolution theorem of the *z*-transform.

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k], \tag{3.62}$$

and check whether the system is causal and stable. In this section, we answer the same questions using *z*-transform techniques.

Input-output relationship From (3.62) and the convolution property (3.52) we obtain

$$Y(z) = H(z)X(z), \tag{3.63}$$

where $X(z)$, $Y(z)$, and $H(z)$ are the *z*-transforms of the system input, output, and impulse response, respectively. From our discussion in Section 3.1, $H(z)$ is known as the system function or transfer function of the system. Since there is a unique relation between $h[n]$ and $H(z)$ many properties of the system can be inferred from $H(z)$ and its ROC. Equation (3.63) provides a convenient approach for the analytical evaluation of convolution using the *z*-transform (see Figure 3.7). We stress that, the only requirement for (3.63) to hold is that the ROCs of $H(z)$ and $X(z)$ overlap.

Example 3.13

We shall determine the response of a system with impulse response $h[n] = a^n u[n], |a| < 1$ to the input $x[n] = u[n]$ using the convolution theorem. The system function and the *z*-transform of the input sequence are

$$H(z) = \sum_{n=0}^{\infty} a^n z^{-n} = \frac{1}{1-az^{-1}}, \quad |z| > |a| \tag{3.64}$$

and

$$X(z) = \sum_{n=0}^{\infty} z^{-n} = \frac{1}{1-z^{-1}}. \quad |z| > 1. \tag{3.65}$$

Since the ROCs of $H(z)$ and $X(z)$ always overlap, the *z*-transform of $Y(z)$ is

$$Y(z) = \frac{1}{(1-az^{-1})(1-z^{-1})}. \quad |z| > \max\{|a|, 1\} = 1. \tag{3.66}$$

The output sequence $y[n]$ can be obtained by determining the inverse z-transform. Using partial fraction expansion, we get

$$Y(z) = \frac{1}{1-a}\left(\frac{1}{1-z^{-1}} - \frac{a}{1-az^{-1}}\right). \qquad |z| > 1 \qquad (3.67)$$

Therefore,

$$y[n] = \frac{1}{1-a}(u[n] - a^{n+1}u[n]) = \frac{1-a^{n+1}}{1-a}u[n], \qquad (3.68)$$

which is exactly the steady-state response derived in Section 2.10. ∎

Causality A causal LTI system has an impulse response $h[n]$ that is zero for $n < 0$. Therefore, for causal systems the power series

$$H(z) = \sum_{n=0}^{\infty} h[n]z^{-n} \qquad (3.69)$$

does not include any positive powers of z and its ROC extends outside of a circle for some radius r, that is, $|z| > r$. Since every right-sided sequence (causal or noncausal) has ROC $|z| > r$ for some r, we have the following property:

Result 3.5.1 A system function $H(z)$ with the ROC that is the exterior of a circle, extending to infinity, is a necessary condition for a discrete-time LTI system to be causal but not a sufficient one.

Stability For a LTI system to be stable, the impulse response must be absolutely summable, that is,

$$\sum_{n=-\infty}^{\infty} |h[n]| < \infty. \qquad (3.70)$$

This is equivalent to the condition

$$|H(z)| \le \sum_{n=-\infty}^{\infty} |h[n]z^{-n}| < \infty, \qquad (3.71)$$

for $|z| = 1$; this implies that the ROC of $H(z)$ must include the unit circle. Therefore:

Result 3.5.2 A LTI system is stable if and only if the ROC of the system function $H(z)$ includes the unit circle $|z| = 1$.

Causal and stable system Combining the above two properties, we can now state that:

Result 3.5.3 An LTI system with rational $H(z)$ is both causal and stable if and only if all poles of $H(z)$ are *inside* the unit circle *and* its ROC is on the exterior of a circle, extending to infinity.

Example 3.14
The ROC for the system function in Example 3.8 is $|z| > |a|$. If $|a| < 1$, the ROC includes the unit circle, and the system is stable. A time domain derivation of the same condition is given in Example 2.5. ■

Stability and causality are not interrelated properties; a causal system may or may not be stable, and vice versa.

System function algebra The z-transform allows the replacement of time-domain operations, like time shifting and convolution, with simpler algebraic operations. This leads to a system function algebra which facilitates the analysis and synthesis of LTI systems involving in-series, parallel, and feedback interconnections of simpler system building blocks.

Consider the parallel interconnection of two systems, as shown in Figure 3.8(a). The impulse response of the overall system is

$$h[n] = h_1[n] + h_2[n], \tag{3.72}$$

and from the linearity of the z-transform, we have

$$H(z) = H_1(z) + H_2(z). \tag{3.73}$$

(a)   (b)
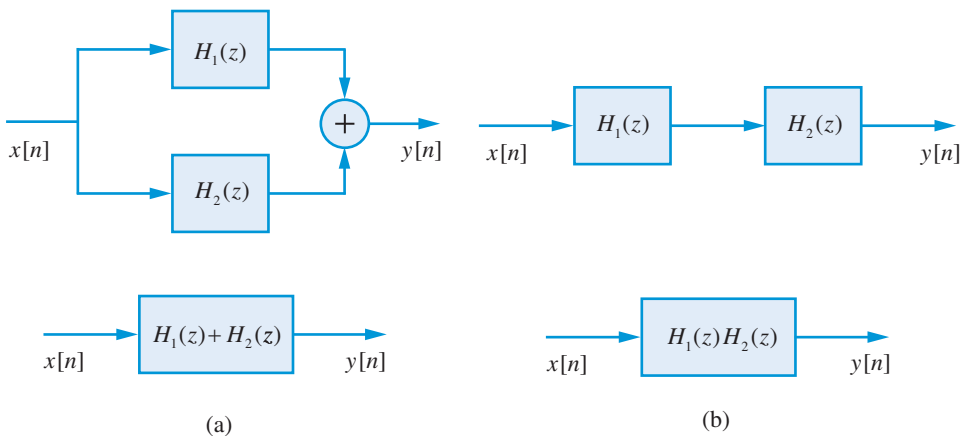
Figure 3.8 Equivalent system function of linear time-invariant systems combined in (a) parallel connection, and (b) cascade connection.

The power of the *z*-transform is more evident when we deal with series interconnection of LTI systems. The impulse response of the overall system in Figure 3.8(b) is

$$h[n] = h_1[n] * h_2[n], \tag{3.74}$$

and from the convolution property of the *z*-transform, we have

$$H(z) = H_1(z)H_2(z). \tag{3.75}$$

These results can be used, in a straightforward manner, to analyze more complex interconnections of LTI systems.

<table>
<tr><td>**3.6**</td><td>## LTI systems characterized by linear constant-coefficient difference equations</td></tr>
</table>

In Section 2.10, we introduced a class of LTI systems whose input and output sequences satisfy a linear constant-coefficient difference equation of the form

$$y[n] + \sum_{k=1}^{N} a_k y[n-k] = \sum_{k=0}^{M} b_k x[n-k]. \tag{3.76}$$

If we assume that the system is causal, we have

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k], \tag{3.77}$$

which can be used to compute the output recursively starting with a set of initial conditions. In signal processing applications, we assume that the system is *initially at rest*, that is, $x[n] = y[n] = 0$ for $n < 0$. Therefore, we can set the initial conditions at $n = 0$ to zero, that is, $x[-1] = \cdots = x[-M] = 0$ and $y[-1] = \cdots = y[-N] = 0$, and then recursively compute the output values $y[0], y[1], \ldots, y[L]$.

In Section 2.10, we mentioned that the *z*-transform is the powerful tool we need for the analysis of systems described by linear constant-coefficient difference equations. We first show that any system described by (3.76) and initially at rest is linear and time-invariant, by showing that it can be expressed in the form $Y(z) = H(z)X(z)$.

Since $x[n]$, $y[n]$ are defined for all n, we can apply the *z*-transform on both sides of (3.76). Using the properties of linearity and time shifting, we obtain

$$\left(1 + \sum_{k=1}^{N} a_k z^{-k}\right) Y(z) = \left(\sum_{k=0}^{M} b_k z^{-k}\right) X(z). \tag{3.78}$$

The system function, obtained using (3.78) and (3.63), is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\displaystyle\sum_{k=0}^{M} b_k z^{-k}}{1 + \displaystyle\sum_{k=1}^{N} a_k z^{-k}}, \tag{3.79}$$

where $H(z)$ is a rational function, that is, the ratio of two polynomials in z^{-1}. Noting that terms of the form $b_k x[n-k]$ correspond to $b_k z^{-k}$ and terms of the form $a_k y[n-k]$ correspond to $a_k z^{-k}$, it is straightforward to obtain the difference equation from the system function and vice versa. From (3.79) we obtain $Y(z) = H(z)X(z)$, hence the system (3.76) is LTI.

Example 3.15

Find the difference equation corresponding to the system function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{6 - 10z^{-1} + 2z^{-2}}{1 - 3z^{-1} + 2z^{-2}}. \tag{3.80}$$

Cross-multiplying the numerator and denominator terms in (3.80), we obtain

$$(1 - 3z^{-1} + 2z^{-2})Y(z) = (6 - 10z^{-1} + 2z^{-2})X(z).$$

Thus, the difference equation is

$$y[n] - 3y[n-1] + 2y[n-2] = 6x[n] - 10x[n-1] + 2x[n-2].$$

If we assume that the system is causal, we have

$$y[n] = 3y[n-1] - 2y[n-2] + 6x[n] - 10x[n-1] + 2x[n-2].$$

With some practice, the conversion from the difference equation to system function and vice versa can be done by simple inspection. ■

Poles and zeros From the fundamental theorem of algebra, we recall that a polynomial of degree M has M roots. Hence, if $z_1, z_2, \ldots, z_M$ are the roots of the numerator polynomial, we have

$$B(z) = b_0 z^{-M} \left(z^M + \frac{b_1}{b_0} z^{M-1} + \cdots + \frac{b_M}{b_0} \right)$$

$$= b_0 z^{-M} (z - z_1) \ldots (z - z_M). \tag{3.81}$$

Similarly, if $p_1, p_2, \ldots, p_N$ are the roots of the denominator polynomial, we have

$$A(z) = z^{-N} \left(z^N + a_1 z^{N-1} + \cdots + a_N \right)$$
$$= z^{-N}(z - p_1) \ldots (z - p_N). \tag{3.82}$$

Therefore,

$$H(z) = \frac{B(z)}{A(z)} = b_0 \frac{z^{-M}}{z^{-N}} \frac{\displaystyle\prod_{k=1}^{M}(z - z_k)}{\displaystyle\prod_{k=1}^{N}(z - p_k)} = b_0 \frac{\displaystyle\prod_{k=1}^{M}(1 - z_k z^{-1})}{\displaystyle\prod_{k=1}^{N}(1 - p_k z^{-1})}, \tag{3.83}$$

where b_0 is a constant gain term. Each of the factors $(1 - z_k z^{-1})$ contributes a zero at $z = z_k$ and a pole at $z = 0$. Similarly, each of the factors $(1 - p_k z^{-1})$ contributes a pole at $z = p_k$ and a zero at $z = 0$. Poles and zeros at the origin are not counted.

Impulse response From (3.42) we recall that any rational function of z^{-1} with distinct poles can be expressed in the form

$$H(z) = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}, \tag{3.84}$$

where the first summation is included only if $M \geq N$. If we assume that the system is causal, then the ROC is the exterior of a circle starting at the outermost pole, and the impulse response is

$$h[n] = \sum_{k=0}^{M-N} C_k \delta[n - k] + \sum_{k=1}^{N} A_k (p_k)^n u[n]. \tag{3.85}$$

Causality and stability The difference equation (3.76) does not uniquely specify the impulse response of a LTI system because there are a number of choices for the ROC of the system function (3.79) (see Example 3.8). Therefore, without additional information or assumptions we cannot make any inferences about the causality and stability of the system.

In all practical applications, where we assume that the system is causal, the impulse response is a causal sequence given by (3.85). For this causal system to be stable, the impulse response must be absolutely summable, that is,

$$\sum_{n=0}^{\infty} |h[n]| \leq \sum_{k=0}^{M-N} |C_k| + \sum_{k=1}^{N} |A_k| \sum_{n=0}^{\infty} |p_k|^n < \infty. \tag{3.86}$$

This is possible if and only if $|p_k| < 1$ for $k = 1, \ldots, N$. Hence, the condition for stability is:

Result 3.6.1 A causal LTI with a rational system function is stable if and only if all poles of $H(z)$ are inside the unit circle in the z-plane. The zeros can be anywhere.

System classifications Linear time-invariant systems can be classified into different classes based on the length of their impulse response, the presence of feedback in their implementation, and pole-zero pattern:

- Length of impulse response
  If at least one nonzero pole of $H(z)$ is not canceled by a zero, there will a term of the form $A_k(p_k)^n u[n]$ in (3.85). In this case $h[n]$ has infinite duration and the system is called an *Infinite Impulse Response (IIR)* system.
  If $N = 0$, the system function (3.79) becomes a polynomial. The impulse response is given by

$$h[n] = \sum_{k=0}^{M} b_k \delta[n - k] = \begin{cases} b_n, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases} \tag{3.87}$$

  and the corresponding systems are called *Finite Impulse Response (FIR)* systems.
- Feedback in implementation
  If $N \ge 1$ the output of the system is fed back into the input and the system is known as a *recursive* system.
  If $N = 0$ the output is a linear combination of the present and previous inputs, only. Such systems are called *nonrecursive* systems.
- Poles and zeros
  If $a_k = 0$, for $k = 1, \ldots, N$, the system has M zeros (*all-zero* systems).
 If $b_k = 0$, for $k = 1, \ldots, M$, the system has N poles (*all-pole* systems).

Example 3.16   Third-order IIR system
Consider a causal system defined by

$$H(z) = \frac{1 - z^{-2}}{1 + 0.9z^{-1} + 0.6z^{-2} + 0.05z^{-3}}. \tag{3.88}$$

This is a third-order system with $M = 2$ zeros and $N = 3$ poles. To check whether the system is stable, we use the function `zplane(b,a)` with `b=[1,0,-1]` and `a=[1,0.9, 0.6,0.05]`. This results in the pole-zero plot shown in Figure 3.9. Since the poles are inside the unit circle, the causal system (3.88) is stable. We recall that although zeros at $z = 0$ are not counted when we determine M, they are plotted by the function `zplane`. An analytical expression for the impulse response $h[n]$ can be determined from $H(z)$ using partial fraction expansion (see Problem 21). However, we can evaluate and plot L samples of $h[n]$ numerically using the function `impz(b,a,L)`. In this sense, we can say that `impz` computes the inverse z-transform of a rational system function $H(z)$. ∎
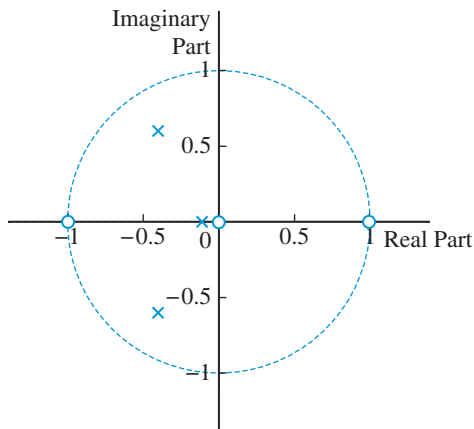
Figure 3.9 Pole-zero plot for the system function given by (3.88). This causal system is stable because its poles are inside the unit circle.

Analysis of LTI systems with MATLAB In practice we deal with FIR systems (causal or noncausal) and causal IIR systems specified by difference equations of the form (3.77). The analysis of causal systems with rational system functions using MATLAB involves the following steps:

1. Determine the coefficient vectors a and b from the difference equation (3.77) or the system function (3.79).
2. Plot the pole-zero pattern with function zplane(b,a). If the poles are inside the unit circle, we conclude that the system is stable.
3. Compute and plot the impulse response $h[n]$ using the function impz(b,a,L), with $L = 100$. If the system is stable, the values of $h[n]$ should asymptotically tend to zero. After inspecting the plot, choose L so that only the "nonzero" values of $h[n]$ are included in the plot.
4. Compute the response $y[n]$ to an arbitrary input $x[n]$ using the function y=filter(b,a,x).

3.7   ## Connections between pole-zero locations and time-domain behavior

We saw that rational system functions with distinct poles can be decomposed as

$$H(z) = \frac{B(z)}{A(z)} = \frac{\displaystyle\sum_{k=0}^{M} b_k z^{-k}}{1 + \displaystyle\sum_{k=1}^{N} a_k z^{-k}} = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{N} \frac{A_k}{1 - p_k z^{-1}}, \qquad (3.89)$$

where the first summation is included only if $M \geq N$ and A_k is given by

$$A_k = (1 - p_k z^{-1}) H(z) \Big|_{z=p_k}. \tag{3.90}$$

Equation (3.89) shows that an Nth order system can be obtained by connecting an $(M - N)$th order FIR system and N first-order systems. The coefficients C_k are always real; however, the poles p_k may be real or complex. To avoid the use of first-order systems with complex coefficients, we use the following result:

Result 3.7.1 The roots of a polynomial with real coefficients either must be real or must occur in complex conjugate pairs.

To prove this result, suppose that p_k is a root of a polynomial $A(z) = \sum_{k=0}^{N} a_k z^{-k}$, that is, $A(p_k) = 0$. Taking the complex conjugate of $\sum_{k=0}^{N} a_k p_k^{-k}$, we have

$$\left(\sum_{k=0}^{N} a_k p_k^{-k} \right)^* = \sum_{k=0}^{N} a_k (p_k^*)^{-k} = A(p_k^*) = 0, \tag{3.91}$$

which shows that if p_k is a complex root, its complex conjugate p_k^* is also a root.

Suppose that $p = r e^{j\omega_0}$ is a pole with partial fraction expansion coefficient $A = |A| e^{j\theta}$. Then, the complex conjugate $p^* = r e^{-j\omega_0}$ is also a pole. From (3.43) it follows that the partial expansion coefficient is $A^* = |A| e^{-j\theta}$. Hence, every pair of complex conjugate poles contributes the term

$$\frac{A}{1 - pz^{-1}} + \frac{A^*}{1 - p^* z^{-1}} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}}, \tag{3.92}$$

which is a proper rational function with real coefficients

$$b_0 = 2\mathcal{R}e(A) = 2|A| \cos\theta, \tag{3.93a}$$

$$b_1 = -2\mathcal{R}e(Ap^*) = -2r|A| \cos(\omega_0 - \theta), \tag{3.93b}$$

$$a_1 = -2\mathcal{R}e(p) = -2r \cos\omega_0, \tag{3.93c}$$

$$a_2 = |p|^2 = r^2. \tag{3.93d}$$

We conclude that any system with a rational $H(z)$ is equivalent to a parallel combination of an FIR system, K_1 first-order systems with real poles, and K_2 second-order systems with complex conjugate poles, where $N = K_1 + 2K_2$. More specifically

$$H(z) = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{K_1} \frac{A_k}{1 - p_k z^{-1}} + \sum_{k=1}^{K_2} \frac{b_{k0} + b_{k1} z^{-1}}{1 + a_{k1} z^{-1} + a_{k2} z^{-2}}. \tag{3.94}$$

Therefore, the behavior of a system with a rational system function can be understood in terms of the behavior of a first-order system with a real pole and a second-order system with complex conjugate poles.

3.7.1   First-order systems

Each first-order term in (3.94) corresponds to a system with system function

$$H(z) = \frac{b}{1 - az^{-1}}. \quad a, b \text{ real} \tag{3.95}$$

Assuming a causal system, the impulse response is given by the following real exponential sequence:

$$h[n] = ba^n u[n]. \tag{3.96}$$

The system is stable if and only if the pole $p = a$ is inside the unit circle, that is, $-1 < a < 1$. Figure 3.10 shows how the location of the real pole $p = a$ affects the shape of the impulse response.
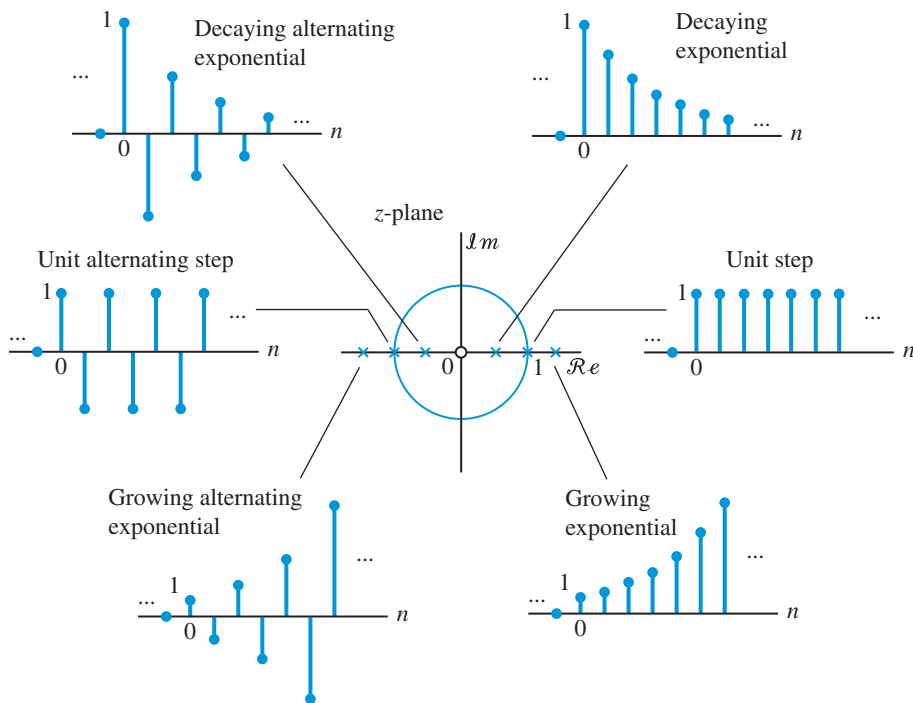
Figure 3.10 Impulse responses associated with real poles in the z-plane. Only the two poles inside the unit circle correspond to stable systems.

3.7.2 Second-order systems

Each second-order term in (3.94) corresponds to a system with system function

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{z(b_0 z + b_1)}{z^2 + a_1 z + a_2}. \tag{3.97}$$

The system has two real zeros at $z_1 = 0$ and $z_2 = -b_1/b_0$. The poles are obtained by solving the quadratic equation $z^2 + a_1 z + a_2 = 0$. The result is

$$p_{1,2} = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2}}{2}. \tag{3.98}$$

There are three possible cases:

Poles	Condition
Real and distinct	$a_1^2 > 4a_2$
Real and equal	$a_1^2 = 4a_2$
Complex conjugate	$a_1^2 < 4a_2$

The impulse response of a causal system with a pair of complex conjugate poles can be found as follows (see (3.92)):

$$h[n] = A p^n u[n] + A^* (p^*)^n u[n]$$

$$= |A| e^{j\theta} r^n e^{j\omega_0 n} u[n] + |A| e^{-j\theta} r^n e^{-j\omega_0 n} u[n]$$

$$= |A| r^n \left[e^{j(\omega_0 n + \theta)} + e^{-j(\omega_0 n + \theta)} \right] u[n]$$

$$= 2|A| r^n \cos(\omega_0 n + \theta) u[n],$$

where ω_0 and r are obtained by (3.93c) and (3.93a). Therefore, the impulse response of a causal second-order system with complex conjugate poles is a sinusoidal sequence with exponentially varying amplitude. More specifically

$$h[n] = 2|A| r^n \cos(\omega_0 n + \theta) \, u[n], \tag{3.99}$$

where r determines the type of the exponential amplitude variation and ω_0 determines the frequency of the sinusoidal oscillation. The coefficients b_0 and b_1 or equivalently the two zeros of the system simply introduce a constant scaling factor and a constant phase shift.

From (3.99) it can easily be seen that the system is stable if $r < 1$, that is, if the complex conjugate poles are inside the unit circle. Figure 3.11 illustrates how the location of the poles affects the shape of the impulse response and the stability of the system.

Pole locations   Impulse response
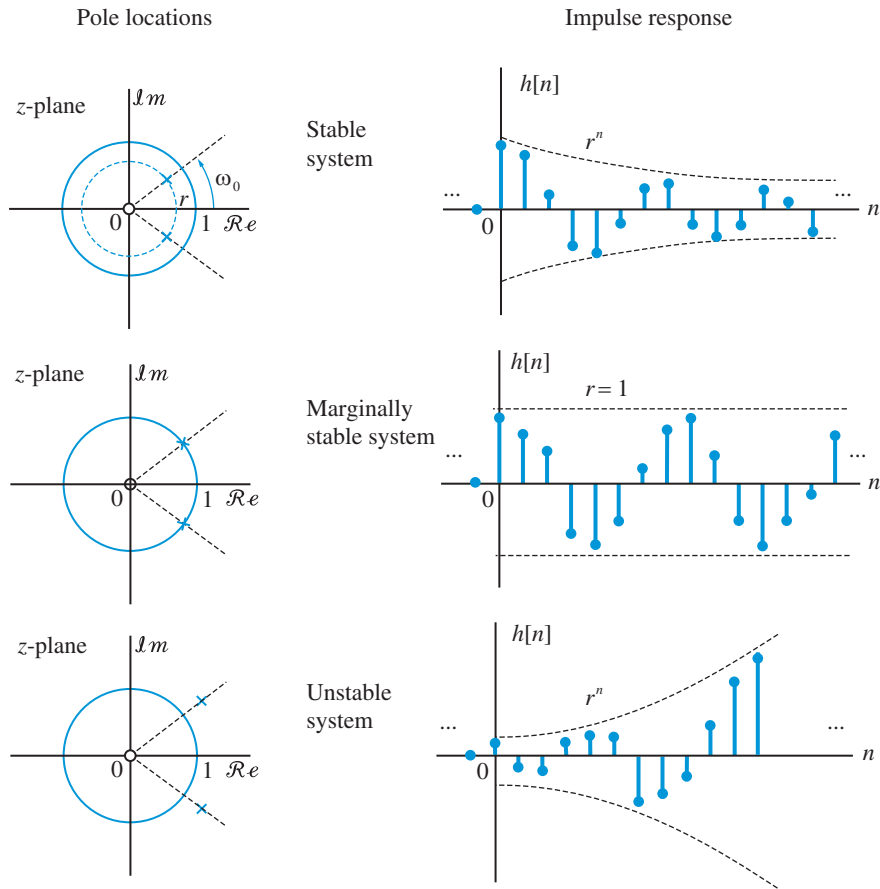
Figure 3.11 Impulse responses associated with a pair of complex conjugate poles in the z-plane. Only the two poles inside the unit circle correspond to stable systems.

3.8   The one-sided z-transform

The z-transform defined by (3.9) is known as the *two-sided* or *bilateral* z-transform because it represents the entire two-sided sequence. However, there are problems whose solutions require use of the *one-sided* or *unilateral* z-transform, defined by the formula

$$X^+(z) \triangleq \mathcal{Z}^+\{x[n]\} \triangleq \sum_{n=0}^{\infty} x[n]z^{-n}. \tag{3.100}$$

The difference between the one-sided z-transform and the two-sided z-transform is that the lower limit of the sum in (3.100) is always zero, regardless of the values of $x[n]$ for $n < 0$. Therefore, sequences which are equal for $n \geq 0$ and differ for $n < 0$ have the same one-sided z-transform. Since $\mathcal{Z}^+\{x[n]\} = \mathcal{Z}\{x[n]u[n]\}$, the ROC of $X^+(z)$ is always the exterior of a circle.

Almost all properties we have studied for the two-sided z-transform carry over to the one-sided z-transform with the exception of the time shifting property. To illustrate this property, we shall determine $\mathcal{Z}^+\{x[n-1]\}$. From (3.100), we have

$$
\begin{aligned}
\mathcal{Z}^+\{x[n-1]\} &= x[-1] + x[0]z^{-1} + x[1]z^{-2} + x[2]z^{-3} + \cdots \\
&= x[-1] + z^{-1}(x[0] + x[1]z^{-1} + x[2]z^{-2} + \cdots) \\
&= x[-1] + z^{-1}X^+(z).
\end{aligned}
\tag{3.101}
$$

In a similar fashion, we can show that

$$
\mathcal{Z}^+\{x[n-2]\} = x[-2] + x[-1]z^{-1} + z^{-2}X^+(z).
\tag{3.102}
$$

In general, for any $k > 0$, we can show that

$$
\mathcal{Z}^+\{x[n-k]\} = z^{-k}X^+(z) + \sum_{m=1}^{k} x[-m]z^{(m-k)}.
\tag{3.103}
$$

When we shift $x[n]$ to the right (because $k > 0$) to obtain $x[n-k]$, the samples $x[-k], \ldots, x[-1]$ enter the positive time axis and should be included in the computation of the one-sided z-transform. This results in the second term on the right hand side of (3.103); the first term is due to the shifting of the samples of $x[n]$ for $n \geq 0$. This property makes possible the solution of linear constant-coefficient difference equations with *nonzero* initial conditions. Although we use zero-initial conditions in the majority of digital signal processing applications, there are some cases where nonzero initial conditions may appear (see Problem 63). In the next example, we use the one-sided z-transform to determine the zero-input and zero-state responses of the first order system discussed in Section 2.10.

Example 3.17
Let

$$
y[n] = ay[n-1] + bx[n], \quad n \geq 0
\tag{3.104}
$$

with $y[-1] \neq 0$. Taking the one-sided z-transform of (3.104) and using linearity and (3.101), we have

$$
Y^+(z) = ay[-1] + az^{-1}Y^+(z) + bX^+(z).
\tag{3.105}
$$

Solving for $Y^+(z)$ we obtain

$$
Y^+(z) = \underbrace{\frac{ay[-1]}{1 - az^{-1}}}_{\text{initial condition}} + \underbrace{\frac{b}{1 - az^{-1}}X^+(z)}_{\text{zero-state}}.
\tag{3.106}
$$

If the input $x[n] = 0$ for all $n \geq 0$, then the response $y[n]$ is solely due to the initial condition $y[-1]$. Hence the first term on the right hand side of (3.106) can be identified

as an *initial condition* response or the zero-input response $y_{zi}[n]$ discussed in (2.83) and is given by (after taking an inverse z-transform)

$$y_{zi}[n] = ay[-1]\, a^n = y[-1]a^{n+1}, \quad n \geq 0 \tag{3.107}$$

which agrees with (2.83).

On the other hand, if the initial condition is zero in (3.104) then the system is at rest or at zero-state. The first term in (3.106) is now zero, and we have $Y^+(z) = H(z)X^+(z)$ in which the system function is $H(z) = b/(1 - az^{-1})$ or the impulse response is $h[n] = b\, a^n u[n]$, and hence the second term can be identified as the zero-state response $y_{zs}[n]$ in (2.84). The complete response is given by

$$y[n] = y[-1]a^{n+1} + \sum_{k=0}^{n} h[k]x[n - k], \quad n \geq 0 \tag{3.108}$$

which agrees with (2.85). To obtain the transient step response we set $x[n] = u[n]$ in (3.104). Then from (3.106), we have

$$Y^+(z) = \frac{ay[-1]}{1 - az^{-1}} + \frac{b}{(1 - az^{-1})(1 - z^{-1})}$$

$$= \frac{ay[-1]}{1 - az^{-1}} + \frac{\frac{b}{1-a}}{1 - z^{-1}} + \frac{-\frac{ba}{1-a}}{1 - az^{-1}},$$

and hence the complete response is given by

$$y[n] = y[-1]a^{n+1} + \frac{b}{1 - a}\left(1 - a^{n+1}\right), \quad n \geq 0 \tag{3.109}$$

which again agrees with (2.82). ∎

Example 3.17 illustrates the use of a one-sided z-transform in obtaining the output response of a discrete-time system described by LCCDE with nonzero initial conditions. In MATLAB this solution is obtained by using the `filter` function with invocation

$$y = \text{filter}(b,a,x,xic), \tag{3.110}$$

where `xic` is the equivalent initial condition input array obtained from the given initial values of both the input and output signals using

$$xic = \text{filtic}(b,a,Y,X), \tag{3.111}$$

in which `Y` and `X` are the respective initial condition arrays. Thus in Example 3.17, the statements

```
xic = filtic(b,[1,-a],yic,0); % yic = y[-1]
y = filter(b,[1,-a],x,xic);
```

will compute the complete response. This approach is explored in more detail in Tutorial Problem 24.

Learning summary

- Any sequence $x[n]$ can be uniquely characterized by its z-transform: a complex function $X(z)$, of the complex variable z, accompanied by a given ROC.
- The z-transform converts convolution equations and linear constant coefficient difference equations (LCCDEs) into algebraic equations, which are easier to manipulate analytically. Figure 3.12 graphically shows relationships between difference equation, system function, and impulse response.
- In the z-domain, a LTI system is uniquely described by its system function

$$H(z) = \sum_{n=-\infty}^{\infty} h[n]z^{-n} = \frac{Y(z)}{X(z)}.$$

- Systems described by the linear constant coefficient difference equation

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$$

have a rational system function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = b_0 \frac{\prod_{k=1}^{M}(1 - z_k z^{-1})}{\prod_{k=1}^{N}(1 - p_k z^{-1})},$$

with M zeros z_k, $1 \le k \le M$ and N poles p_k, $1 \le k \le N$. The poles of the system determine its stability and the time-domain behavior of its impulse response:

  – If all poles are inside the unit circle, that is, $|p_k| < 1$ for all k, the system is stable. In practice, unstable systems lead to numerical overflow.

  – Real poles contribute exponentially decaying components in the impulse response. The distance of poles from the origin determines the speed of decay.

  – Complex-conjugate poles contribute exponentially decaying sinusoidal components in the impulse response. The distance of poles from the origin determines the decay of the envelop and the angle with the real axis of the frequency of the oscillation.
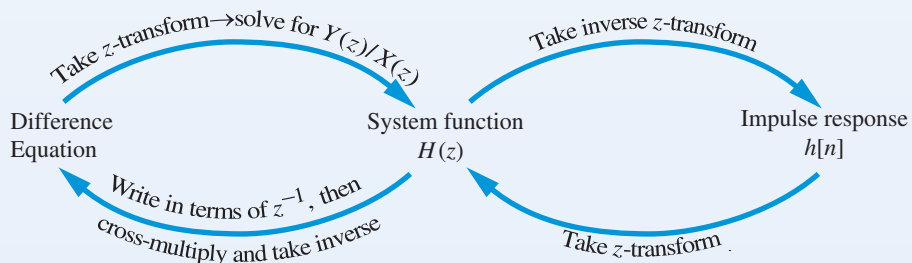

Figure 3.12 System representations and their relationships in graphical form.

- The *z*-transform allows the decomposition of systems with high-order rational system functions into first-order systems with real poles and second-order systems with complex-conjugate poles.
- The two major contributions of *z*-transforms to the study of LTI systems are:
  – The location of the poles determines whether the system is stable or not.
  – We can construct systems, whose impulse response has a desired shape in the time domain, by properly placing poles in the complex plane.
- The major application of one-sided *z*-transforms is in the solution of LCCDEs with nonzero initial conditions. Most DSP applications involve LCCDEs with zero initial conditions.

TERMS AND CONCEPTS

All-pole system An LTI system whose system function has only poles (and trivial zeros at the origin).

All-zero system An LTI system whose system function has only zeros (and trivial poles at the origin).

Anticausal sequence A sequence that is zero for positive *n*, i.e. $n > 0$. Also called a left-sided sequence.

Causal sequence A sequence that is zero for negative *n*, i.e. $n < 0$. Also called a right-sided sequence.

FIR system An LTI system characterized by a finite(-duration) impulse response.

IIR system An LTI system characterized by an infinite(-duration) impulse response.

Impulse response Response of an LTI system to an impulse sequence, denoted by $h[n]$.

Left-sided sequence A sequence that is zero for positive *n*, i.e. $n > 0$. Also called an anti-causal sequence.

Noncausal sequence A sequence that is nonzero for positive as well as negative values of *n*. Also called a two-sided sequence.

Partial fraction expansion (PFE) A decomposition of a higher degree rational function into a sum of first-order rational functions.

Pole of a system function A value of *z* at which the system function has a singularity (or becomes infinite).

Region of convergence (ROC) A set of values of *z* for which the series (3.9) converges. It is always bounded by a circle.

Residue A complex number that describes the behavior of the inverse-*z*-transform of a function around its pole singularity. For a rational function, it is needed in the partial fraction expansion method.

Right-sided sequence A sequence that is zero for negative *n*, i.e. $n < 0$. Also called a causal sequence.

System function The *z*-transform of the impulse response $h[n]$ of an LTI system, denoted by $H(z)$. Also called the transfer function.

Transfer function The *z*-transform of the impulse response $h[n]$ of an LTI system, denoted by $H(z)$. Also called the system function.

Two-sided sequence A sequence that is nonzero for positive as well as for negative values of *n*. Also called a noncausal sequence.

Zero of a system function A value of *z* at which the system function becomes zero.

z-transform (one-sided) A mapping of a positive-time sequence $x[n]$, $n \geq 0$, into a complex-valued function $X(z)$ of a complex variable *z*, given by the series in (3.100).

z-transform (two-sided) A mapping of a sequence $x[n]$ into a complex-valued function $X(z)$ of a complex variable *z*, given by the series in (3.9).

MATLAB functions and scripts

Name	Description	Page
conv	Multiplies two polynomials	104
deconv	Evaluation of long division	99
filter	Determines response of a LCCDE	120
filtic	Determines initial condition array for use in filter	120
poly	Determines the coefficients of a polynomial from its roots	98
polyval	Evaluates the value of a polynomial	98
residuez	Determines the coefficients of partial fraction expansion	102
roots	Computes the roots of a polynomial	98
zplane	Plots the pole-zero pattern of a rational system function	113

FURTHER READING

1. A more detailed discussion of z-transforms, including additional properties and more rigorous derivations, is provided in Proakis and Manolakis (2007) and Oppenheim and Schafer (2010).
2. A complete treatment of partial fraction expansion, for both continuous-time signals and discrete-time signals, is given in Oppenheim *et al.* (1997) and Haykin and Van Veen (2003).
3. Proakis and Manolakis (2007) and Oppenheim *et al.* (1997) cover the one-sided z-transform, which is used to compute the output of difference equations with nonzero initial conditions.
4. A classical introduction to the theory of complex variables and functions is given in Brown and Churchill (2004).

Review questions

1. There exists a sequence that retains its shape when it passes through an LTI system. Do you agree or disagree? Explain.
2. Describe the property that forms the basis for the analysis of LTI systems using the z-transform.
3. Explain the importance of the ROC in the z-transform operation. Why is it always bounded by a circle?
4. Describe ROC shapes of the z-transforms for the causal, anticausal, and noncausal sequences.
5. The ROC for the z-transform of every finite-length sequence is the entire z-plane. Do you agree or disagree? Explain.
6. Can you obtain the z-transform of $u[n]$ at $z = 0$? If you can, what is its value? If you cannot, why not?
7. What is the preferred method for obtaining the inverse z-transform of rational functions? Describe this method.
8. Explain what multiplication of two polynomials can be performed using convolution of their coefficients?

9. Every stable system can always be described by a system function in z. Does an existence of a system function imply that the system is stable? Explain.

10. What are the zeros and poles of a rational function? How are they indicated in the pole-zero plot?

11. If all poles of a system function are inside the unit circle then the system is always stable. Do you agree or disagree? Explain.

12. If the ROC of a system function is the exterior of a circle extending to infinity then that system is causal. True or false? Explain.

13. If a system function has zeros at the origin of the z-plane then the system function is a proper rational function. Why?

14. Describe various classes of LTI system.

15. The impulse response of a causal first-order system is unbounded. Where is the pole of its system function located?

16. The impulse response of a causal second-order system is oscillatory. Where are the poles of its system function located?

17. Two different sequences can have the same one-sided z-transform. Under what condition is this statement true?

18. Explain why a one-sided z-transform is able to determine response to a LCCDE with initial conditions while the two-sided z-transform cannot.

Problems

Tutorial problems

1. Determine the z-transform and sketch the pole-zero plot with the ROC for each of the following sequences:

   (a) $x[n] = \left(\frac{1}{2}\right)^n (u[n] - u[n-10])$,

   (b) $x[n] = \left(\frac{1}{2}\right)^{|n|}$,

   (c) $x[n] = 5^{|n|}$,

   (d) $x[n] = \left(\frac{1}{2}\right)^n \cos(\pi n/3)u[n]$.

2. The `filter` function in MATLAB can be used to verify the z-transform expression of a causal sequence. Let $x[n]$ be a causal sequence with a rational $X(z) \triangleq B(z)/A(z)$ expression.

   (a) Show that the fragment

   ```
   x=filter(b,a,[1,zeros(1,N)]);
   ```

   will generate the first `N+1` samples of $x[n]$ where `b` and `a` contain polynomial coefficients of $B(z)$ and $A(z)$, respectively.

   (b) Let $x[n] = \left[\left(\frac{1}{2}\right)^n + \left(-\frac{1}{3}\right)^n\right]u[n]$. Determine $X(z)$.

   (c) Verify your expression in (b) using MATLAB by comparing output of the `filter` function with the given sequence.

3. Prove the following z-transform pair:

$$x[n] = (r^n \sin \omega_0 n)u[n] \xleftrightarrow{\mathcal{Z}} X(z) = \frac{r(\sin \omega_0)z^{-1}}{1 - 2(r \cos \omega_0)z^{-1} + r^2 z^{-2}}, |z| > r.$$

4. Use the method of partial fraction expansion to determine the sequences corresponding to the following z-transforms:

(a) $X(z) = \dfrac{1 - \frac{1}{3}z^{-1}}{(1 - z^{-1})(1 + 2z^{-1})}$, all possible ROCs.

(b) $X(z) = \dfrac{1 - z^{-1}}{1 - \frac{1}{4}z^{-1}}$, $x[n]$ is causal.

(c) $X(z) = \dfrac{1}{(1 - 0.5z^{-1})(1 - 0.25z^{-1})}$, $x[n]$ is absolutely summable.

5. Determine the inverse z-transform of

$$X(z) = z^2(1 - \tfrac{1}{3}z^{-1})(1 - z^{-1})(1 + 2z^{-2}).$$

6. Given the z-transform pair $x[n] \leftrightarrow X(z) = 1/(1 - 2z^{-1})$ with ROC: $|z| < 2$, use the z-transform properties to determine the z-transform of the following sequences:

(a) $y[n] = x[n - 3]$,

(b) $y[n] = \left(\frac{1}{3}\right)^n x[n]$,

(c) $y[n] = x[n] * x[-n]$,

(d) $y[n] = nx[n]$,

(e) $y[n] = x[n - 1] + x[n + 2]$,

(f) $y[n] = x[n] * x[n - 2]$.

7. Given the z-transform pair $x[n] \leftrightarrow X(z) = 1/(1 - \frac{1}{2}z^{-1})$ with ROC: $|z| > \frac{1}{2}$, use the z-transform properties to determine the sequences corresponding to the following transforms:

(a) $Y(z) = X(1/z)$,

(b) $Y(z) = dX(z)/dz$,

(c) $Y(z) = X^2(z)$.

8. If $X(z)$ is the z-transform of the sequence $x[n] = x_R[n] + jx_I[n]$, prove the following z-transform pairs:

(a) $x^*[n] \xleftrightarrow{\mathcal{Z}} X^*(z^*)$,

(b) $x[-n] \xleftrightarrow{\mathcal{Z}} X(1/z)$,

(c) $x_R[n] \xleftrightarrow{\mathcal{Z}} \frac{1}{2}[X(z) + X^*(z^*)]$,

(d) $x_I[n] \xleftrightarrow{\mathcal{Z}} \frac{1}{2j}[X(z) - X^*(z^*)]$.

9. The z-transform $X(z)$ of a causal sequence $x[n]$ has a zero at $z_1 = 0$ and three poles at $p_1 = -3/4$ and $p_{2,3} = (1/2)(1 \pm j)$. Determine the z-transform $Y(z)$ of the sequence $y[n] = x[-n + 3]$, its pole-zero pattern, and its ROC.

10. Compute $y[n] = h[n] * x[n]$ for $h[n] = a^n u[n]$ and $x[n] = u[-n - 1]$.

11. Determine the convolution $y[n] = h[n] * x[n]$ in the following Cases:

(a) $h[n] = a^n u[n]$ and $x[n] = b^n u[n]$, $a \neq b$.

(b) $h[n] = a^n u[n]$ and $x[n] = b^n u[n]$, $a = b$.

(c) $h[n] = a^n u[n]$ and $x[n] = a^{-n} u[-n]$, $0 < a < 1$.

12. A function called *autocorrelation* for a real-valued, absolutely summable sequence $x[n]$, is defined as

$$r_{xx}[\ell] \triangleq \sum_n x[n]x[n - \ell]. \tag{3.112}$$

Let $X(z)$ be the z-transform of $x[n]$ with ROC $\alpha < |z| < \beta$.

(a) Show that the z-transform of $r_{xx}[\ell]$ is given by

$$R_{xx}(z) = X(z)X(z^{-1}). \tag{3.113}$$

What is the ROC of $R_{xx}(z)$?

(b) Let $x[n] = a^n u[n]$, $|a| < 1$. Determine $R_{xx}(z)$ and sketch its pole-zero plot and the ROC.

(c) Determine the autocorrelation $r_{xx}[\ell]$ for the $x[n]$ in (b) above.

13. Determine the impulse response of the system described by

$$y[n] - \frac{5}{2}y[n - 1] + y[n - 2] = x[n - 1]$$

for all possible regions of convergence.

14. Given a causal system described by $y[n] = \frac{1}{2}y[n - 1] + x[n]$, compute its response to the following inputs:

(a) $x[n] = e^{j(\pi/4)n}$, $-\infty < n < \infty$

(b) $x[n] = e^{j(\pi/4)n}u[n]$,

(c) $x[n] = (-1)^n$, $-\infty < n < \infty$

(d) $x[n] = (-1)^n u[n]$.

15. Consider a LTI described by the following input-output relation:

$$y[n] = \frac{3}{4}y[n - 1] - \frac{1}{8}y[n - 2] + x[n].$$

(a) Find the system function $H(z)$ and check whether the system is stable.

(b) Determine the impulse response $h[n]$ of the system.

(c) Determine the step response $s[n]$ of the system.

(d) Compute $h[n]$ and $s[n]$ for $0 \le n \le 10$ using the formulas in (b) and (c) and compare with the values obtained using the function `filter`.

16. The response of a LTI system to the input $x[n] = u[n]$ is $y[n] = 2(1/3)^n u[n]$.

(a) Find the impulse response $h[n]$ of the system.

(b) Find the output $y[n]$ for the input $x[n] = (1/2)^n u[n]$.

(c) Check the results in (a) and (b) using the function `filter`.

17. Consider the cases $b_0 = 0$ or $b_1 = 0$ in formulas (3.97)–(3.99) and compare the results with the last two entries in Table 3.1.

18. Find the impulse response of the system (3.97) for the case of real and equal poles and use the result to determine how the location of the poles affects (a) the stability of the system, and (b) the shape of the impulse response. Hint: Use MATLAB to replicate Figure 3.10 for a double pole.

19. Consider a causal LTI system described by the difference equation

$$y[n] = \frac{1}{2}y[n-1] + x[n] - \frac{1}{1024}x[n-10].$$

(a) Determine the system function $H(z)$ and plot the pole-zero pattern using the function `zplane`.

(b) Compute and plot the impulse response of the system using `impz`.

(c) Explain the length of $h[n]$ using the pole-zero pattern plot.

(d) Find an equivalent difference equation for the description of the system.

20. Consider a causal system with input $x[n]$ and output $y[n]$. If the input is given by

$$x[n] = -(1/3)(1/2)^n u[n] - (4/3)2^n u[-n-1],$$

the output has a z-transform given by

$$Y(z) = \frac{1 - z^{-2}}{(1 - \frac{1}{2}z^{-1})(1 - 2z^{-1})}.$$

(a) Determine the z-transform of the input $x[n]$.

(b) Find all possible choices for the impulse response of the system.

21. Consider the causal and stable system given in Example 3.16.

(a) Plot the pole-zero pattern using the function `zplane(a,b)`.

(b) Compute and plot the impulse response using the functions `filter` and `stem`. Compare with the plot obtained using the function `impz`.

(c) Use the function `residuez` and the z-transform pairs in Table 3.1 to find an analytical expression for the impulse response $h[n]$.

(d) Compute the first ten samples of $h[n]$ using the formula obtained in Part (c) and compare with the values obtained from the difference equation.

22. Repeat Problem 21 for a causal system defined by the difference equation

$$y[n] = -\frac{1}{4}y[n-1] + \frac{1}{8}y(n-2) + x[n] + x[n-1].$$

23. Write a MATLAB script to generate the plots shown in Figure 3.11 for $\omega_0 = \pi/3$ and $r = 0.8, 1, 1.25$. Run the script by changing the values of ω_0 and r to appreciate their effect on the form of $h[n]$.

24. The linear constant coefficient difference equation

$$y[n] = \frac{1}{3}\{x[n] + x[n-1] + x[n-2]\} + 0.95y[n-1] - 0.9025y[n-2]$$

is excited by the input $x[n] = \cos(\pi n/3)u[n]$ subject to the initial conditions:

$$y[-1] = -2, \quad y[-2] = -3, \quad x[-1] = 1, \quad x[-2] = 1.$$

(a) Determine analytically the complete response $y[n]$.

(b) Verify your answer using MATLAB.

Basic problems

25. Determine the z-transform and sketch the pole-zero plot with the ROC for each of the following sequences:

(a) $x[n] = (1/2)^n u[n] + (1/3)^n u[n]$,

(b) $x[n] = (1/2)^n u[n] + (1/3)^n u[-n-1]$,

(c) $x[n] = (1/3)^n u[n] + (1/2)^n u[-n-1]$.

26. Show that the z-transform of the two-sided sequence $x[n] = a^{|n|}$ is given by

$$X(z) = \frac{(a - a^{-1})z^{-1}}{(1 - az^{-1})(1 - a^{-1}z^{-1})}$$

with ROC: $|a| < |z| < 1/|a|$. Also, explain why the z-transform does not exist if $|a| > 1$. Hint: Follow the steps in Example 3.6.

27. Let $x[n] = 0.8^n u[n]$ and let

$$y[n] = \begin{cases} x[n/2], & n = 0, \pm 2, \pm 4, \dots \\ 0. & \text{otherwise} \end{cases}$$

(a) Show that $Y(z) = X(z^2)$.

(b) Determine $Y(z)$.

(c) Using MATLAB verify that $y[n]$ has the z-transform $Y(z)$. (Hint: See Problem 2.)

28. Use the method of partial fraction expansion to determine the sequences corresponding to the following z-transforms:

(a) $X(z) = \dfrac{z^3 - 3z^2 + 4z + 1}{z^3 - 4z^2 + z - 0.16}$. All possible ROCs.

(b) $X(z) = z/(z^3 + 2z^2 + \frac{5}{4}z + \frac{1}{4})$, $|z| > 1$.

(c) $X(z) = z/(z^2 - \frac{1}{3})^2$, $|z| < 0.5$.

29. The z-transform of a signal $x[n]$ is given by

$$X(z) = \frac{2z^2 + 3z}{z^2 - z + 0.81}. \quad |z| > 0.9$$

(a) Express $x[n]$ as a real-valued signal.

(b) Using MATLAB, determine the first 30 samples of $x[n]$ and compare them with your answer in (a). (See Problem 2.)

30. Given the z-transform pair $x[n] \leftrightarrow X(z) = 1/(1 - \frac{1}{3}z^{-1})$ with ROC: $|z| > \frac{1}{3}$, use the z-transform properties to determine the z-transform of the following sequences:

(a) $y[n] = x[n-2]$,

(b) $y[n] = 2^n x[n]$,

(c) $y[n] = x[n] * x[-n-1]$,

(d) $y[n] = n^2 x[n]$,

(e) $y[n] = 2x[n+1] + 3x[n-3]$,

(f) $y[n] = x[n-1] * x[n]$.

31. Given the z-transform pair $x[n] \leftrightarrow X(z) = 1/(1 - (0.8)z^{-1})$ with ROC: $|z| > 0.8$, use the z-transform properties to determine the sequences corresponding to the following transforms:
   (a) $Y(z) = X(1/z)$,
   (b) $Y(z) = dX(z)/dz$,
   (c) $Y(z) = X^2(z)$.

32. Determine the z-transform and the ROC of the sequence

$$y[n] = \sum_{k=-n}^{n} a^{|k|} u[n]. \quad |a| < 1$$

33. The z-transform $X(z)$ of a stable sequence $x[n]$ has two zeros at $z_{1,2} = \pm j$ and three poles at $p_{1,2} = \pm 0.8$ and $p_{3,4} = \pm j0.8$. Furthermore, $X(1) = 1$. Determine the z-transform $Y(z)$ of the sequence $y[n] = x[n-2]$, its pole-zero pattern, and its ROC.

34. Compute $y[n] = h[n] * x[n]$ for $h[n] = (1/2)^n u[n]$ and $x[n] = 3^n u[-n]$.

35. Determine the convolution $y[n] = h[n] * x[n]$ in the following cases:
   (a) $h[n] = (0.8)^n u[n]$ and $x[n] = (1.2)^n u[-n]$,
   (b) $h[n] = 2^{-n} u[n] + 3^n u[-n - 1]$ and $x[n] = (0.75)^n u[n]$,
   (c) $h[n] = (0.8)^n u[n] - (1.2)^{-n-1}$ and $x[n] = (0.9)^n u[n] + (1.5)^n u[-n]$.

36. The autocorrelation of a complex-valued, absolutely summable sequence $x[n]$, is defined as

$$r_{xx}[\ell] \triangleq \sum_{n} x[n] x^*[n - \ell]. \tag{3.114}$$

Let $X(z)$ be the z-transform of $x[n]$ with ROC $\alpha < |z| < \beta$.
   (a) Show that the z-transform of $r_{xx}[\ell]$ is given by

$$R_{xx}(z) = X(z) X^* \left(1/z^*\right). \tag{3.115}$$

What is the ROC of $R_{xx}(z)$?
   (b) Let $x[n] = \left(re^{j\theta}\right)^n u[n]$, $0 < r < 1$. Determine $R_{xx}(z)$ and sketch its pole-zero plot and the ROC.
   (c) Determine the autocorrelation $r_{xx}[\ell]$ for the $x[n]$ in (b) above.

37. Determine the impulse response of the system described by $y[n] + 0.2y[n - 1] - 0.18y[n - 2] + 0.891y[n - 3] = x[n - 1] + x[n - 2]$ for all possible regions of convergence.

38. Given a causal system described by $y[n] = 0.8y[n-1] - 0.81y[n-2] + x[n-1] + x[n-2]$, compute its response to the following inputs:
   (a) $x[n] = e^{j(\pi/3)n}$, $\quad -\infty < n < \infty$
   (b) $x[n] = e^{j(\pi/3)n} u[n]$,
   (c) $x[n] = 1$, $\quad -\infty < n < \infty$
   i. $x[n] = (-1)^n u[n]$.

39. The step response of a LTI system is given by $y[n] = (1/2)^{n-1} u[n + 1]$. Find the impulse response $h[n]$ and determine whether the system is causal and stable.

40. The response of a LTI system to the input $x[n] = \left(\frac{1}{4}\right)^n u[n]$ is $y[n] = 5\left(\frac{3}{4}\right)^n u[n]$.
    (a) Find the impulse response $h[n]$ of the system.
    (b) Find the output $y[n]$ for the input $x[n] = \left(\frac{1}{3}\right)^n u[n]$.
    (c) Check the results in (a) and (b) using the function `filter`.

41. Consider a causal system with input $x[n]$ and output $y[n]$. Determine its impulse response $h[n]$ if we are given that:
    (a) $x[n] = (1/2)^n u[n] + 2^n u[-n-1]$ and $y[n] = 6(1/2)^n u[n] - 6(3/4)^n u[n]$.
    (b) $x[n] = (-3)^n u[n]$ and $y[n] = 4(2)^n u[n] - (1/2)^n u[n]$.

42. Consider a causal system with input $x[n]$ and output $y[n]$. If the input is given by

$$x[n] = 3^{1-n} u[n] - 4^{n+1} u[-n-1],$$

    the output has a z-transform given by

$$Y(z) = \frac{1 + z^{-1} - z^{-2}}{(1 - \frac{1}{3}z^{-1})(1 - 4z^{-1})}.$$

    (a) Determine the z-transform of the input $x[n]$.
    (b) Find all possible choices for the impulse response of the system.

43. A difference equation is given by

$$y[n] = x[n] - x[n-1] + 0.81y[n-2], \quad n \geq 0$$

    with initial conditions $y[-1] = y[-2] = 2$ and excited by $x[n] = (0.7)^n u[n+1]$.
    (a) Determine the solution $y[n]$, $n \geq 0$.
    (b) Generate the first 50 samples of $y[n]$ using MATLAB and compare these samples with those in (a) above.

44. Determine zero-input, zero-state, transient, and steady-state responses of the system

$$y[n] = \frac{1}{4}y[n-1] + x[n] + 3x[n-1], \quad n \geq 0$$

    to the input $x[n] = e^{j\pi n/4}u[n]$ with $y[-1] = 2$.

Assessment problems

45. Determine the z-transform and sketch the pole-zero plot with the ROC for each of the following sequences:
    (a) $x[n] = 2^n u[n] + 3(1/2)^n u[n]$,
    (b) $x[n] = (1/2)^n u[n+1] + 3^n u[-n-1]$,
    (c) $x[n] = (1/3)^n \sin(\pi n/4)u[n]$,
    (d) $x[n] = |n|(1/2)^{|n|}$.

46. Use the method of partial fraction expansion to determine the sequences corresponding to the following z-transforms:
    (a) $X(z) = \dfrac{1 - z^{-1} - 4z^{-2} + 4z^{-3}}{1 - \frac{11}{4}z^{-1} + \frac{13}{8}z^{-2} - \frac{1}{4}z^{-3}}$. The sequence is causal.
    (b) $X(z) = \dfrac{z^3 - 3z^2 + 4z + 1}{z^3 - 4z^2 + z - 0.16}$. The sequence is left-sided.

(c) $X(z) = z/\left(z^3 + 2z^2 + 1.25z + 0.25\right), |z| > 1.$

(d) $X(z) = z/\left(z^2 - 0.25\right)^2, |z| > 0.5.$

47. Given the z-transform pair $x[n] \leftrightarrow X(z) = z^{-1}/(1 + 0.8z^{-1})$ with ROC: $|z| > 0.8$, use the z-transform properties to determine the z-transform of the following sequences:

(a) $y[n] = x[n + 2],$

(b) $y[n] = x[3 - n],$

(c) $y[n] = \left(\frac{5}{4}\right)^n x[n],$

(d) $y[n] = (n + 1)x[n - 1],$

(e) $y[n] = x[n] * x[2 - n],$

(f) $y[n] = x[n + 2] + x[3 - n].$

48. Consider the finite length sequence $x[n] = u[n] - u[n - N].$

(a) Determine the z-transform $X(z)$ of the sequence $x[n].$

(b) Determine and plot the sequence $y[n] = x[n] * x[n].$

(c) Determine the z-transform $Y(z)$ of the sequence $y[n].$

49. Show the following properties of the z-transform:

(a) If $x[n] = x[-n]$ (even), then $X(z^{-1}) = X(z).$

(b) If $x[n] = -x[-n]$ (odd), then
$X(z^{-1}) = -X(z).$

(c) In case (b) there is a zero in $X(z)$ at $z = 1.$

50. Let $x_3[n] = x_1[n] * x_2[n].$ Show that

$$\sum_{n=-\infty}^{\infty} x_3[n] = \left(\sum_{n=-\infty}^{\infty} x_1[n]\right)\left(\sum_{n=-\infty}^{\infty} x_2[n]\right). \qquad (3.116)$$

51. The z-transform $X(z)$ of a causal sequence $x[n]$ has a zero at $z_1 = -1$ and three poles at $p_1 = \frac{4}{5}$ and $p_{2,3} = \frac{1}{3}(-1 \pm j).$ Determine the z-transform $Y(z)$ of the sequence $y[n] = x[-n + 2],$ its pole-zero pattern, and its ROC.

52. Compute $y[n] = h[n] * x[n]$ for $h[n] = n(0.8)^n u[n]$ and $x[n] = 2^n u[-n].$

53. Determine the convolution $y[n] = h[n] * x[n]$ in the following cases:

(a) $h[n] = (0.5)^n u[n], x[n] = 2^n u[n].$

(b) $h[n] = 3^{-n} u[n] + 3^n u[-n - 1], x[n] = 2^{-n} u[n].$

(c) $h[n] = (0.5)^n u[n] - 2^n u[-n - 1], x[n] = (0.25)^n u[n] - 4^n u[-n - 1].$

54. Consider the autocorrelation function given in (3.112) for a real-valued, absolutely summable sequence. Let $x[n] = b^n u[-n - 1], |b| > 1.$

(a) Determine $R_{xx}(z).$

(b) Sketch its pole-zero plot and the ROC.

(c) Determine the autocorrelation $r_{xx}[\ell].$

55. Consider the autocorrelation function given in (3.112) for a real-valued, absolutely summable sequence. Let $x[n] = \left(\frac{1}{2}\right)^n u[n] + 3^n u[-n - 1].$

(a) Determine $R_{xx}(z).$

(b) Sketch its pole-zero plot and the ROC.

(c) Determine the autocorrelation $r_{xx}[\ell].$

56. Consider the autocorrelation function given in (3.114) for a complex-valued, absolutely summable sequence. Let $x[n] = \left(0.9e^{j\pi/3}\right)^n u[n].$

(a) Determine $R_{xx}(z)$.

(b) Sketch its pole-zero plot and the ROC.

(c) Determine the autocorrelation $r_{xx}[\ell]$.

57. Determine the impulse response of the system described by

$$y[n] + \frac{11}{6}y[n-1] + \frac{1}{2}y[n-2] = 2x[n]$$

for all possible regions of convergence.

58. Given a stable system described by $y[n] = x[n] - \frac{4}{5}y[n-1]$, compute its response to the following inputs:

(a) $x[n] = (1+j)^n, \quad -\infty < n < \infty$

(b) $x[n] = \cos(\pi n/4)u[n]$,

(c) $x[n] = (-1)^n, \quad -\infty < n < \infty$

(d) $x[n] = (-1)^n u[n]$.

In each case, identify the transient and steady-state responses.

59. The response of a LTI system to the input $x[n] = \left(\frac{1}{2}\right)^n u[n] + 2^n u[-n-1]$ is $y[n] = 3(0.7)^n u[n]$.

(a) Find the impulse response $h[n]$ of the system.

(b) Find the output $y[n]$ for the input $x[n] = (0.9)^n u[-n]$.

60. Consider a stable system with input $x[n]$ and output $y[n]$. Determine its impulse response $h[n]$ if we are given that:

(a) $x[n] = \left(\frac{1}{3}\right)^{|n|}$ and $y[n] = 2(1/3)^n u[n] - 2^{n+2} u[-n]$.

(b) $x[n] = \left(\frac{3}{4}\right)^n u[n]$ and $y[n] = (0.75)^n u[n] - (4)^n u[-n-1]$.

61. Consider a causal system with input $x[n]$ and output $y[n]$. If the input is given by

$$x[n] = \left(\frac{2}{3}\right) 3^n u[-n-1] - \left(\frac{1}{2}\right)\left(\frac{1}{3}\right)^n u[n],$$

the output has a z-transform given by

$$Y(z) = \frac{-1}{(1 - \frac{1}{2}z^{-1})(1 - 2z^{-1})}.$$

(a) Determine the z-transform of the input $x[n]$.

(b) Find all possible choices for the impulse response of the system.

62. A stable system has the following pole-zero locations:

$$z_{1,2} = \pm j, \quad p_{1,2} = \frac{-1 \pm j}{2}.$$

It is also known that $H(1) = 0.8$.

(a) Determine $H(z)$ and obtain its ROC.

(b) Determine the difference equation representation.

(c) Determine the transient and steady-state responses if the input is

$$x[n] = \frac{1}{\sqrt{2}} \sin\left(\frac{\pi n}{2}\right) u[n].$$

63. Consider the following LCCDE:

$$y[n] = 2\cos(\omega_0)y[n-1] - y[n-2],$$

with no input but with initial conditions $y[-1] = 0$ and $y[-2] = -A\sin(\omega_0)$.

(a) Show that the solution of the above LCCDE is given by the sequence $y[n] = A\sin[(n+1)\omega_0]u[n]$. This system is known as a *digital oscillator*.

(b) For $A = 2$ and $\omega_0 = 0.1\pi$, verify the operation of the above digital oscillator using MATLAB.

Review problems

64. A causal system is described by $H(z) = \frac{1-2z^{-1}}{1-\frac{1}{2}z^{-1}}$. When an input $x[n]$ is applied, the output of the system is $y[n] = 90.90^n u[n]$.

(a) Determine at least two possible inputs $x[n]$ that could produce the given $y[n]$.

(b) What is the input $x[n]$ if it is known that the input is absolutely summable.

(c) What is the input $x[n]$ if it is known that a stable system exists for which $x[n]$ is the output if $y[n]$ is the input? Determine the impulse response $h[n]$ of this system.

65. For LTI systems described by the system functions below, determine their (i) impulse response, (ii) difference equation, (iii) pole-zero plot, and (iv) output $y[n]$ if the input $x[n] = \left(\frac{1}{4}\right)^n u[n]$.

(a) $H(z) = (z+1)/(z-0.5)$, causal system.

(b) $H(z) = \dfrac{1 + z^{-1} + z^{-2}}{1 + 0.5z^{-1} - 0.25z^{-2}}$, stable system.

(c) $H(z) = (z^2 - 1)/(z - 3)^2$, anticausal system.

(d) $H(z) = (1 + z^{-1} + z^{-2})^2$, causal system.

4 Fourier representation of signals

In this chapter we introduce the concept of Fourier or frequency-domain representation of signals. The basic idea is that any signal can be described as a sum or integral of sinusoidal signals. However, the exact form of the representation depends on whether the signal is continuous-time or discrete-time and whether it is periodic or aperiodic. The underlying mathematical framework is provided by the theory of Fourier series, introduced by Jean Baptiste Joseph Fourier (1768–1830).

The major justification for the frequency domain approach is that LTI systems have a simple behavior with sinusoidal inputs: the response of a LTI system to a sinusoid is a sinusoid with the same frequency but different amplitude and phase.

Study objectives

After studying this chapter you should be able to:

- Understand the fundamental differences between continuous-time and discrete-time sinusoidal signals.
- Evaluate analytically the Fourier representation of continuous-time signals using the Fourier series (periodic signals) and the Fourier transform (aperiodic signals).
- Evaluate analytically and numerically the Fourier representation of discrete-time signals using the Fourier series (periodic signals) and the Fourier transform (aperiodic signals).
- Choose the proper mathematical formulas to determine the Fourier representation of any signal based on whether the signal is continuous-time or discrete-time and whether it is periodic or aperiodic.
- Understand the use and implications of the various properties of the discrete-time Fourier transform.

4.1   Sinusoidal signals and their properties

The goal of Fourier analysis of signals is to break up all signals into summations of sinusoidal components. Thus, we start our discussion with the definitions and properties of continuous-time and discrete-time sinusoidal signals. Fourier analysis is like a glass prism, which splits a beam of light into frequency components corresponding to different colors.

4.1.1   Continuous-time sinusoids

A continuous-time sinusoidal signal may be represented as a function of time t by the equation

$$x(t) = A\cos(2\pi F_0 t + \theta), \quad -\infty < t < \infty \tag{4.1}$$

where A is the amplitude, θ is the phase in radians, and F_0 is the frequency. If we assume that t is measured in seconds, the units of F_0 are cycles per second or Hertz (Hz). In analytical manipulations it is more convenient to use the *angular* or *radian* frequency

$$\Omega_0 = 2\pi F_0 \tag{4.2}$$

measured in radians per second. The meaning of these quantities is illustrated in Figure 4.1.

Using Euler's identity, $e^{\pm j\phi} = \cos\phi \pm j\sin\phi$, we can express every sinusoidal signal in terms of two complex exponentials with the same frequency, as follows:

$$A\cos(\Omega_0 t + \theta) = \frac{A}{2}e^{j\theta}e^{j\Omega_0 t} + \frac{A}{2}e^{-j\theta}e^{-j\Omega_0 t}. \tag{4.3}$$

Therefore, we can study the properties of the sinusoidal signal (4.1) by studying the properties of the complex exponential $x(t) = e^{j\Omega_0 t}$.

Frequency, viewed as the number of cycles completed per unit of time, is an inherently positive quantity. However, the use of negative frequencies is a convenient way to describe signals in terms of complex exponentials. The concept of negative frequencies is used throughout this book, mainly for mathematical convenience.
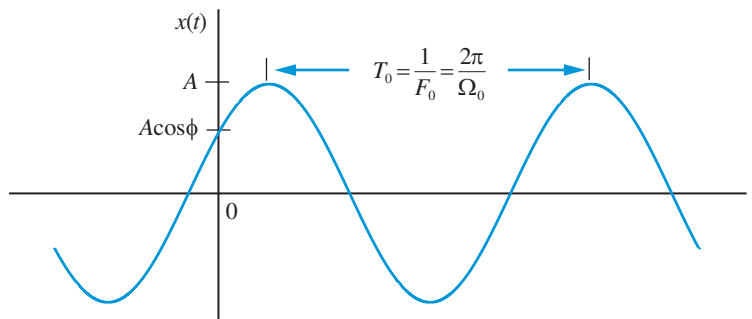

Figure 4.1 Continuous-time sinusoidal signal and its parameters.

To understand the importance of complex exponentials in the study of LTI systems, we determine the response $y(t)$ of the system to the input $x(t) = e^{j\Omega t}$ using the convolution integral. The result is

$$y(t) = \int_{-\infty}^{\infty} h(\tau) x(t - \tau) \, d\tau = \int_{-\infty}^{\infty} h(\tau) e^{j\Omega(t-\tau)} \, d\tau$$

$$= \int_{-\infty}^{\infty} h(\tau) e^{j\Omega t} e^{-j\Omega \tau} \, d\tau = \left(\int_{-\infty}^{\infty} h(\tau) e^{-j\Omega \tau} \, d\tau \right) e^{j\Omega t}. \tag{4.4}$$

When the integral in the parenthesis of (4.4) exists, its value is a complex constant, say $H(j\Omega)$, whose value depends on the input frequency. Thus, the response to $e^{j\Omega t}$ is of the form

$$y(t) = H(j\Omega) e^{j\Omega t}. \quad -\infty < t < \infty \tag{4.5}$$

This implies that the complex exponentials are eigenfunctions of continuous-time LTI systems. For a specific value of Ω, the constant $H(j\Omega)$ is an eigenvalue associated with the eigenfunction $e^{j\Omega t}$ (see discussion in Section 3.1). Choosing $h(t)$ so that $H(j\Omega) \approx 1$ over some range of frequencies and $H(j\Omega) \approx 0$ over another range of frequencies, provides the basis for the design of frequency-selective filters (see Chapter 5).

The continuous-time sinusoid (4.1) is characterized by the following important properties:

1. A continuous-time sinusoid is periodic, with fundamental period $T_0 = 1/F_0$, for every value of the frequency F_0. Indeed, since $e^{j2\pi} = 1$, we have
   $e^{j2\pi F_0(t+T_0)} = e^{j2\pi F_0 t} e^{j2\pi F_0 T_0} = e^{j2\pi F_0 t}$.
2. Two sinusoids with different frequencies are different. That is, if $F_1 \neq F_2$ then $x_1(t) = e^{j2\pi F_1 t} \neq x_2(t) = e^{j2\pi F_2 t}$. Furthermore, as illustrated in Figure 4.2, $T_1 > T_2$ implies that $F_1 < F_2$, and vice versa.
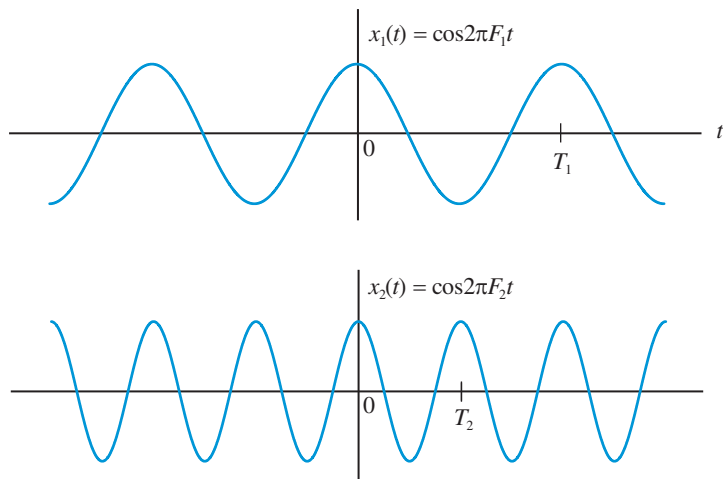

Figure 4.2 For continuous-time sinusoids, $F_1 < F_2$ always implies that $T_1 > T_2$.

3. The rate of oscillation (that is, the number of cycles completed in one second) of a continuous-time sinusoid increases indefinitely with increasing frequency. Indeed, since t is a continuous variable, $F_0 = 1/T_0 \to \infty$ as $T_0 \to 0$.

A set of *harmonically related* complex exponential signals, with fundamental frequency $\Omega_0 = 2\pi/T_0 = 2\pi F_0$, is defined by

$$s_k(t) = e^{jk\Omega_0 t} = e^{j2\pi kF_0 t}, \quad k = 0, \pm 1, \pm 2, \ldots \tag{4.6}$$

We say that $s_1(t)$ is the *fundamental harmonic* of the set and $s_k(t)$ is the kth *harmonic* of the set. Clearly all harmonics $s_k(t)$ are periodic with period T_0. Furthermore, if $k_1 \neq k_2$, then $s_{k_1}(t) \neq s_{k_2}(t)$. A very important characteristic of harmonically related complex exponentials is the following *orthogonality property* (see Tutorial Problem 6):

$$\int_{T_0} s_k(t) s_m^*(t) \, dt = \int_{T_0} e^{jk\Omega_0 t} e^{-jm\Omega_0 t} \, dt = \begin{cases} T_0, & k = m \\ 0, & k \neq m \end{cases} \tag{4.7}$$

where by $\int_{T_0}$ we denote integration over any interval of length T_0, that is, from t_0 to $t_0 + T_0$ for any choice of t_0. The choice of t_0 is usually a matter of convenience.

Figure 4.3(a) shows a periodic signal composed of three sinusoids with harmonically related frequencies

$$x_1(t) = \frac{1}{3} \cos(2\pi F_0 t) - \frac{1}{10} \cos(2\pi 3F_0 t) + \frac{1}{20} \cos(2\pi 5F_0 t), \tag{4.8}$$

where $F_0 = 10$ Hz. The fundamental period is given by $T_0 = 1/F_0 = 0.1$ s. Suppose now that the frequencies of the three sinusoids are *not* harmonically related. For example

$$x_2(t) = \frac{1}{3} \cos(2\pi F_0 t) - \frac{1}{10} \cos\left(2\pi\sqrt{8}F_0 t\right) + \frac{1}{20} \cos\left(2\pi\sqrt{51}F_0 t\right). \tag{4.9}$$

(a)

(b)
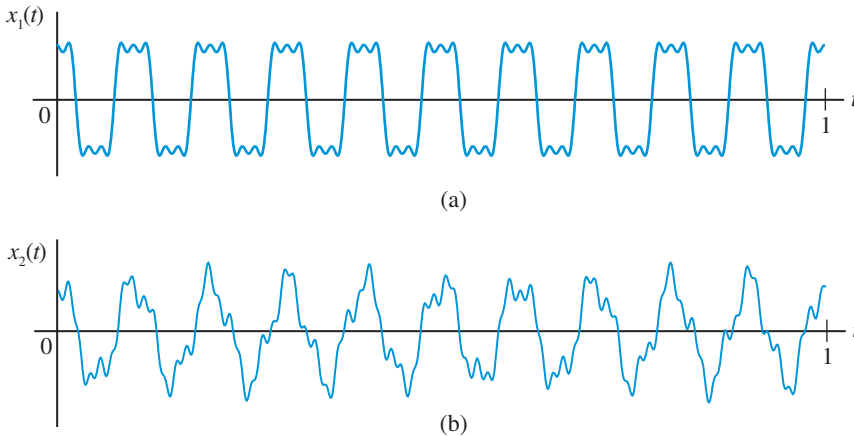
Figure 4.3 Examples of (a) a periodic signal $x_1(t)$, and (b) an "almost"-periodic signal $x_2(t)$.
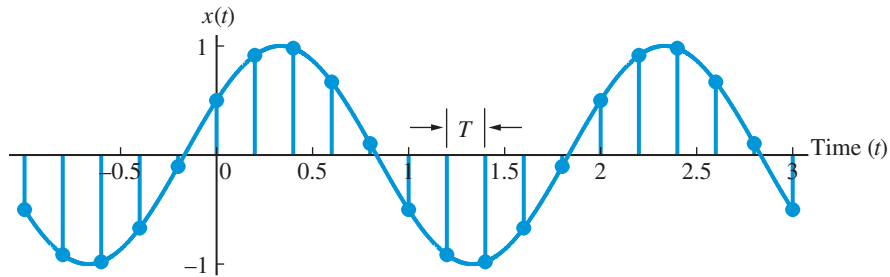
Figure 4.4 Sampling of a continuous-time sinusoidal signal.

Although each sinusoidal signal on the right-hand side of (4.9) is periodic, there is no period T_0 in which $x_2(t)$ repeats itself. The signal $x_2(t)$, which is shown in Figure 4.3(b), is said to be "almost"-periodic or "quasi"-periodic. It turns out that we can create aperiodic finite duration signals ("pulse-like") by combining sinusoidal components with frequencies within a continuous frequency range through integration (see Section 4.2.2).

4.1.2   Discrete-time sinusoids

A discrete-time sinusoidal signal is conveniently obtained by sampling the continuous-time sinusoid (4.1) at equally spaced points $t = nT$ as shown in Figure 4.4. The resulting sequence is

$$x[n] = x(nT) = A\cos(2\pi F_0 nT + \theta) = A\cos\left(2\pi \frac{F_0}{F_s} n + \theta\right). \tag{4.10}$$

If we define the *normalized frequency* variable

$$f \triangleq \frac{F}{F_s} = FT, \tag{4.11}$$

and the *normalized angular frequency* variable

$$\omega \triangleq 2\pi f = 2\pi \frac{F}{F_s} = \Omega T, \tag{4.12}$$

we can express the discrete-time sinusoid (4.10) as

$$x[n] = A\cos(2\pi f_0 n + \theta) = A\cos(\omega_0 n + \theta), \quad -\infty < n < \infty \tag{4.13}$$

where A is the amplitude, f_0 (or ω_0) the frequency, and θ the phase (see Figure 4.5).

If the input to a discrete-time LTI system is a complex exponential sequence, its output is a complex exponential with the same frequency. Indeed, we have

$$x[n] = e^{j\omega n} \overset{\mathcal{H}}{\longmapsto} y[n] = H(e^{j\omega})e^{j\omega n}, \quad \text{for all } n, \tag{4.14}$$

which is obtained from (3.6) by setting $z = e^{j\omega}$. Thus, the complex exponentials $e^{j\omega n}$ are eigenfunctions of discrete-time LTI systems with eigenvalues given by the system function $H(z)$ evaluated at $z = e^{j\omega}$. As we will see in the next chapter, this property is of major importance in signal and system analysis.
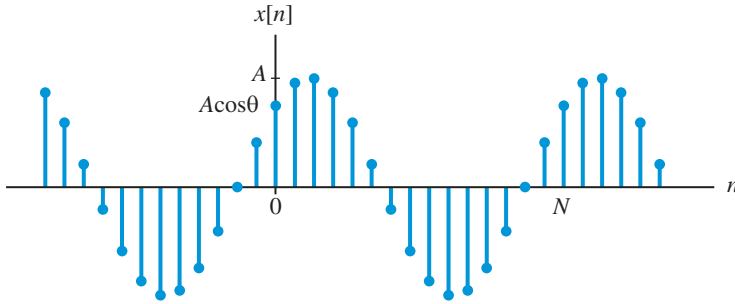
Figure 4.5 Discrete-time sinusoidal signal.

The fact that n is a discrete variable whereas t is a continuous variable leads to some important differences between discrete-time and continuous-time sinusoidal signals.

Periodicity in time By definition $x[n]$ is periodic if $x[n + N] = x[n]$ for all n. For the sequence (4.13) to be periodic, we should have

$$x[n + N] = A \cos(2\pi f_0 n + 2\pi f_0 N + \theta) = A \cos(2\pi f_0 n + \theta) = x[n]. \tag{4.15}$$

This is possible if and only if $2\pi f_0 N = 2\pi k$, where k is an integer. Hence:

Result 4.1.1 The sequence $x[n] = A \cos(2\pi f_0 n + \theta)$ is periodic if and only if $f_0 = k/N$, that is, f_0 is a rational number. If k and N are a pair of prime numbers, then N is the fundamental period of $x[n]$.

To understand the physical meaning of this property, suppose that we sample a continuous-time sinusoid every T seconds. The relative frequency is

$$f_0 = \frac{F_0}{F_s} = \frac{k}{N} = \frac{1/T_0}{1/T} = \frac{T}{T_0}, \tag{4.16}$$

which implies that $NT = kT_0$. Thus, a discrete-time sinusoid, obtained by sampling, is periodic if its period in seconds, NT, is equal to an integer number of periods, kT_0, of the corresponding continuous-time sinusoid.

Periodicity in frequency From the definition (4.13), we can easily see that

$$A \cos[(\omega_0 + k2\pi)n + \theta] = A \cos(\omega_0 n + kn2\pi + \theta)$$
$$= A \cos(\omega_0 n + \theta),$$

because $(kn)2\pi$ is always an integer multiple of 2π. Therefore, we have the result:

Result 4.1.2 The sequence $x[n] = A \cos(\omega_0 n + \theta)$ is periodic in ω_0 with fundamental period 2π and periodic in f_0 with fundamental period one.

This property has a number of very important implications:

1. Sinusoidal sequences with radian frequencies separated by integer multiples of 2π are identical.
2. All distinct sinusoidal sequences have frequencies within an interval of 2π radians. We shall use the so-called *fundamental* frequency ranges

$$-\pi < \omega \le \pi \quad \text{or} \quad 0 \le \omega < 2\pi. \tag{4.17}$$

Therefore, if $0 \le \omega_0 < 2\pi$, the frequencies ω_0 and $\omega_0 + m2\pi$ are indistinguishable from observation of the corresponding sequences.

3. Since $A\cos[\omega_0(n + n_0) + \theta] = A\cos[\omega_0 n + (\omega_0 n_0 + \theta)]$, a time shift is equivalent to a phase change.
4. The rate of oscillation of a discrete-time sinusoid increases as ω_0 increases from $\omega_0 = 0$ to $\omega_0 = \pi$. However, as ω_0 increases from $\omega_0 = \pi$ to $\omega_0 = 2\pi$, the oscillations become slower (see Figure 4.6). Therefore, low-frequencies (slow oscillations) are at the vicinity of $\omega_0 = k2\pi$ and high-frequencies (rapid oscillations) at the vicinity of $\omega_0 = \pi + k2\pi$.

Similar properties hold for the discrete-time complex exponentials

$$s_k[n] = A_k e^{j\omega_k n}. \quad -\infty < n < \infty \tag{4.18}$$

For $s_k[n]$ to be periodic with fundamental period N, the frequency ω_k should be a rational multiple of 2π, that is, $\omega_k = 2\pi k/N$. Therefore, all distinct complex exponentials with period N and frequency in the fundamental range, have frequencies given by $\omega_k = 2\pi k/N$, $k = 0, 1, \ldots, N - 1$. The set of sequences

$$s_k[n] = e^{j\frac{2\pi}{N}kn}, \quad -\infty < k, n < \infty \tag{4.19}$$

are periodic both in n (time) and k (frequency) with fundamental period N. As a result of the periodicity in k, there are only N distinct harmonically related complex exponentials with fundamental frequency $f_0 = 1/N$ and harmonics at frequencies $f_k = k/N, 0 \le k \le N - 1$. In summary, we have the properties

$$s_k[n + N] = s_k[n], \qquad \text{(periodic in time)} \tag{4.20}$$

$$s_{k+N}[n] = s_k[n]. \qquad \text{(periodic in frequency)} \tag{4.21}$$

Another very important feature of harmonically related discrete-time complex exponentials is the following orthogonality property (see Tutorial Problem 12):

$$\sum_{n=\langle N\rangle} s_k[n]s_m^*[n] = \sum_{n=\langle N\rangle} e^{j\frac{2\pi}{N}kn}e^{-j\frac{2\pi}{N}mn} = \begin{cases} N, & k = m \\ 0, & k \ne m \end{cases} \tag{4.22}$$

where by $\sum_{n=\langle N\rangle}$ we denote summation over any range of length N, that is, from $n = n_0$ to $n = n_0 + N - 1$ for any choice of n_0. The choice of n_0 is usually a matter of convenience. Most often, we choose $n_0 = 0$.
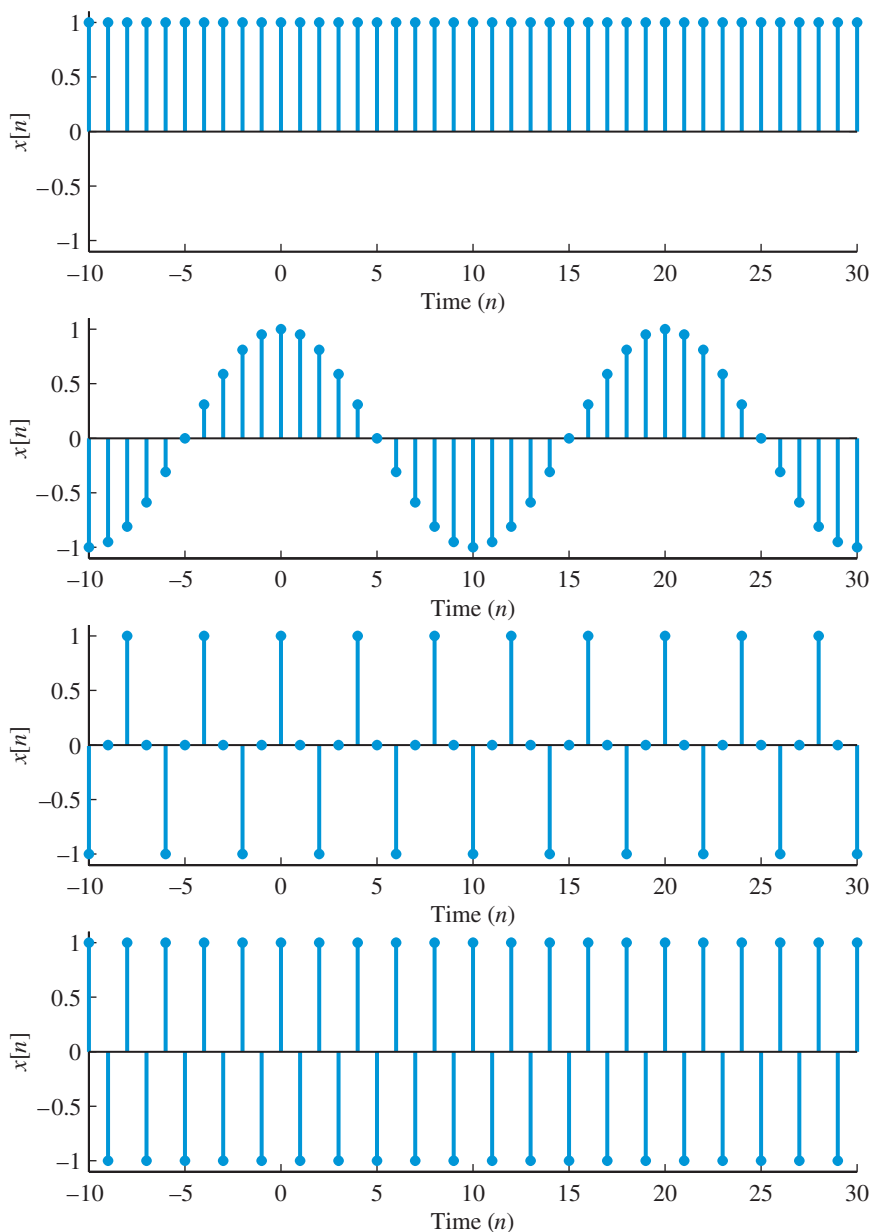
Figure 4.6 The signal $x[n] = \cos \omega_0 n$ for different values of ω_0. The rate of oscillation increases as ω_0 increases from 0 to π and decreases again as ω_0 increases from π to 2π.

Frequency variables and units After studying continuous- and discrete-time sinusoidal (or complex exponential) signals it is quite obvious that we are dealing with different (but related) frequency variables. To keep these variables in perspective and to avoid confusion it is important to be careful and consistent in using units to express them.
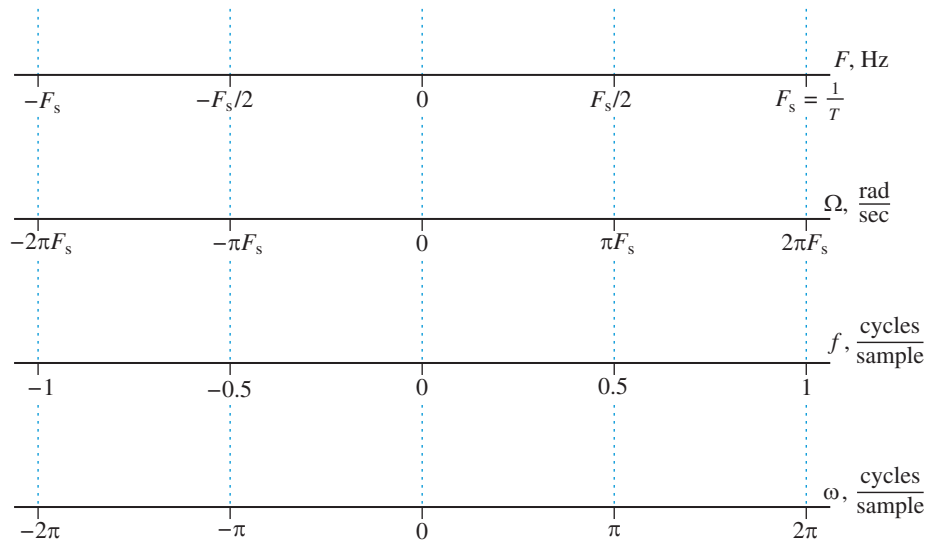
Figure 4.7 Frequency variables and their units.

In the continuous-time case, it is natural to represent the time variable t in units of *seconds*; the argument of the cosine function, $2\pi F t$, in units of *radians*; and the constant 2π in units of *radians per cycle* (or revolution). Therefore, it is reasonable to express the "analog" frequency, F, in units of *cycles per second* or Hertz (Hz), and analog angular frequency, $\Omega = 2\pi F$, in units of *radians per second*.

In the discrete-time case, if we assume that the units of the dimensionless integer index n are "samples" (*sampling intervals* would be a more appropriate term), then the units of "normalized" frequency, f, are *cycles per sample* and the units of normalized angular frequency, ω, are *radians per sample*. In the literature, this normalized frequency is also known as digital frequency. The frequency is normalized in the sense that it does not depend on the sampling interval.

However, if we specify the sampling interval, T, in seconds (or equivalently, the sampling frequency, F_s, in *samples per second*), we can use the "natural" time variable $t = nT$ instead of the index n. In this case, we can turn back to the "unnormalized" (or analog) frequency variables F (cycles per second) and Ω (radians per second). Basically, the meaning of frequency is the same in both continuous-time and discrete-time, namely *number of cycles per unit of the independent variable*; only the units change.

These notions of frequency variables and their units are graphically explained in Figure 4.7.

4.2 Fourier representation of continuous-time signals

In 1807, Fourier astounded some of his contemporaries by claiming that an arbitrary periodic function can be expressed as a linear combination of sines and cosines (Fourier series).

Euler and Lagrange thought that this was only possible for continuous functions. Fourier's discovery was that Fourier series representations are also valid for discontinuous functions. However, a rigorous mathematical proof of this result was provided by Dirichlet in 1837. In this section we use the theory of Fourier series to develop representations of continuous-time signals as series or integrals of complex exponentials. The corresponding representations for discrete-time signals are discussed in Section 4.3.

4.2.1 Fourier series for continuous-time periodic signals

Given a set of harmonically related complex exponentials $e^{jk\Omega_0 t}$, $k = 0, \pm 1, \pm 2, \ldots$, we synthesize a signal $x(t)$ using a linear combination of the form

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\Omega_0 t}, \tag{4.23}$$

where the coefficients c_k are constants. Each term in the summation is periodic with period $T_0 = 2\pi/\Omega_0$. Thus, if the infinite summation converges for all t, then its sum $x(t)$ is also periodic with period T_0. We note that the term $e^{jk\Omega_0 t}$ has fundamental period T_0/k. However, T_0 is the period shared by all terms of the series.

Suppose now that (4.23) is a valid representation of a periodic signal $x(t)$. What is the relation between the coefficients c_k and the function $x(t)$? To answer this, we multiply both sides of (4.23) by $e^{-jm\Omega_0 t}$, we change the order of integration with summation, we integrate over a full period, and then simplify the result using (4.7). The answer is

$$c_k = \frac{1}{T_0} \int_{T_0} x(t) e^{-jk\Omega_0 t} dt. \tag{4.24}$$

The pair of equations (4.23) and (4.24), when it exists, defines the *Fourier series representation* of a continuous-time periodic signal. We say that $x(t)$ and c_k are a *Continuous-Time Fourier Series (CTFS)* pair denoted by

Fourier Synthesis Equation   **Fourier Analysis Equation**

$$x(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\Omega_0 t} \quad \xleftarrow{\text{CTFS}} \quad c_k = \frac{1}{T_0} \int_{T_0} x(t) e^{-jk\Omega_0 t} dt. \tag{4.25}$$

The set of coefficients $\{c_k\}$ are known as the *Fourier series coefficients*.

Equation (4.24) analyzes ("breaks-up") a periodic signal $x(t)$ into a set of harmonic components $\{c_k e^{jk\Omega_0 t}\}$, whereas (4.23) synthesizes the signal $x(t)$ from its harmonic components. Equation (4.23) is known as the *synthesis* equation and (4.24) is known as the *analysis* equation.

The plot of $x(t)$ as a function of time t (waveform) provides a description of the signal in the time-domain. The plot of c_k as a function of frequency $F = kF_0$ (*spectrum*) constitutes a description of the signal in the frequency-domain. In view of (4.25) the two descriptions

are equivalent because we can go from $x(t)$ to c_k using the direct transform (4.24) and back from c_k to $x(t)$ using the inverse transform (4.23). In this sense, the CTFS is a pair of mutually inverse relations in that one undoes what the other does.

Since the coefficients c_k are, in general, complex-valued, we can express them in polar form

$$c_k = |c_k| e^{j\angle c_k}. \tag{4.26}$$

The plot of $|c_k|$ is known as the *magnitude spectrum* of $x(t)$, while the plot of $\angle c_k$ is known as the *phase spectrum* of $x(t)$. If c_k is real-valued, we can use a single plot, known as the *amplitude spectrum*.

Parseval's relation The average power in one period of $x(t)$ can be expressed in terms of the Fourier coefficients using Parseval's relation (see Problem 6):

$$P_{av} = \frac{1}{T_0} \int_{T_0} |x(t)|^2 dt = \sum_{k=-\infty}^{\infty} |c_k|^2. \tag{4.27}$$

The value of $|c_k|^2$ provides the portion of the average power of signal $x(t)$ that is contributed by the kth harmonic of the fundamental component. The graph of $|c_k|^2$ as a function of $F = kF_0$ is known as the *power spectrum* of the periodic signal $x(t)$. Because the power is distributed at a set of discrete frequencies, we say that periodic continuous-time signals have *discrete* or *line* spectra.

The following observations are useful when we deal with spectra of periodic signals:

- To emphasize the frequency-domain interpretation we define $c(kF_0) = c_k$ and plot $|c(kF_0)|$ and $\angle c(kF_0)$ as functions of the frequency $F = kF_0$.
- The spectral lines have uniform spacing $F_0 = 1/T_0$ determined by the fundamental period T_0 of $x(t)$. The shape of $x(t)$ is specified by the values of the Fourier coefficients c_k.
- If $x(t)$ is a real function of time, we have

$$c_{-k} = c_k^* = |c_k| e^{-j\angle c_k}, \tag{4.28}$$

which follows from (4.24) with k replaced by $-k$. Hence

$$|c_{-k}| = |c_k| \quad \text{and} \quad \angle(c_{-k}) = -\angle c_k, \tag{4.29}$$

which means that the magnitude spectrum has even symmetry and the phase spectrum has odd symmetry.

We now consider an example that brings out more clearly the relation between $x(t)$ and its Fourier series representation, $\{c_k\}$.
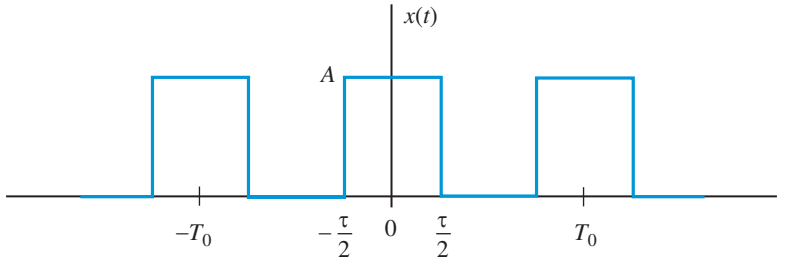
Figure 4.8 Rectangular pulse train.

Example 4.1   Rectangular pulse train
Consider the periodic rectangular pulse train in Figure 4.8. The Fourier coefficients are
given by

$$
c_k = \frac{1}{T_0} \int_{-\tau/2}^{\tau/2} A e^{-j2\pi k F_0 t} dt = \frac{A}{T_0} \left[\frac{e^{-j2\pi k F_0 t}}{-j2\pi k F_0} \right]_{-\tau/2}^{\tau/2}
$$

$$
= \frac{A}{\pi F_0 k T_0} \frac{e^{j\pi k F_0 \tau} - e^{-j\pi k F_0 \tau}}{2j}
$$

$$
= \frac{A\tau}{T_0} \frac{\sin \pi k F_0 \tau}{\pi k F_0 \tau}. \quad k = 0, \pm 1, \pm 2, \ldots \tag{4.30}
$$

The values of c_k are obtained by evaluating the function $(A\tau/T_0) \sin(\phi)/\phi$ at equidistant
points $\phi = k(\pi F_0 \tau)$. Since $\lim_{\phi \to 0}[\sin(\phi)/\phi] = 1$, we have $c_0 = A\tau/T_0$. The function
$\sin(\phi)/\phi$ has zero crossings at multiples of π, that is, at $\phi = m\pi$, $m = 0, \pm 1, \pm 2, \ldots$ The
zero crossings occur at $\phi = \pi F \tau = m\pi$ or $F = m/\tau$. The spacing $F = 1/\tau$ between the
zero crossings is determined by the width τ of the pulse, whereas the spacing $F_0 = 1/T_0$
between the spectral lines is determined by the fundamental period T_0.   ■

When the Fourier coefficients are real, we can plot c_k on a single graph. However, for
consistency, we plot the magnitude and phase spectra (see Figure 4.9). To obtain these
magnitude and phase spectra, we use the following general conventions:

- Phase angles are always measured with respect to cosine waves. Thus, sine waves have
  a phase of $-\pi/2$ since $\sin \Omega t = \cos(\Omega t - \pi/2)$.
- Magnitude spectra are always positive. Hence, negative signs should be absorbed in the
  phase using the identity: $-A \cos \Omega t = \cos(\Omega t \pm \pi)$. It does not matter whether we take
  $+\pi$ or $-\pi$ because $\cos(-\pi) = \cos \pi$. However, we use both $+\pi$ and $-\pi$ to bring out
  the odd symmetry of the phase.

Sinc function The function $\sin(\phi)/\phi$, known as a *sinc* function, arises frequently in
Fourier analysis and in the study of LTI systems. A commonly used definition is

$$
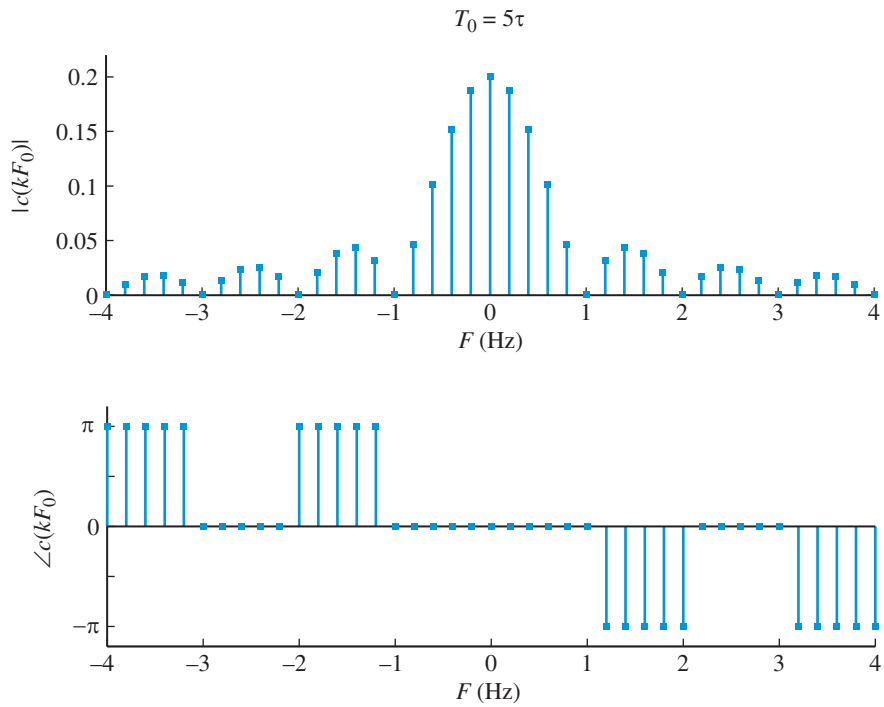\text{sinc}(\theta) = \frac{\sin \pi \theta}{\pi \theta}. \tag{4.31}
$$

Figure 4.9 Magnitude and phase spectra of a rectangular pulse train with $A = 1$ and $T_0 = 5\tau$.
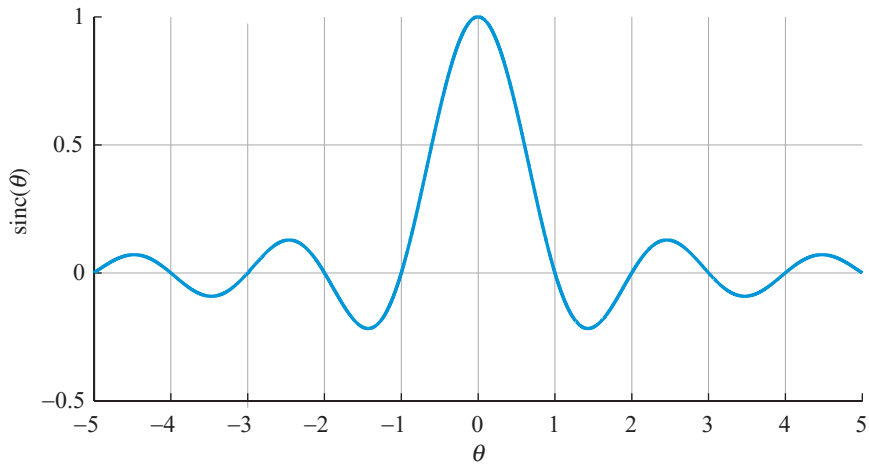

Figure 4.10 The sinc function.

From (4.31) and Figure 4.10, we see that the sinc function has the following properties:

1. The sinc function is an even function of θ, that is, $\text{sinc}(-\theta) = \text{sinc}(\theta)$.
2. $\text{sinc}(\theta) = 0$ when $\sin\theta = 0$, except at $\theta = 0$, where it appears indeterminate. This means that $\text{sinc}(\theta) = 0$ when $\theta = \pm 1, \pm 2, \ldots$

3. Using l'Hôpital's rule, we can show that $\text{sinc}(0) = 1$.
4. $\text{sinc}(\theta)$ is the product of the periodic function $\sin \pi \theta$ with the monotonically decreasing function $1/(\pi \theta)$. Hence, $\text{sinc}(\theta)$ exhibits sinusoidal oscillations of period $\theta = 2$ with amplitude decreasing continuously as $1/(\pi \theta)$.

In MATLAB (4.31) can be evaluated using the function `sinc(theta)`.

Convergence conditions For a periodic signal $x(t)$ to have a Fourier series representation, it is necessary that (1) the coefficients obtained from the analysis equation (4.24) should be finite; and (2) when these coefficients are substituted into the synthesis equation (4.23), the resulting infinite series should converge (in some sense) to the signal $x(t)$.

The following set of sufficient conditions, known as Dirichlet conditions, guarantee the existence of Fourier series for all periodic signals of practical interest:

1. The periodic signal $x(t)$ is absolutely integrable over any period, that is, $x(t)$ has a finite area per period

$$\int_{T_0} |x(t)| dt < \infty. \tag{4.32}$$

This condition guarantees that the Fourier coefficients are finite.
2. The periodic signal $x(t)$ has a finite number of maxima, minima, and finite discontinuities per period.
   This condition guarantees that, as $m \to \infty$, the partial sum

$$x_m(t) = \sum_{k=-m}^{m} c_k e^{jk\Omega_0 t} \tag{4.33}$$

converges to $x(t)$ wherever $x(t)$ is continuous, and to the average of the values on either side of t_0, that is, to $[x(t_0-) + x(t_0+)]/2$, if $x(t)$ has a finite discontinuity at t_0.

Another type of convergence is assured if the signal $x(t)$ is square integrable,

$$\int_{T_0} |x(t)|^2 dt < \infty. \tag{4.34}$$

Under this condition, the series converges in the mean square error sense, that is,

$$\lim_{m \to \infty} \int_{T_0} |x(t) - x_m(t)|^2 dt = 0. \tag{4.35}$$

We emphasize that (4.35) does not imply that the signal $x(t)$ and the Fourier series are equal at every value of t; it simply guarantees that the energy of the approximation error signal is zero.

To understand how Fourier series represent periodic signals, we consider how the partial sum (4.33) approximates a periodic signal $x(t)$ and what is the nature of the approximation error

$$e_m(t) = x(t) - x_m(t) = x(t) - \sum_{k=-m}^{m} c_k e^{jk\Omega_0 t}. \tag{4.36}$$
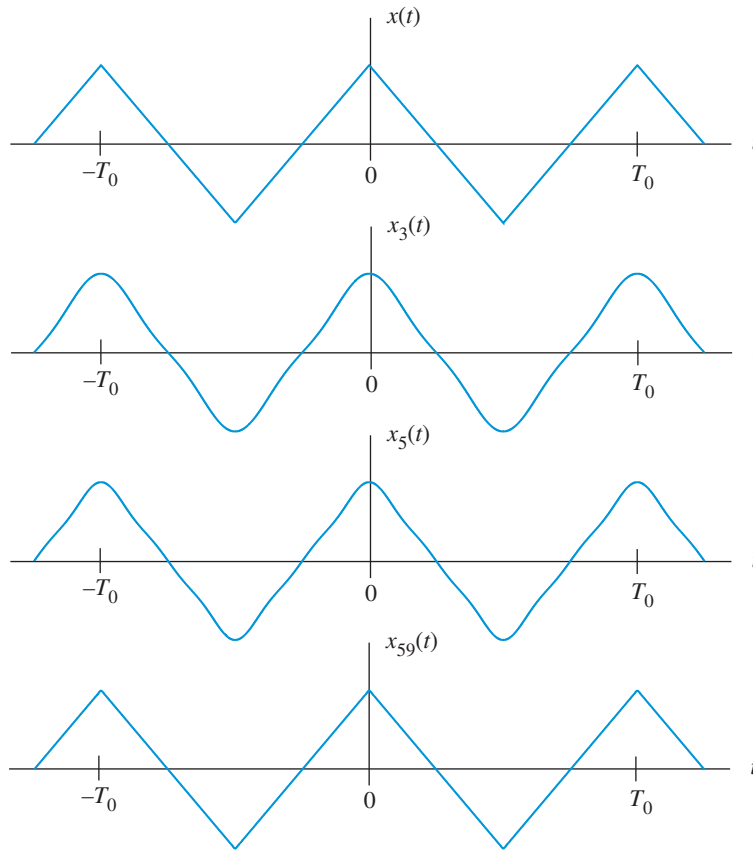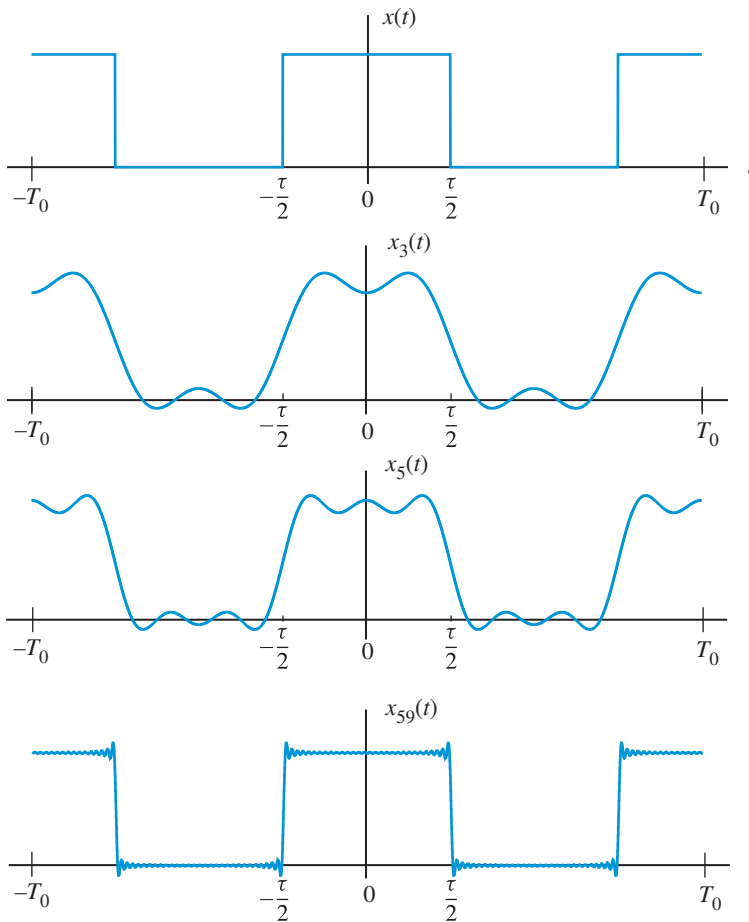
Figure 4.11 Fourier series approximation of a triangular pulse train.

By looking at graphs of $x_m(t)$ or the error $e_m(t)$ and comparing them to the graph of $x(t)$, we can get an intuitive feel for whether a given function can be represented by a Fourier series, and, when a function can be so represented, for the number of terms needed to get a good approximation to the function.

Figure 4.11 shows the partial sum for $m = 3, 5, 59$ for a triangular pulse train (see Problem 23). We note that as m increases, the approximation curves $y = x_m(t)$ approach more and more nearly to the curve $y = x(t)$. Since there are no jumps, we expect $x_m(t)$ to converge to $x(t)$ everywhere. However, as expected, the convergence is better at the continuous segments and poorer at the "corners" of $x(t)$.

Figure 4.12 illustrates the approximation of a rectangular pulse train using the partial sum with $m = 3, 5, 59$. We note that, even for large values of m, the approximation curves $y = x_m(t)$ differ significantly from the curve $y = x(t)$. The key difference between the rectangular and triangular pulses is the finite jumps of the rectangular train at $\pm\tau/2$, and so on. Since the values of the function $x(t)$ are *not* defined at these points of discontinuity, it is not reasonable to expect convergence of the Fourier series at these points. However, the Fourier series handles such situations in a very reasonable way: it converges to the average

Figure 4.12 Fourier series approximation of a rectangular pulse train.

of the left- and right-hand limits at the jump. As expected, the approximation is poorest in the vicinity of the jump points.

These examples are representative of the waveforms encountered in practical applications. Indeed, all periodic signals of practical interest are continuous with the possibility of some "corners" or "jumps" in a single period. For such signals, the Fourier series representation converges and equals the original signal $x(t)$ at every value of t except at the isolated points of discontinuity at which the series converges to the average of the left- and right-hand limits of the discontinuity.

Gibbs phenomenon Regardless of whether a periodic signal is absolutely integrable or square integrable, the Fourier series exhibits a behavior known as *Gibbs phenomenon* at the vicinity of points of discontinuity. Figure 4.13 illustrates this effect for one of the discontinuities of the rectangular pulse discussed in Example 4.1. The partial sum, even for large values of m, exhibits an oscillatory overshoot with period $T_0/(2m)$ and peak value of about 9 percent of the height of the jump. As m increases, the ripples are squeezed closer to
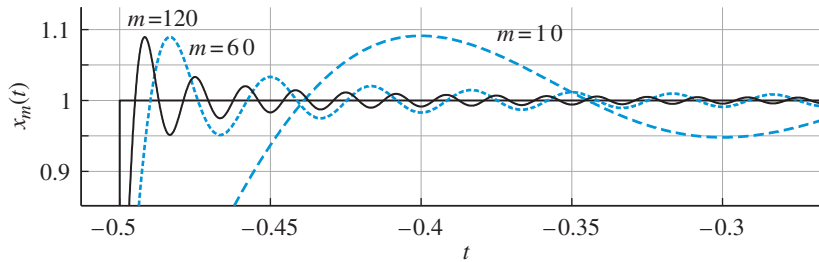
Figure 4.13 Illustration of Gibbs phenomenon.

the discontinuity and the area "under the ripples" decreases; eventually, the area becomes zero as $m \to \infty$. However, the size of the overshoot does not decrease and remains the same for any finite value of m. Therefore, when we approximate a discontinuous signal with a truncated Fourier series we should choose a large value of m to ensure that the total energy of the ripples is insignificant. The Gibbs phenomenon has significant implications for the design of practical filters for processing real-world signals.

4.2.2 Fourier transforms for continuous-time aperiodic signals

We noted that a continuous-time periodic signal, with fundamental period T_0, can be expressed as a linear combination of complex exponentials with frequencies uniformly spaced at intervals of $F_0 = 1/T_0$. Since we can think of an aperiodic signal as a periodic signal with infinite period, we shall use an intuitive limiting process to develop a Fourier representation for aperiodic signals using the Fourier series.

Example 4.2   From Fourier series to Fourier transform
We start with a pictorial illustration using the Fourier series of the rectangular pulse train discussed in Example 4.1. More specifically, over one period

$$x(t) = \begin{cases} A, & |t| < \tau \\ 0, & \tau < |t| < T_0/2 \end{cases} \tag{4.37}$$

and periodically repeats with period T_0. The Fourier coefficients are given by

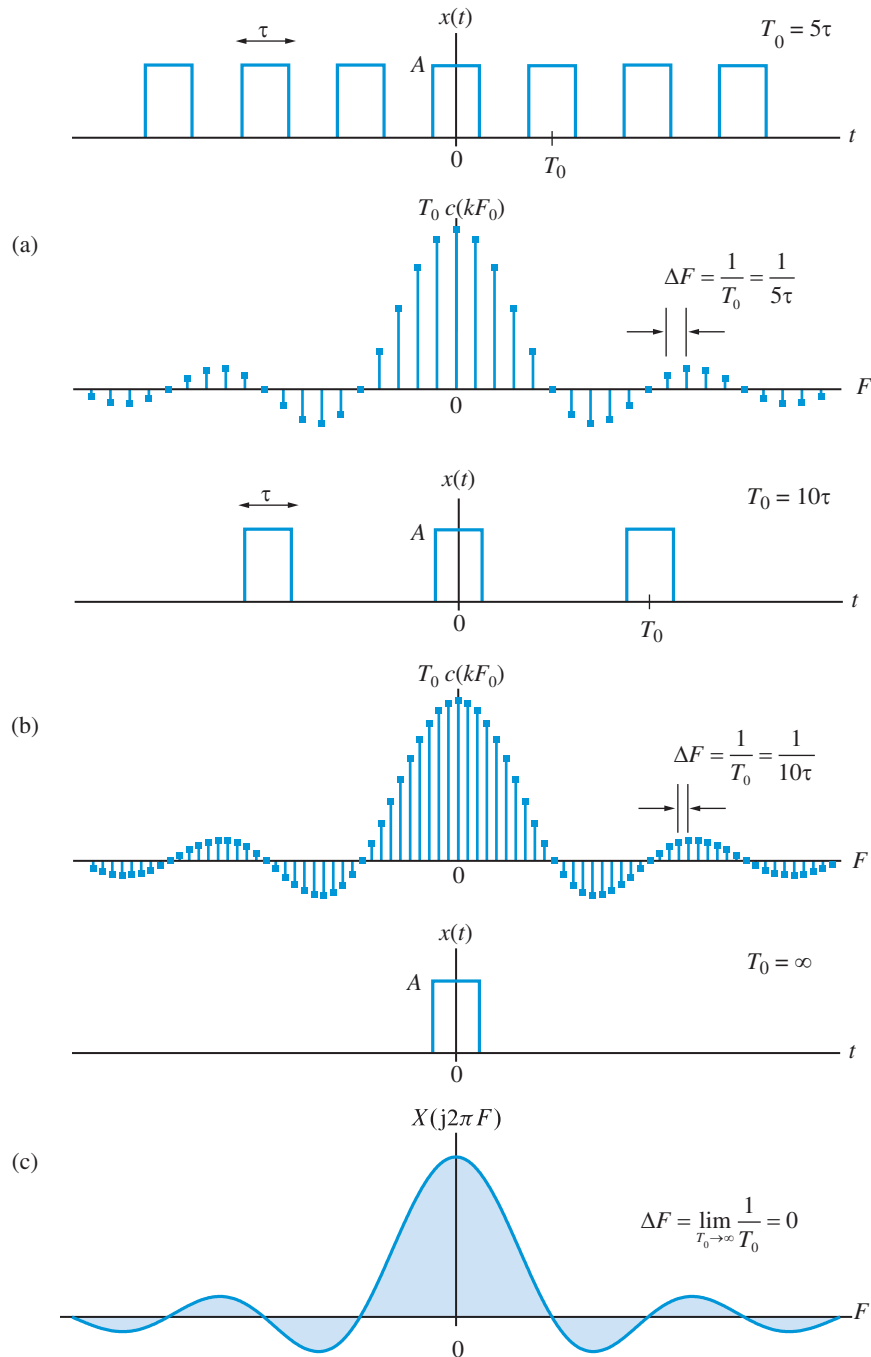$$c_k = \frac{A\tau}{T_0} \frac{\sin \pi k F_0 \tau}{\pi k F_0 \tau} \triangleq c(kF_0). \tag{4.38}$$

The size of the coefficients c_k depends on the period T_0 and $c_k \to 0$ as $T_0 \to \infty$. To avoid this problem, we consider the scaled coefficients

$$c(kF_0)T_0 = A\tau \left. \frac{\sin \pi F \tau}{\pi F \tau} \right|_{F=kF_0}, \tag{4.39}$$

which can be thought of as equally spaced samples of the envelope function. As T_0 increases, the spacing $\Delta F = F_0 = 1/T_0$ between the spectral lines decreases. As $T_0 \to \infty$

Figure 4.14 Transition from the CTFS to CTFT: (a) the periodic signal $x(t)$ and its scaled CTFS for the fundamental period $T_0 = 5\tau$, (b) the periodic signal $x(t)$ and its scaled CTFS for the fundamental period $T_0 = 10\tau$, and (c) the aperiodic signal $x(t)$ and its CTFT when the period extends to infinity.

(a) in the time domain the result is an aperiodic signal corresponding to one period of the rectangular pulse train, and (b) in the frequency domain the result is a "continuum" of spectral lines. This limiting process is illustrated in Figure 4.14. ■

To find the mathematical expression for the Fourier representation of aperiodic signals, we consider a finite duration signal $x(t) = 0$ for $|t| > \tau/2$, and repeat it with period $T_0 > \tau$ to create a periodic signal $x_p(t)$. The Fourier series representation of $x_p(t)$ is

$$x_p(t) = \sum_{k=-\infty}^{\infty} c_k e^{j2\pi kF_0 t}, \tag{4.40}$$

and

$$c_k = \frac{1}{T_0} \int_{-T_0/2}^{T_0/2} x_p(t) e^{-j2\pi kF_0 t} dt. \tag{4.41}$$

Since $x(t) = x_p(t)$ for $|t| < T_0/2$ and $x(t) = 0$ for $|t| > T_0/2$, we can write (4.41) as

$$c(kF_0)T_0 = \int_{-\infty}^{\infty} x(t) e^{-j2\pi kF_0 t} dt. \tag{4.42}$$

If we set $F = kF_0$, the integral in (4.42) becomes a function $X(j2\pi F)$ which is basically the envelope of $c_k T_0$. Therefore, we can express (4.42) as

$$X(j2\pi F) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi Ft} dt, \tag{4.43}$$

which is called the *Fourier transform* or *Fourier integral* of $x(t)$. Comparing (4.43) with (4.42), we obtain

$$c_k = \frac{X(j2\pi kF_0)}{T_0} = F_0 X(j2\pi F)|_{F=kF_0} = X(j2\pi K\Delta F)\Delta F. \tag{4.44}$$

Thus, the Fourier coefficients c_k of a periodic signal $x_p(t)$ are proportional to uniformly spaced samples of the Fourier transform of one period of $x_p(t)$. This relation holds for every signal $x(t)$ that is equal to $x_p(t)$ over exactly one period, that is

$$x(t) = \begin{cases} x_p(t), & t_0 < t < t_0 + T_0 \\ 0. & \text{otherwise} \end{cases} \tag{4.45}$$

The Fourier series (4.40) represents the periodic signal $x_p(t)$ as a summation of complex exponentials. To obtain the corresponding equation for the aperiodic signal $x(t)$, we recall from (4.44) that $c_k = X(j2\pi k\Delta F)\Delta F$ and express (4.40) as

$$x_p(t) = \sum_{k=-\infty}^{\infty} X(j2\pi k\Delta F) e^{j2\pi k\Delta Ft} \Delta F. \tag{4.46}$$

We note that as $T_0 \to \infty$, $x_p(t) \to x(t)$. Furthermore, $\Delta F \to 0$ as $T_0 \to \infty$, and the right hand side of (4.46) passes to an integral. The result is

$$x(t) = \int_{-\infty}^{\infty} X(j2\pi F)\, e^{j2\pi Ft} dF, \qquad (4.47)$$

which is called the *inverse Fourier transform*. The integral (4.47) provides a representation of the aperiodic signal $x(t)$ as a "continuous" summation of complex exponentials. Basically, the Fourier series becomes a Fourier integral in the limit as $T_0 \to \infty$. This is illustrated in Figure 4.14.

We say that $x(t)$ and $X(j2\pi F)$ constitute a *Continuous-Time Fourier Transform (CTFT)* pair, which is denoted by

Fourier Synthesis Equation　　　　　**Fourier Analysis Equation**

$$x(t) = \int_{-\infty}^{\infty} X(j2\pi F)\, e^{j2\pi Ft} dF \xleftarrow{\text{CTFT}} X(j2\pi F) = \int_{-\infty}^{\infty} x(t) e^{-j2\pi Ft} dt, \qquad (4.48)$$

or more concisely

$$x(t) = \mathcal{F}^{-1}\{X(j2\pi F)\} \xleftarrow{\text{CTFT}} X(j2\pi F) = \mathcal{F}\{x(t)\}. \qquad (4.49)$$

A comparison of (4.25) and (4.48) indicates that $X(j2\pi F)$ plays the same role for aperiodic signals that $c(kF_0)$ plays for periodic signals. Thus, $X(j2\pi F)$ is the *spectrum* of the aperiodic signal $x(t)$. Periodic signals must have discrete spectra with lines at harmonically related frequencies; otherwise they cannot be periodic. A continuous spectrum results in an aperiodic signal because almost all frequencies in a continuous interval are not harmonically related. It is helpful to keep in mind that the CTFT is of the same nature as a CTFS with fundamental frequency $F_0 = 1/T_0 \to 0$.

Convergence　The conditions for the convergence of CTFT are similar to those required for CTFS. If $x(t)$ has a finite number of minima, maxima, and discontinuities in any finite interval, and it is absolutely integrable, that is,

$$\int_{-\infty}^{\infty} |x(t)| dt < \infty, \qquad (4.50)$$

the signal $\hat{x}(t) = \mathcal{F}^{-1}\{X(j2\pi F)\}$ converges to $x(t)$ for any t where $x(t)$ is continuous. At a discontinuity, $\hat{x}(t)$ is equal to the average of the values on either side of the discontinuity.

If $x(t)$ has finite energy, that is, it is square integrable

$$\int_{-\infty}^{\infty} |x(t)|^2 dt < \infty, \qquad (4.51)$$

we are only guaranteed that the energy of the error signal $e(t) = x(t) - \hat{x}(t)$ is zero. However, the signals $x(t)$ and $\hat{x}(t)$ may differ significantly at individual points.

Parseval's relation For aperiodic signals with finite energy, Parseval's relation is given by

$$\int_{-\infty}^{\infty} |x(t)|^2 dt = \int_{-\infty}^{\infty} |X(j2\pi F)|^2 dF, \tag{4.52}$$

and states that the total energy of $x(t)$ may be obtained either from the signal itself or from its spectrum. The quantity $|X(j2\pi F)|^2 \Delta F$, for a small ΔF, provides the energy of the signal in a narrow frequency band of width ΔF centered at frequency F. Single frequencies do not contribute into the total energy because $\Delta F = 0$. For this reason, $|X(j2\pi F)|^2$ is known as the *energy-density spectrum* of the signal $x(t)$.

For convenience, we sometimes express the CTFT in terms of the radian frequency $\Omega = 2\pi F$ as follows:

$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega)\, e^{j\Omega t} d\Omega \xleftarrow{\text{CTFT}} X(j\Omega) = \int_{-\infty}^{\infty} x(t) e^{-j\Omega t} dt. \tag{4.53}$$

Example 4.3   Causal exponential signal

Consider the signal

$$x(t) = \begin{cases} e^{-at}, & t > 0 \\ 0. & t < 0 \end{cases} \tag{4.54}$$

This signal is absolutely integrable if $a > 0$. From the definition (4.43),

$$X(j2\pi F) = \int_{0}^{\infty} e^{-at} e^{-j2\pi Ft} dt = -\frac{1}{a + j2\pi F} e^{-(a+j2\pi F)t} \Big|_{0}^{\infty}. \tag{4.55}$$

Hence,

$$X(j2\pi F) = \frac{1}{a + j2\pi F} \quad \text{or} \quad X(j\Omega) = \frac{1}{a + j\Omega}. \quad a > 0 \tag{4.56}$$
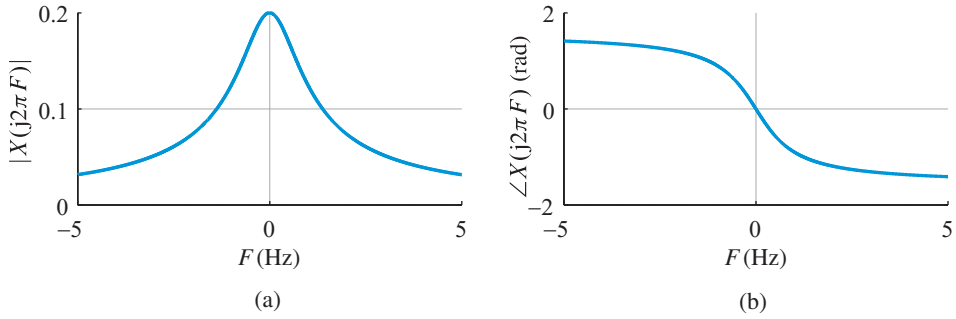
Since $X(j2\pi F)$ is complex valued, we express it in terms of its magnitude and phase

$$|X(j2\pi F)| = \frac{1}{\sqrt{a^2 + (2\pi F)^2}}, \tag{4.57a}$$

and

$$\angle X(j2\pi F) = -\tan^{-1}\left(2\pi \frac{F}{a}\right). \quad a > 0 \tag{4.57b}$$

Figure 4.15 Fourier transform of the signal $x(t) = e^{-at}u(t)$ for $a = 5$. (a) Magnitude and (b) phase of $X(j2\pi F)$ in the finite interval $-5 < F < 5$ (Hz).

The magnitude spectrum $|X(j2\pi F)|$ and the phase spectrum $\angle X(j2\pi F)$ are depicted in Figure 4.15. Because $x(t)$ is a real function of t, $|X(j2\pi F)|$ has even symmetry and $\angle X(j2\pi F)$ has odd symmetry. ∎

Example 4.4 Rectangular pulse signal

Consider the signal

$$x(t) = \begin{cases} A, & |t| < \tau/2 \\ 0. & |t| > \tau/2 \end{cases} \tag{4.58}$$

This signal is absolutely integrable for any finite τ. From the definition (4.43),

$$X(j2\pi F) = \int_{-\tau/2}^{\tau/2} A e^{-j2\pi Ft} dt = A\tau \frac{\sin(\pi F\tau)}{\pi F\tau}. \tag{4.59}$$

The signal $x(t)$ and the spectra $X(j2\pi F)$, $|X(j2\pi F)|$, and $\angle X(j2\pi F)$ are depicted in Figure 4.16. We recall that a negative amplitude can be considered as a positive amplitude with a phase of $-\pi$ or π. Any choices that guarantee the odd symmetry of the phase spectrum are equivalent. ∎

From the development of the CTFT as a limiting form of the CTFS, we might expect that the synthesis equation for the rectangular pulse would exhibit similar convergence properties with the rectangular pulse train. Indeed, let us consider the "partial-integral" approximation

$$\hat{x}(t) = \int_{-B}^{B} X(j2\pi F) e^{j2\pi Ft} dF = \int_{-B}^{B} A\tau \frac{\sin(\pi F\tau)}{\pi F\tau} e^{j2\pi Ft} dF. \tag{4.60}$$

Since $x(t)$ is square integrable, it is guaranteed that

$$\lim_{B\to\infty} \int_{-B}^{B} |x(t) - \hat{x}(t)|^2 dt = 0. \tag{4.61}$$
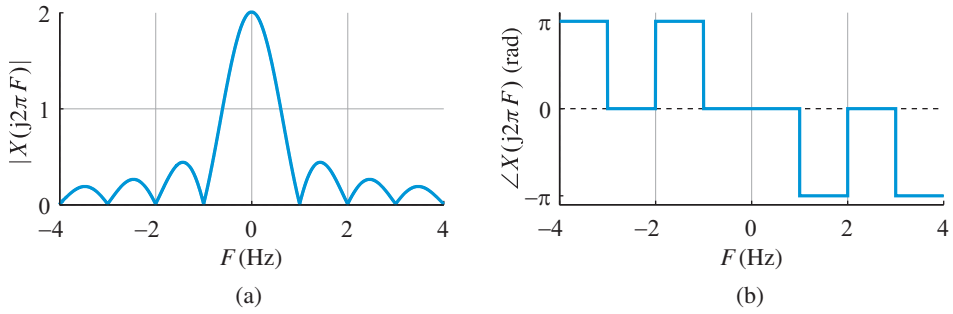
Figure 4.16 (a) Magnitude and (b) phase of the Fourier transform of a continuous-time rectangular pulse.

Furthermore, since $x(t)$ satisfies the Dirichlet conditions, $x(t) = \hat{x}(t)$, except at the discontinuities $t = \pm\tau/2$ where $\hat{x}(t)$ converges to $A/2$. The approximation $\hat{x}(t)$ exhibits a similar Gibbs phenomenon for any finite value of B.

Example 4.5   Multiplying a periodic with an aperiodic signal

Consider an aperiodic signal $x(t)$ with Fourier transform $X(j2\pi F)$ and a periodic signal $s(t)$ with fundamental frequency F_0 and Fourier coefficients c_k. The product $x_s(t) = x(t)s(t)$ is clearly an aperiodic signal. Using the Fourier synthesis equation in (4.25) for $s(t)$, the Fourier transform, $X_s(j2\pi F)$, of $x_s(t)$ is given by

$$X_s(j2\pi F) = \int_{-\infty}^{\infty} x(t) \left[\sum_{k=-\infty}^{\infty} c_k e^{j2\pi F_0 kt} \right] e^{-j2\pi Ft} dt$$

$$= \sum_{k=-\infty}^{\infty} c_k \left[\int_{-\infty}^{\infty} x(t) e^{-j2\pi (F-kF_0)t} dt \right].$$

The term in the brackets can be seen to be the Fourier transform of $x(t)$ shifted so that the center $X(0)$ is located at frequency $F = kF_0$. Thus, we have

$$X_s(j2\pi F) = \sum_{k=-\infty}^{\infty} c_k X[j2\pi (F - kF_0)]. \tag{4.62}$$

The spectrum of $x_s(t)$ is obtained by putting copies of $X(j2\pi F)$, scaled by c_k, at integer multiples of F_0 and then adding them together. We note that if $X(j2\pi F) = 0$ for $|F| < B$ and $F_0 > 2B$, we have $X(j2\pi F) = X_s(j2\pi F)/c_0$ for $|F| < B$; thus, we can recover $x(t)$ from $x_s(t)$ using the inverse Fourier transform. If $s(t)$ is the periodic pulse train in Figure 4.12 with $A = 1$ and $\tau \ll T_0$, $x_s(t)$ provides short segment samples of $x(t)$ every T_0 seconds. This idea, which provides a highly informative approach to sampling theory, is shown in Figure 4.17 and is further discussed in Tutorial Problem 9.   ∎
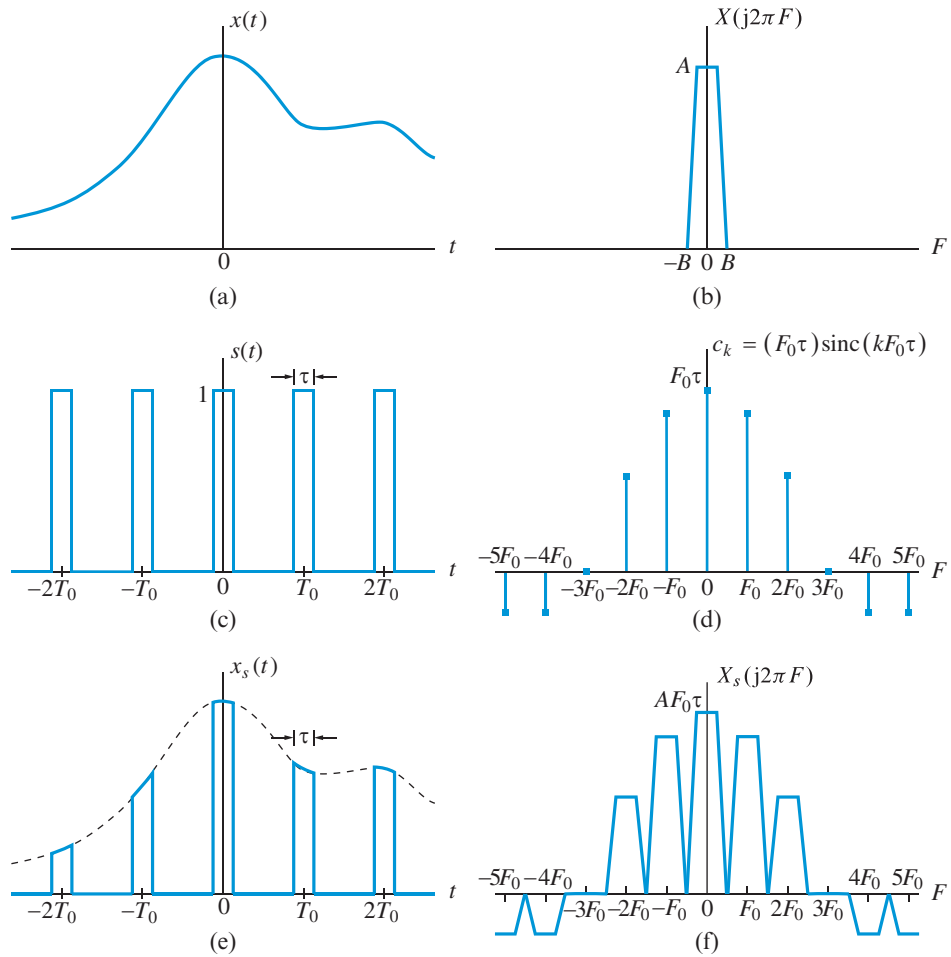
Figure 4.17 Signals and their Fourier descriptions in Example 4.5: (a) aperiodic signal $x(t)$, (b) Fourier transform $X(j2\pi F)$, (c) periodic signal $s(t)$, (d) Fourier series c_k, (e) aperiodic signal $x_s(t)$, and (f) Fourier transform $X_s(j2\pi F)$.

4.3   Fourier representation of discrete-time signals

In this section, we develop Fourier representations for discrete-time signals by following an approach which parallels that for continuous-time signals.

4.3.1   Fourier series for discrete-time periodic signals

Consider a linear combination of N harmonically related complex exponentials

$$x[n] = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn}. \qquad (4.63)$$

The sequence $x[n]$ is periodic with fundamental period N. Indeed, we have

$$x[n+N] = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}(k+N)n} = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn} e^{j2\pi n} = x[n]. \tag{4.64}$$

To determine the coefficients c_k from the values of the periodic signal $x[n]$, we exploit the orthogonality property (4.22) of harmonically related complex exponentials. Indeed, after multiplying both sides of (4.63) by $e^{-j(2\pi/N)mn}$ and summing from $n = 0$ to $n = N - 1$, we obtain

$$\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}mn} = \sum_{n=0}^{N-1}\sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}(k-m)n}. \tag{4.65}$$

After interchanging the order of the summations on the right hand side, we have

$$\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}mn} = \sum_{k=0}^{N-1} c_k \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(k-m)n}. \tag{4.66}$$

Using relation (4.22), the right hand side is equal to Nc_m for $k = m$ and zero for $k \neq m$. Solving for c_m and changing the index m to k, we obtain

$$c_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}. \tag{4.67}$$

The sequence c_k, $k = 0, \pm 1, \pm 2, \cdots$, is periodic with fundamental period N.

Equation (4.67) provides a closed-form expression for obtaining the Fourier series coefficients required by the Fourier series (4.63). The result is the *Discrete-Time Fourier Series (DTFS)* pair:

Fourier Synthesis Equation   **Fourier Analysis Equation**

$$x[n] = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn} \quad \xleftarrow{\text{DTFS}} \quad c_k = \frac{1}{N}\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}. \tag{4.68}$$

Parseval's relation The average power in one period of $x[n]$ can be expressed in terms of the Fourier series coefficients using the following form of Parseval's relation (see Problem 41):

$$P_{\text{av}} = \frac{1}{N}\sum_{n=0}^{N-1} |x[n]|^2 = \sum_{k=0}^{N-1} |c_k|^2. \tag{4.69}$$

The value of $|c_k|^2$ provides the portion of the average power of $x[n]$ that is contributed by its kth harmonic component. Since $c_{k+N} = c_k$, there are only N distinct harmonic components. The graph of $|c_k|^2$ as a function of $f = k/N$, $\omega = 2\pi k/N$, or simply k, is known as the *power spectrum* of the periodic signal $x[n]$. The following examples illustrate the use and properties of DTFS.

Example 4.6   Sinusoidal sequence

Consider the signal

$$x[n] = \cos \omega_0 n = \cos 2\pi f_0 n, \tag{4.70}$$

which is periodic only if f_0 is a ratio of two integers. Suppose that $f_0 = k_0/N$, $0 \leq k_0 \leq N - 1$. Then, $x[n]$ has a DTFS representation. To determine the Fourier coefficients, we first express (4.70) as a sum of complex exponentials as follows:

$$x[n] = \frac{1}{2} e^{j \frac{2\pi}{N} k_0 n} + \frac{1}{2} e^{-j \frac{2\pi}{N} k_0 n} = \frac{1}{2} e^{j \frac{2\pi}{N} k_0 n} + \frac{1}{2} e^{j \frac{2\pi}{N} (N-k_0) n}. \tag{4.71}$$

Comparing (4.71) with (4.63), we obtain

$$c_{k_0} = \frac{1}{2}, \quad c_{N-k_0} = \frac{1}{2}, \tag{4.72}$$

and the remaining coefficients in the interval $0 \leq k \leq N - 1$ are zero. If k_0 and N are prime, $x[n]$ has fundamental period N. Figure 4.18 shows the amplitude spectrum of $x[n]$ for $k_0 = 2$ and $N = 5$. The coefficients outside the fundamental interval $0 \leq k \leq N - 1$ are obtained by periodic repetition. ∎

Example 4.7   Periodic impulse train

Consider the Fourier series representation of the periodic impulse train

$$\delta_N[n] \triangleq \sum_{\ell=-\infty}^{\infty} \delta[n - \ell N] = \begin{cases} 1, & n = mN, \ m \text{ any integer} \\ 0. & \text{otherwise} \end{cases} \tag{4.73}$$
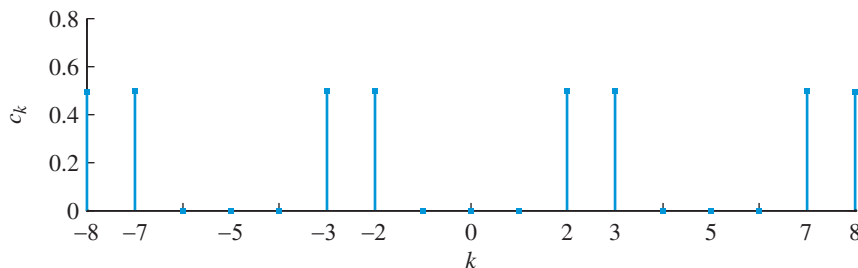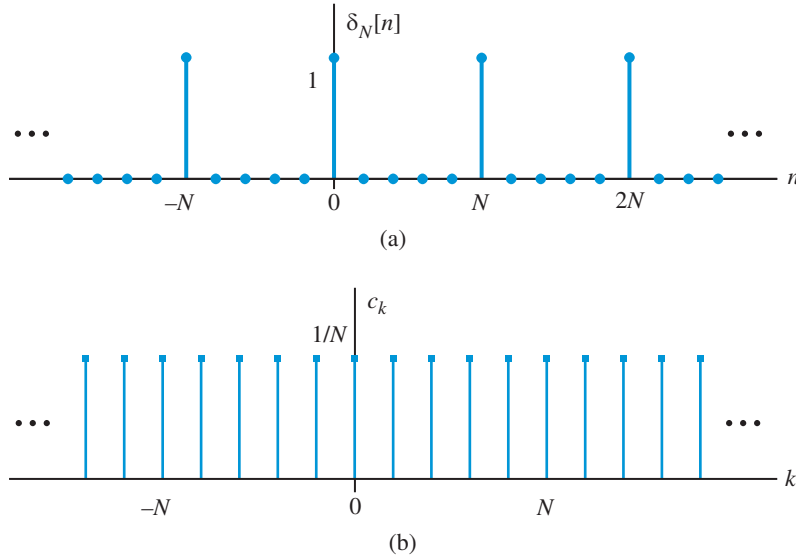

Figure 4.18 Plot of the DTFS of the sinusoidal sequence $x[n] = \cos(2\pi(2/5)n)$.

(a)

(b)

Figure 4.19 The periodic impulse train sequence $\delta_N[n]$ (a), and its DTFS c_k (b).

Since $\delta_N[n] = \delta[n]$ for $0 \le n \le N - 1$, the Fourier series coefficients are given by

$$c_k = \frac{1}{N} \sum_{n=0}^{N-1} \delta[n] e^{-j\frac{2\pi}{N}kn} = \frac{1}{N}, \quad \text{all } k. \tag{4.74}$$

The Fourier series representation of $\delta_N[n]$ is

$$\delta_N[n] = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn} = \frac{1}{N} \sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}kn}, \quad \text{all } n. \tag{4.75}$$

The periodic impulse train $\delta_N[n]$ and its spectrum are illustrated in Figure 4.19. ■

Example 4.8 Rectangular pulse train

Consider the rectangular pulse train sequence shown in Figure 4.20(a), where $N > 2L + 1$. Due to the even symmetry of $x[n]$, it is convenient to compute the Fourier coefficients using the following summation:

$$c_k = \frac{1}{N} \sum_{n=-L}^{L} e^{-j\frac{2\pi}{N}kn}. \tag{4.76}$$

Changing the index of summation, from n to $m = n + L$, equation (4.76) becomes

$$c_k = \frac{1}{N} \sum_{m=0}^{2L} e^{-j\frac{2\pi}{N}k(m-L)} = \frac{1}{N} e^{j\frac{2\pi}{N}kL} \sum_{m=0}^{2L} \left(e^{-j\frac{2\pi}{N}k} \right)^m. \tag{4.77}$$
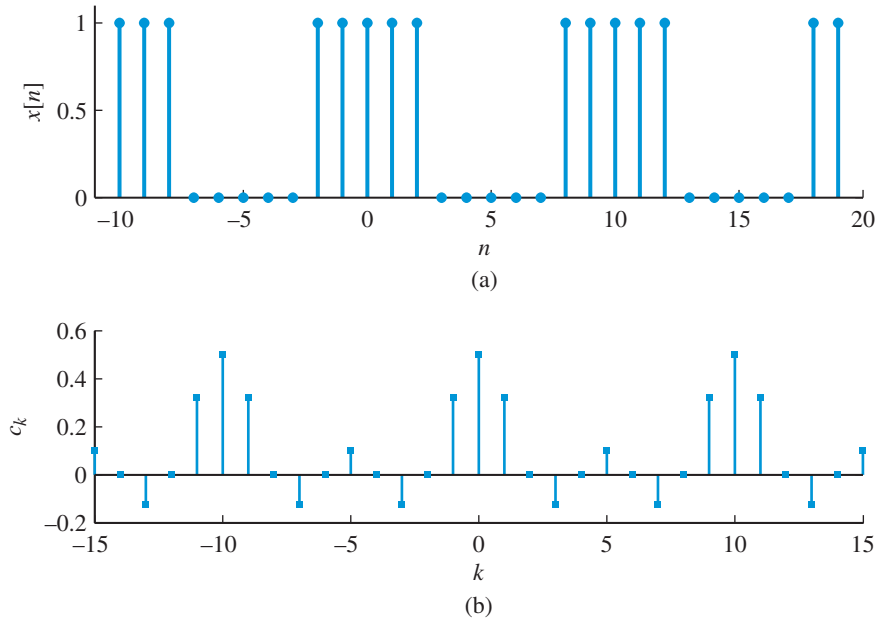
Figure 4.20 The periodic rectangular pulse train sequence $x[n]$ with $L=2$ and period $N=10$ (a) has a DTFS c_k with period $N=10$ (b).

For $k = 0$, we can easily see that $c_0 = (2L+1)/N$. To determine c_k for $k = 1, \ldots, N-1$, we use the geometric summation formula (2.120). The result is

$$c_k = \frac{1}{N} e^{j\frac{2\pi}{N} kL} \left[\frac{1 - e^{-j\frac{2\pi}{N} k(2L+1)}}{1 - e^{-j\frac{2\pi}{N} k}} \right] = \frac{1}{N} \frac{\sin\left[\frac{2\pi}{N} k \left(L + \frac{1}{2}\right)\right]}{\sin\left(\frac{2\pi}{N} k \frac{1}{2}\right)}, \tag{4.78}$$

where we have used the identity

$$(1 - e^{-j\theta}) = e^{-j\theta/2}(e^{j\theta/2} - e^{-j\theta/2}) = 2je^{-j\theta/2} \sin(\theta/2).$$

The remaining coefficients are obtained by periodic repetition. Therefore,

$$c_k = \begin{cases} \frac{2L+1}{N}, & k = 0, \pm N, \pm 2N, \ldots \\ \frac{1}{N} \frac{\sin\left[\frac{2\pi}{N} k \left(L + \frac{1}{2}\right)\right]}{\sin\left(\frac{2\pi}{N} k \frac{1}{2}\right)}. & \text{otherwise} \end{cases} \tag{4.79}$$

The amplitude spectrum of $x[n]$ is given in Figure 4.20(b) for $L = 2$ and $N = 10$. ∎

Dirichlet's function It is convenient to define the function

$$D_L(\omega) = \frac{\sin(\omega L/2)}{L \sin(\omega/2)}, \tag{4.80}$$
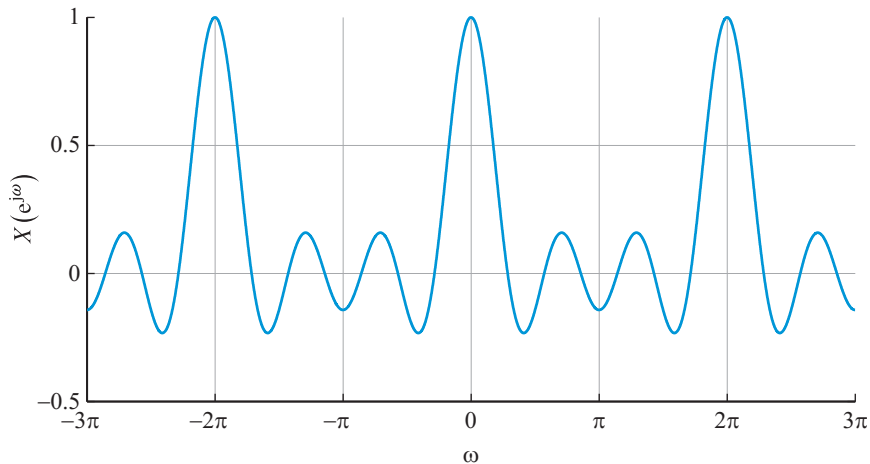
Figure 4.21 The Dirichlet or "digital sinc" function (4.80) for $L = 7$.

which is sometimes known as *Dirichlet's function*. The Dirichlet function (4.80) is the discrete-time counterpart of the sinc function (4.31); in MATLAB it can be evaluated by the function x=diric(omega,L). The presence of the sine function in the denominator makes Dirichlet's function periodic in ω. Figure 4.21 shows three periods of Dirichlet's function for $L = 7$. Note that $D_L(\omega)$ has period 2π for L odd and period 4π for L even (see Problem 42).

Numerical computation of DTFS The analysis and synthesis equations for the DTFS involve the computation of a finite number of items using finite summations. Therefore, they can be *exactly* evaluated by numerical computation. All other Fourier decompositions can be computed only *approximately* because they involve integrals and infinite summations. This subject, which is very important in many signal processing problems, is discussed in Chapter 7.

In MATLAB the analysis and synthesis formulas of the DTFS are implemented using the following fft and ifft functions

$$c_k = \frac{1}{N}\left(\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn}\right) \Rightarrow \text{c} = \frac{1}{N}*\text{fft(x)} = \text{dtfs(x)}, \qquad (4.81)$$

$$x[n] = N\left(\frac{1}{N}\sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn}\right) \Rightarrow \text{x} = \text{N*ifft(c)} = \text{idtfs(x)}. \qquad (4.82)$$

The functions fft and ifft are computationally efficient implementations of the equations within the parentheses (see Chapter 8). We stress that MATLAB assumes that the vector x includes the first N samples of the sequence $x[n]$, that is, $x[0], x[1], \ldots, x[N-1]$. When c_k or $x[n]$ assume real values, we should use *only* the real part of c or x. The imaginary parts, due to numerical accuracy limitations, are not zero; they take small values in the range of $\pm 10^{-16}$ about zero.

Example 4.9   Use of `fft` and `ifft`
To compute the DTFS of a rectangular pulse train with $L = 2$ and $N = 10$, we run the commands

```
>> x=[1 1 1 0 0 0 0 0 1 1]; N=length(x); c=fft(x)/N
c =
  Columns 1 through 8
    0.5000 0.3236   0 -0.1236 0 0.1000   0 -0.1236
  Columns 9 through 10
  0 0.3236
>> x=ifft(c)*N
x =
  Columns 1 through 8
    1.0000 1.0000 1.0000 -0.0000 -0.0000 0.0000 -0.0000 -0.0000
  Columns 9 through 10
    1.0000 1.0000
```

Note that vectors x and c provide one period of $x[n]$ for $0 \le n \le N - 1$, and one period of c_k for $0 \le k \le N - 1$. More details are discussed in Tutorial Problem 10.   ■

4.3.2   Fourier transforms for discrete-time aperiodic signals

Since an aperiodic sequence can be viewed as a periodic sequence with infinite period, we could obtain its Fourier representation by taking the limit of DTFS as the period increases indefinitely.

Example 4.10
We start with a pictorial illustration using the rectangular pulse train $x[n]$ in Example 4.8 and its DTFS coefficients

$$c_k = \frac{1}{N} \frac{\sin\left[\frac{2\pi}{N} k \left(L + \frac{1}{2}\right)\right]}{\sin\left(\frac{2\pi}{N} k \frac{1}{2}\right)}. \tag{4.83}$$

We keep the width $2L + 1 = 5$ of the pulse fixed and we plot the scaled coefficients Nc_k as a function of frequency $\omega_k = (2\pi/N)k$ for $N = 10, 20, 40$. We note that the spacing $\Delta\omega = (2\pi/N)$ of the spectral lines decreases as the period N increases. Eventually as $N \to \infty$, $x[n]$ becomes an aperiodic sequence and its Fourier representation becomes a continuous function of ω. This limiting process is illustrated in Figure 4.22.   ■

From Fourier series to Fourier transform   Consider a finite duration sequence $x[n]$, such that $x[n] = 0$ outside the range $-L_1 \le n \le L_2$ (see Figure 4.23(a)). From this aperiodic
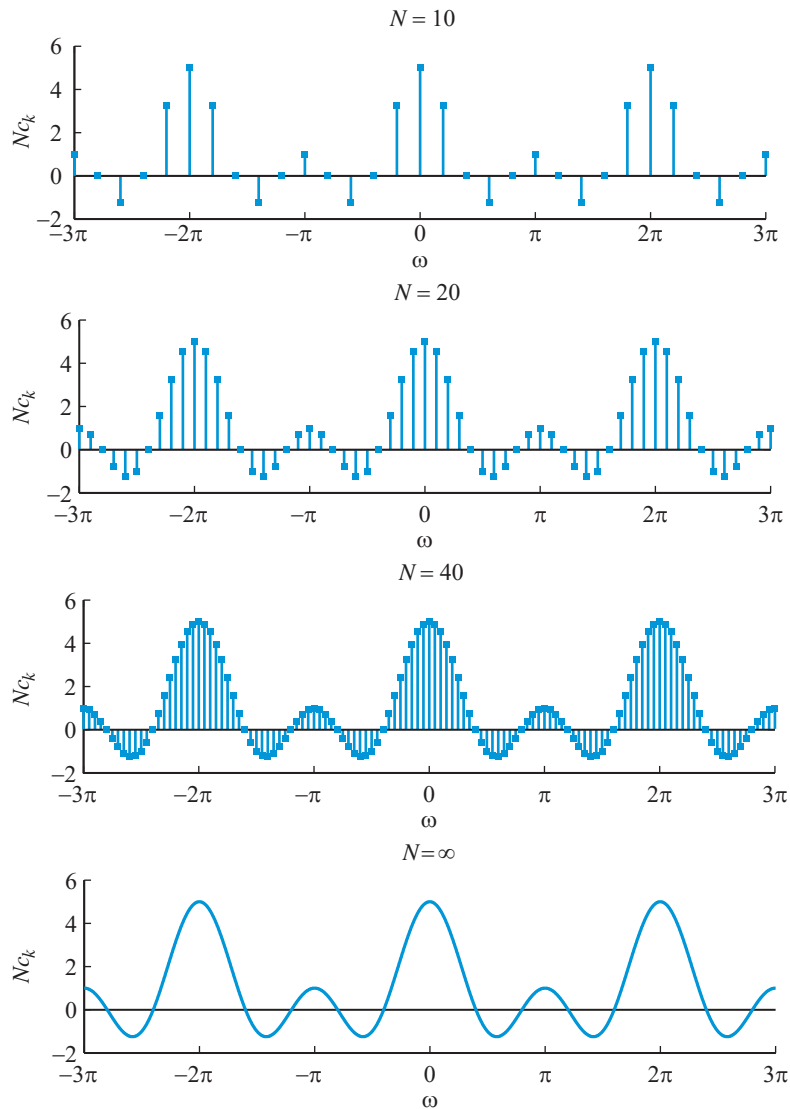
Figure 4.22 How the DTFS converges to the DTFT as the period N of a fixed-width $(2L + 1 = 5$ samples) rectangular pulse tends to infinity.

signal, we construct a periodic signal $x_p[n]$ by repeating $x[n]$ every $N > L_2 + L_1 + 1$ samples as shown in Figure 4.23(b). The DTFS of $x_p[n]$ is given by

$$x_p[n] = \sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn}, \tag{4.84}$$

$$c_k = \frac{1}{N} \sum_{n=0}^{N-1} x_p[n] e^{-j\frac{2\pi}{N}kn}. \tag{4.85}$$
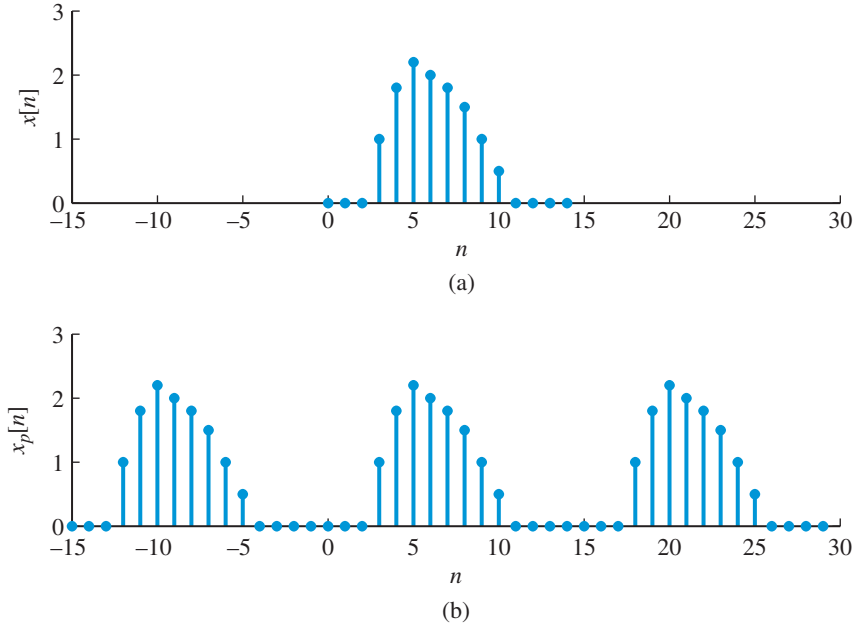
Figure 4.23 Finite duration ($L = 8$) sequence $x[n]$ (a) and its periodic extension $x_p[n]$ (b) obtained by replicating $x[n]$ every $N = 15$ samples.

Recall that the limits in (4.84) and (4.85) can be chosen over any interval covering one period. Since $x_p[n] = x[n]$ for $-L_1 \leq n \leq L_2$, we can express (4.85) as

$$c_k = \frac{1}{N} \sum_{n=-\infty}^{\infty} x[n] e^{-j\frac{2\pi}{N}kn}. \tag{4.86}$$

If we define the "envelope" function as

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}, \tag{4.87}$$

we see that the Fourier series coefficients c_k can be obtained by taking equidistant samples of $X(e^{j\omega})$ as follows:

$$c_k = \frac{1}{N} X(e^{j\omega}) \Big|_{\omega = k\omega_0}, \tag{4.88}$$

where $\omega_0 = 2\pi/N = \Delta\omega$ is the spacing between successive spectral samples. Using (4.86), (4.88), and $1/N = \Delta\omega/(2\pi)$, we obtain

$$x_p[n] = \frac{1}{2\pi} \sum_{k=0}^{N-1} X(e^{jk\Delta\omega}) e^{j(k\Delta\omega)n} \Delta\omega. \tag{4.89}$$

Recall that, as $N \to \infty$, $x_p[n] = x[n]$ for any finite n. Furthermore, as $N \to \infty$, $\Delta\omega \to 0$, and the summation in (4.89) passes to the integral of $X(e^{j\omega})e^{j\omega n}$ over the frequency range from 0 to 2π, since $(2\pi/N)(N-1) \to 2\pi$ as $N \to \infty$. Thus, (4.89) becomes

$$x[n] = \frac{1}{2\pi} \int_0^{2\pi} X(e^{j\omega})e^{j\omega n}d\omega. \tag{4.90}$$

Since $X(e^{j\omega})e^{j\omega n}$ is periodic with period 2π, we can use any interval of integration of length 2π. Equations (4.87) and (4.90) are known as the *Discrete-Time Fourier Transform (DTFT)* pair

Fourier Synthesis Equation   **Fourier Analysis Equation**

$$x[n] = \frac{1}{2\pi} \int_{2\pi} X(e^{j\omega})e^{j\omega n}d\omega \xleftarrow{\text{DTFT}} X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \tag{4.91}$$

The quantities $X(e^{j\omega})$, $|X(e^{j\omega})|$, and $\angle X(e^{j\omega})$ are known as the *spectrum*, *magnitude spectrum*, and *phase spectrum* of the aperiodic sequence $x[n]$, respectively.

Finally, we note that the derivation of DTFT reveals an important relationship between DTFT and DTFS. If $x_p[n] = x_p[n+N]$ and for any n_0,

$$x[n] = \begin{cases} x_p[n], & n_0 \le n \le n_0 + N - 1 \\ 0, & \text{otherwise} \end{cases} \tag{4.92}$$

and the Fourier coefficients of $x_p[n]$ can be expressed in terms of equally spaced samples of the DTFT $X(e^{j\omega})$ of $x[n]$ by

$$c_k = \frac{1}{N}X(e^{j\frac{2\pi}{N}k}). \tag{4.93}$$

Example 4.11 Finite length pulse

Evaluate and plot the magnitude and phase of the DTFT of the sequence

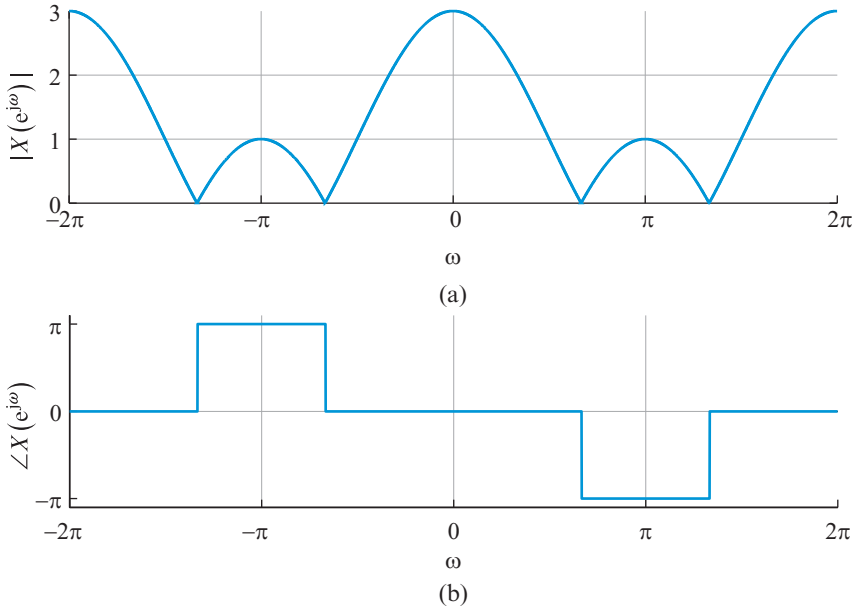$$x[n] = \delta[n+1] + \delta[n] + \delta[n-1].$$

From the definition (4.91), we have

$$X(e^{j\omega}) = \sum_{n=-1}^{1} x[n]e^{-j\omega n} = e^{j\omega} + 1 + e^{-j\omega} = 1 + 2\cos(\omega).$$

Therefore

$$|X(e^{j\omega})| = |1 + 2\cos(\omega)|$$

Figure 4.24 Magnitude (a) and phase (b) of the DTFT of the three-point pulse sequence $x[n] = \delta[n+1] + \delta[n] + \delta[n-1]$.

and

$$\angle X(\mathrm{e}^{\mathrm{j}\omega}) = \begin{cases} 0, & X(\mathrm{e}^{\mathrm{j}\omega}) > 0 \\ \pi. & X(\mathrm{e}^{\mathrm{j}\omega}) < 0 \end{cases}$$

The function $X(\mathrm{e}^{\mathrm{j}\omega})$ changes sign when $1 + 2\cos(\omega) = 0$ or at $\omega = 2\pi/3$ and $\omega = 4\pi/3$. The magnitude and phase plots of $X(\mathrm{e}^{\mathrm{j}\omega})$ are shown in Figure 4.24. ■

Parseval's relation If $x[n]$ has finite energy, we have the following Parseval's relation:

$$E_x = \sum_{n=-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi} \int_{2\pi} |X(\mathrm{e}^{\mathrm{j}\omega})|^2 \mathrm{d}\omega, \tag{4.94}$$

which allows us to determine the signal energy in the time-domain from $x[n]$ or in the frequency-domain from $X(\mathrm{e}^{\mathrm{j}\omega})$. If we consider a small band $\Delta\omega$, centered at $\omega = \omega_0$, then the energy of the frequency components in this $\Delta\omega$ band is given by

$$\frac{|X(\mathrm{e}^{\mathrm{j}\omega_0})|^2}{2\pi} \Delta\omega = |X(\mathrm{e}^{\mathrm{j}2\pi f_0})|^2 \Delta f,$$

that is, the area of $|X(\mathrm{e}^{\mathrm{j}\omega})|^2/(2\pi)$ under the $\Delta\omega$ band. Therefore, $|X(\mathrm{e}^{\mathrm{j}\omega})|^2/(2\pi)$ or $|X(\mathrm{e}^{\mathrm{j}2\pi f})|^2$ are known as the *energy density spectrum* of $x[n]$.

Convergence conditions Although the DTFT pair was derived for a finite duration signal, the formulas hold for infinite duration signals as long as the infinite summation of the analysis equation converges. If we consider the partial approximation

$$X_M(e^{j\omega}) = \sum_{n=-M}^{M} x[n]e^{-j\omega n}, \tag{4.95}$$

the condition for uniform convergence is

$$\sum_{n=-\infty}^{\infty} |x[n]| < \infty \Rightarrow \lim_{M \to \infty} X_M(e^{j\omega}) = X(e^{j\omega}), \tag{4.96}$$

whereas the condition for mean square convergence is

$$\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty \Rightarrow \lim_{M \to \infty} \int_{2\pi} \left| X(e^{j\omega}) - X_M(e^{j\omega}) \right|^2 d\omega = 0. \tag{4.97}$$

More details will be provided when we discuss the approximation problem for filter design in Chapter 10.

Numerical computation of DTFT If $x[n]$ has finite duration, we can compute $X(e^{j\omega})$ at any frequency ω_k using the finite summation

$$X(e^{j\omega_k}) = \sum_{n=N_1}^{N_2} x[n]e^{-j\omega_k n}. \tag{4.98}$$

The exact computation of the inverse DTFT (4.90) is not possible because it requires integration. MATLAB provides the function

$$\texttt{X=freqz(x,1,om)}, \tag{4.99}$$

which computes (4.98) for $N_1 = 0$, $N_2 = N$, and K frequencies ω_k for $0 \le k \le K - 1$.

When $X(e^{j\omega})$ or $x[n]$ assume real values, we should use *only* the real part of X or x. The imaginary parts, due to numerical accuracy limitations, are not zero; they take small values in the range $\pm 10^{-16}$.

To compute (4.98) for arbitrary N_1 and N_2, we first note that changing the index of summation from n to $m = n - N_1$ yields

$$X(e^{j\omega_k}) = \sum_{n=N_1}^{N_2} x[n]e^{-j\omega_k n} = e^{-j\omega_k N_1} \sum_{n=0}^{N_2-N_1} x[n+N_1]e^{-j\omega_k n}. \tag{4.100}$$

The computation of (4.100) can be done using the following MATLAB function:

```
function X=dtft12(x,Nstart,om)
x=freqz(x,1,om); X=exp(-j*om*Nstart).*X;
```
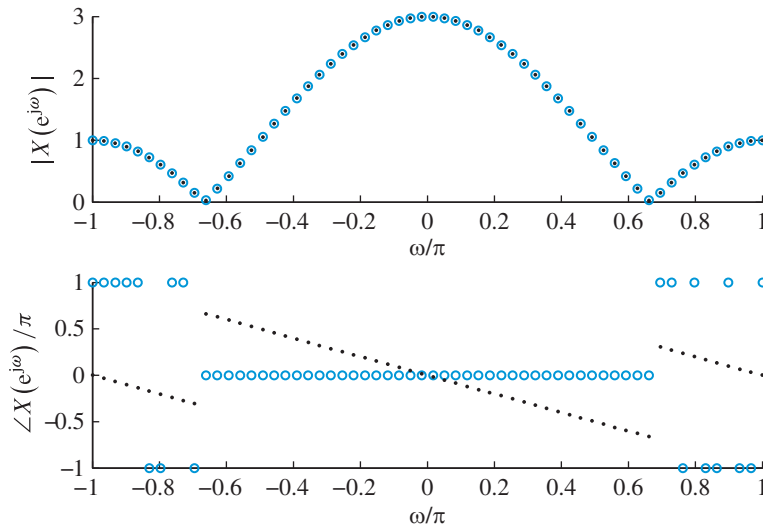
Figure 4.25 Computation of the DTFT of $x[n] = \delta[n+1] + \delta[n] + \delta[n-1]$ using MATLAB functions freqz ($\cdot$) and dtft12 ($\circ$), respectively.

where x is the vector containing the signal samples and Nstart is the index of the first sample.
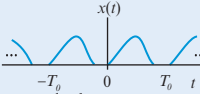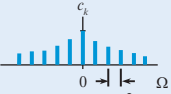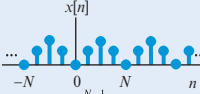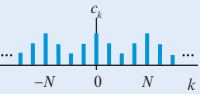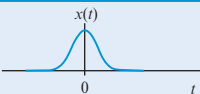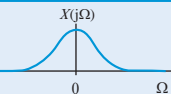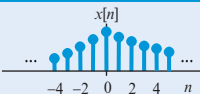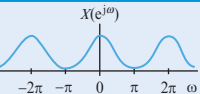
To appreciate the difference between freqz and dtft12, suppose that we wish to compute the DTFT of the noncausal finite length sequence $x[n]$ in Example 4.11. Since the first nonzero sample is at $n = -1$, we should use dtft12. If we run script (freqzdtft12.m) and plot the resulting transforms, we obtain Figure 4.25.

```
% Script file: freqzdtft12.m
x=[1 1 1]; % n=-1,0,1
om=linspace(-pi,pi,60); X=dtft12(x,-1,om);
X1=freqz(x,1,om);
```

We note that the Fourier transforms obtained from the two functions have the same magnitude but different phase. This happens because freqz always assumes that $N_1 = 0$. In addition, if we compare with Figure 4.24(b), we note some irregularities when the phase takes the values $\pm\pi$. This results from the way MATLAB evaluates the inverse tangent function. The user should assign the values properly using the odd symmetry of the Fourier transform phase.

4.4 Summary of Fourier series and Fourier transforms

The principle of Fourier representation of signals is to break up all signals into summations of sinusoidal or complex exponential components. The analytical, numerical, or graphical representations of magnitude and phase of each component as a function of frequency

Table 4.1 Summary of Fourier representation of signals.

		Continuous-time signals		Discrete-time signals	
		Time-domain	Frequency-domain	Time-domain	Frequency-domain
Periodic signals	Fourier series	$x(t)$ $-T_0 \quad 0 \quad T_0 \quad t$ $c_k = \dfrac{1}{T_0}\displaystyle\int_{T_0} x(t)\,e^{-jk\Omega_0 t}dt$ $\xrightarrow{\text{CTFS}}$	c_k $0 \rightarrow\!\!\mid\leftarrow \Omega$ $\Omega_0 = \dfrac{2\pi}{T_0}$ $\xleftarrow{\text{ICTFS}}$ $x(t) = \displaystyle\sum_{k=-\infty}^{\infty} c_k e^{jk\Omega_0 t}$	$x[n]$ $-N \quad 0 \quad N \quad n$ $c_k = \dfrac{1}{N}\displaystyle\sum_{n=0}^{N-1} x[n]\,e^{-j\frac{2\pi}{N}kn}$ $\xrightarrow{\text{DTFS}}$	c_k $-N \quad 0 \quad N \quad k$ $\xleftarrow{\text{IDTFS}}$ $x[n] = \displaystyle\sum_{k=0}^{N-1} c_k e^{j\frac{2\pi}{N}kn}$
		Continuous and periodic	Discrete and aperiodic	Discrete and periodic	Discrete and periodic
Aperiodic signals	Fourier transforms	$x(t)$ $0 \qquad t$ $X(j\Omega) = \displaystyle\int_{-\infty}^{\infty} x(t)\,e^{-j\Omega t}dt$ $\xrightarrow{\text{CTFT}}$	$X(j\Omega)$ $0 \qquad \Omega$ $\xleftarrow{\text{ICTFT}}$ $x(t) = \dfrac{1}{2\pi}\displaystyle\int_{-\infty}^{\infty} X(j\Omega)\,e^{j\Omega t}d\Omega$	$x[n]$ $-4\ -2\ 0\ 2\ 4\quad n$ $X(e^{j\omega}) = \displaystyle\sum_{n=-\infty}^{\infty} x[n]\,e^{-j\omega n}$ $\xrightarrow{\text{DTFT}}$	$X(e^{j\omega})$ $-2\pi\ -\pi\ 0\ \pi\ 2\pi\ \omega$ $\xleftarrow{\text{IDTFT}}$ $x[n] = \dfrac{1}{2\pi}\displaystyle\int_{2\pi} X(e^{j\omega})\,e^{j\omega n}d\omega$
		Continuous and aperiodic	Continuous and aperiodic	Discrete and aperiodic	Continuous and periodic

are known as the magnitude spectrum and phase spectrum of the signal. The magnitude and phase spectra are collectively called the Fourier spectrum or simply the spectrum of a signal. The exact form of the mathematical formulas used to determine the spectrum from the signal (Fourier analysis equation) and the signal from its spectrum (Fourier synthesis equation) depend on whether the time is continuous or discrete and whether the signal is periodic or aperiodic. This leads to the four different Fourier representations summarized in Table 4.1. Careful inspection of this table leads to the following conclusions:

- Continuous-time periodic signals are represented by an infinite Fourier series of harmonically related complex exponentials. Therefore, the spectrum exists only at $F = 0, \pm F_0, \pm 2F_0, \ldots$, that is, at discrete values of F. The spacing between the lines of this discrete or line spectrum is $F_0 = 1/T_0$, that is the reciprocal of the fundamental period.
- Continuous-time aperiodic signals are represented by a Fourier integral of complex exponentials over the entire frequency axis. Therefore, the spectrum exists for all F, $-\infty < F < \infty$. Knowledge of $X(j2\pi F)$ for $-\infty < F < \infty$ is needed to represent $x(t)$ for $-\infty < t < \infty$.
- Discrete-time periodic signals are represented by a finite Fourier series of harmonically related complex exponentials. The spacing between the lines of the resulting discrete spectrum is $\Delta\omega = 2\pi/N$, where N is the fundamental period. The DTFS coefficients of a periodic signal are periodic and the analysis equation involves a finite sum over a range of 2π.
- Discrete-time aperiodic signals are represented by a Fourier integral of complex exponentials over any frequency range of length 2π radians. Knowledge of the periodic DTFT function $X(e^{j\omega})$ over any interval of length 2π is needed to recover $x[n]$ for $-\infty < n < \infty$.

- Discrete-time complex exponentials that differ in frequency by a multiple of 2π are identical. This has the following implications:
  - Low-frequencies, corresponding to slowly-varying signals, are around the points $\omega = 0, \pm 2\pi, \pm 4\pi, \dots$
  - High-frequencies, corresponding to rapidly-varying signals, are around the points $\omega = \pm\pi, \pm 3\pi, \dots$

The key "sampling-periodicity" features of the Fourier representations in Table 4.1 can be summarized by the following rule, which is further discussed in Chapter 6.

Result 4.4.1 Periodicity with "period" α in one domain implies discretization with "spacing" of $1/\alpha$ in the other domain, and vice versa.

For aperiodic signals, the area under the curve $|X(\mathrm{j}2\pi F)|^2$, $-\infty < F < \infty$ or $|X(\mathrm{e}^{\mathrm{j}\omega})|^2/2\pi$, $0 \le \omega < 2\pi$, is equal to the total energy of $x(t)$ or $x[n]$. The contribution of a given frequency band may be found by integrating the desired area. Each point on the $X(\mathrm{j}2\pi F)$ or $X(\mathrm{e}^{\mathrm{j}\omega})$ curves contributes nothing to the total energy; only an area under a finite band can contribute. This justifies the term energy spectrum density for a continuous Fourier spectrum.

In contrast, periodic signals have all their frequency components at discrete frequencies. Each of these discrete frequencies contributes to the total power of the signal. However, there is no contribution from frequencies between the lines. The power of continuous-time signals is distributed to an infinite number of spectral lines, whereas the power of discrete-time signals is distributed to a finite number of N spectral lines.

Bandlimited signals Signals whose frequency components are zero or "small" outside a finite interval $0 \le B_1 \le |F| \le B_2 < \infty$ are said to be *bandlimited*. The quantity $B = B_2 - B_1$ is known as the *bandwidth* of the signal. For discrete-time signals we should also have the condition $B_2 < F_s/2$. Depending on the values of B_1 and B_2, we distinguish the following types of signal:

Type	Continuous-time	Discrete-time
Lowpass	$0 \le |F| \le B < \infty$	$0 \le |F| \le B < F_s/2$
Bandpass	$0 < B_1 \le |F| \le B_2 < \infty$	$0 < B_1 \le |F| \le B_2 < F_s/2$
Highpass	$0 < B \le |F|$	$0 < B \le |F| \le F_s/2$

4.5 Properties of the discrete-time Fourier transform

The various properties of the DTFT are used to simplify the solution of problems and sometimes to check the validity of a solution. When a signal has a symmetry property in the time domain, this property imposes another unique symmetry on its DTFT. Furthermore,

some operations on a signal or between two signals result in different operations between their DTFTs.

Relationship to the z-transform and periodicity

The z-transform of a sequence $x[n]$ was defined in Section 3.2 by

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n}. \tag{4.101}$$

If the ROC of $X(z)$ includes the unit circle, defined by $z = \mathrm{e}^{\mathrm{j}\omega}$ or equivalently $|z| = 1$, we obtain

$$X(z)|_{z=\mathrm{e}^{\mathrm{j}\omega}} = \sum_{n=-\infty}^{\infty} x[n]\mathrm{e}^{-\mathrm{j}\omega n} = X(\mathrm{e}^{\mathrm{j}\omega}), \tag{4.102}$$

that is, the z-transform reduces to the Fourier transform. The magnitude of DTFT is obtained by intersecting the surface $|H(z)|$ with a vertical cylinder of radius one, centered at $z = 0$. This is illustrated in Figure 4.26, which provides a clear demonstration of the periodicity of DTFT. The radiant frequency ω is measured with respect to the positive real axis and the unit circle is mapped on the linear frequency axis as shown in Figure 4.26. Multiple rotations around the unit circle create an inherent periodicity, with period 2π
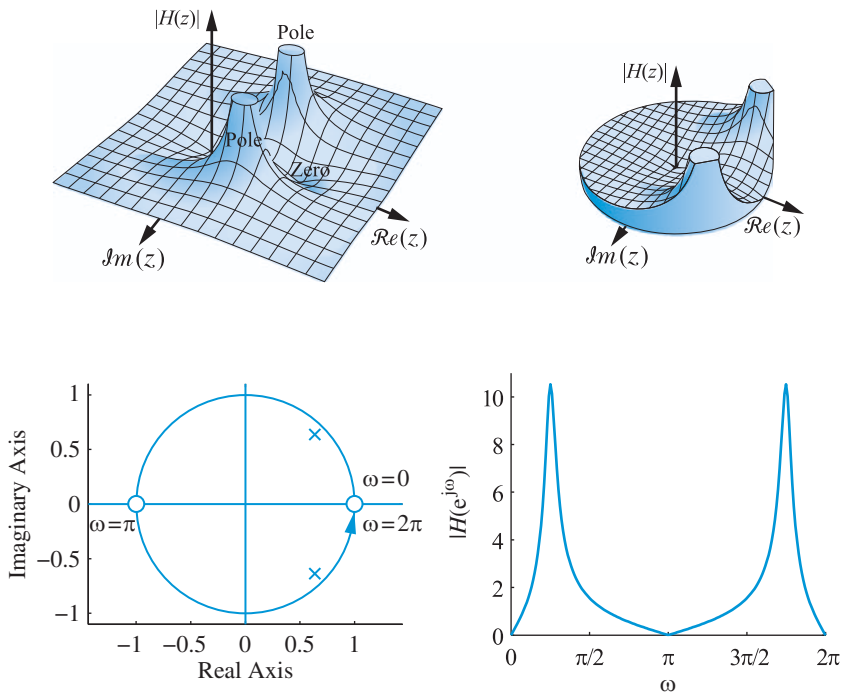
Figure 4.26 The relationship between the z-transform and the DTFT for a sequence with two complex-conjugate poles at $z = 0.9\mathrm{e}^{\mathrm{j}\pm\pi/4}$ and two zeros at $z = \pm 1$.

radians, in the frequency domain. The details of generating Figure 4.26 with MATLAB are discussed in Problem 47.

If we express z in polar form as $z = re^{j\omega}$, the z-transform can be written as

$$X(re^{j\omega}) = \sum_{n=-\infty}^{\infty} \left(x[n]r^{-n} \right) e^{-j\omega n}, \tag{4.103}$$

which shows that the z-transform of $x[n]$ is equal to the DTFT of the exponentially weighted sequence $r^{-n}x[n]$. Due to the exponential weighting, it is possible for the z-transform to exist even if the DTFT does not exist. However, we note that there are sequences that have a DTFT but not a z-transform (for example, an ideal lowpass sequence in Figure 4.28), and vice versa.

4.5.2   Symmetry properties

Suppose that both the signal $x[n]$ and its DTFT $X(e^{j\omega})$ are complex-valued functions. Then, we can express them in rectangular form as follows

$$x[n] = x_R[n] + jx_I[n], \tag{4.104}$$

$$X(e^{j\omega}) = X_R(e^{j\omega}) + jX_I(e^{j\omega}). \tag{4.105}$$

We next substitute (4.104) and $e^{-j\omega} = \cos\omega - j\sin\omega$ into (4.87) and separate real and imaginary parts. The result is

$$X_R(e^{j\omega}) = \sum_{n=-\infty}^{\infty} \{x_R[n]\cos(\omega n) + x_I[n]\sin(\omega)n\}, \tag{4.106}$$

$$X_I(e^{j\omega}) = -\sum_{n=-\infty}^{\infty} \{x_R[n]\sin(\omega n) - x_I[n]\cos(\omega n)\}. \tag{4.107}$$

If we substitute (4.105) and $e^{j\omega} = \cos\omega + j\sin\omega$ into (4.90) and separate real and imaginary parts, we obtain

$$x_R[n] = \frac{1}{2\pi} \int_{2\pi} \left[X_R(e^{j\omega})\cos(\omega n) - X_I(e^{j\omega})\sin(\omega n) \right] d\omega, \tag{4.108}$$

$$x_I[n] = \frac{1}{2\pi} \int_{2\pi} \left[X_R(e^{j\omega})\sin(\omega n) + X_I(e^{j\omega})\cos(\omega n) \right] d\omega. \tag{4.109}$$

We now discuss the special cases of real and imaginary signals.

Real signals If $x[n]$ is real, then $x_R[n] = x[n]$ and $x_I[n] = 0$. In this case, (4.106) and (4.107) are simplified to

$$X_R(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\cos(\omega n) \quad \text{and} \quad X_I(e^{j\omega}) = -\sum_{n=-\infty}^{\infty} x[n]\sin(\omega n). \tag{4.110}$$

Since $\cos(-\omega n) = \cos(\omega n)$ and $\sin(-\omega n) = -\sin(\omega n)$, we can easily see that

$$X_R(e^{-j\omega}) = X_R(e^{j\omega}), \qquad \text{(even symmetry)} \qquad (4.111)$$

$$X_I(e^{-j\omega}) = -X_I(e^{j\omega}), \qquad \text{(odd symmetry)} \qquad (4.112)$$

or by combining into a single equation

$$X^*(e^{j\omega}) = X(e^{-j\omega}). \qquad \text{(Hermitian symmetry)} \qquad (4.113)$$

Thus, the DTFT of a real signal has Hermitian (or complex-conjugate) symmetry.

The magnitude and phase of the DTFT are given by

$$|X(e^{j\omega})| = \sqrt{X_R^2(e^{j\omega}) + X_I^2(e^{j\omega})}, \qquad (4.114)$$

$$\angle X(e^{j\omega}) = \tan^{-1}\frac{X_I(e^{j\omega})}{X_R(e^{j\omega})}. \qquad (4.115)$$

Using the symmetry properties (4.111) and (4.112), we obtain the following symmetry properties for the magnitude and phase spectra

$$|X(e^{-j\omega})| = |X(e^{j\omega})|, \qquad \text{(even symmetry)} \qquad (4.116)$$

$$\angle X(e^{-j\omega}) = -\angle X(e^{j\omega}). \qquad \text{(odd symmetry)} \qquad (4.117)$$

The inverse DTFT for real signals is given by (4.108) by replacing $x_R[n]$ by $x[n]$

$$x[n] = \frac{1}{2\pi}\int_{2\pi}\left[X_R(e^{j\omega})\cos(\omega n) - X_I(e^{j\omega})\sin(\omega n)\right]d\omega. \qquad (4.118)$$

Since $X_R(e^{j\omega})\cos\omega n$ and $X_I(e^{j\omega})\sin\omega n$ are even functions of ω, we have

$$x[n] = \frac{1}{\pi}\int_0^{\pi}\left[X_R(e^{j\omega})\cos(\omega n) - X_I(e^{j\omega})\sin(\omega n)\right]d\omega, \qquad (4.119)$$

which requires integration over half the fundamental frequency range.

Real and even signals If $x[n]$ is real and even, that is, $x[-n] = x[n]$, then $x[n]\cos\omega n$ is an even and $x[n]\sin\omega n$ is an odd function of n. Therefore, from (4.110) and (4.119) we obtain

$$X_R(e^{j\omega}) = x[0] + 2\sum_{n=1}^{\infty}x[n]\cos(\omega n), \qquad \text{(even symmetry)} \qquad (4.120)$$

$$X_I(e^{j\omega}) = 0, \qquad (4.121)$$

$$x[n] = \frac{1}{\pi}\int_0^{\pi}X_R(e^{j\omega})\cos(\omega n)\,d\omega. \qquad \text{(even symmetry)} \qquad (4.122)$$

Table 4.2 Special cases of the DTFT for real signals.

Signal	Fourier transform
Real and even	real and even
Real and odd	imaginary and odd
Imaginary and even	imaginary and even
Imaginary and odd	real and odd

Thus, real signals with even symmetry have real spectra with even symmetry. These four special cases are summarized in Table 4.2.

Real and odd signals If $x[n]$ is real and odd, that is, $x[-n] = -x[n]$, then $x[n]\cos\omega n$ is an odd and $x[n]\sin\omega n$ is an even function of n. Therefore, from (4.110) and (4.119) we obtain

$$X_R(e^{j\omega}) = 0, \tag{4.123}$$

$$X_I(e^{j\omega}) = -2\sum_{n=1}^{\infty} x[n]\sin(\omega n), \qquad \text{(odd symmetry)} \tag{4.124}$$

$$x[n] = -\frac{1}{\pi}\int_0^{\pi} X_I(e^{j\omega})\sin(\omega n)\, d\omega. \qquad \text{(odd symmetry)} \tag{4.125}$$

Thus, real signals with odd symmetry have purely imaginary spectra with odd symmetry.

The symmetry properties of the DTFT are summarized in Table 4.3. We shall illustrate these properties with some examples.

Example 4.12   Causal exponential sequence
Consider the sequence $x[n] = a^n u[n]$. For $|a| < 1$, the sequence is absolutely summable, that is

$$\sum_{n=0}^{\infty} |a|^n = \frac{1}{1-|a|} < \infty. \tag{4.126}$$

Therefore, the DTFT exists and is given by

$$X(e^{j\omega}) = \sum_{n=0}^{\infty} a^n e^{-j\omega n} = \sum_{n=0}^{\infty} (ae^{-j\omega})^n$$

$$= \frac{1}{1-ae^{-j\omega}}. \quad \text{if } |ae^{-j\omega}| < 1 \text{ or } |a| < 1 \tag{4.127}$$

Table 4.3 Symmetry properties of the DTFT.

Sequence $x[n]$	Transform $X(e^{j\omega})$
	Complex signals
$x^*[n]$	$X^*(e^{-j\omega})$
$x^*[-n]$	$X^*(e^{j\omega})$
$x_R[n]$	$X_e(e^{j\omega}) \triangleq \frac{1}{2}\left[X(e^{j\omega}) + X^*(e^{-j\omega})\right]$
$jx_I[n]$	$X_o(e^{j\omega}) \triangleq \frac{1}{2}\left[X(e^{j\omega}) - X^*(e^{-j\omega})\right]$
$x_e[n] \triangleq \frac{1}{2}(x[n] + x^*[-n])$	$X_R(e^{j\omega})$
$x_o[n] \triangleq \frac{1}{2}(x[n] - x^*[-n])$	$jX_I(e^{j\omega})$
	Real signals
Any real $x[n]$	$X(e^{j\omega}) = X^*(e^{-j\omega})$
	$X_R(e^{j\omega}) = X_R(e^{-j\omega})$
	$X_I(e^{j\omega}) = -X_I(e^{-j\omega})$
	$\|X(e^{j\omega})\| = \|X(e^{-j\omega})\|$
	$\angle X(e^{j\omega}) = -\angle X(e^{-j\omega})$
$x_e[n] = \frac{1}{2}(x[n] + x[-n])$	$X_R(e^{j\omega})$
Even part of $x[n]$	real part of $X(e^{j\omega})$ (even)
$x_o[n] = \frac{1}{2}(x[n] - x[-n])$	$jX_I(e^{j\omega})$
Odd part of $x[n]$	imaginary part of $X(e^{j\omega})$ (odd)

If $x[n]$ is real ($-1 < a < 1$), using the properties of complex numbers, we obtain

$$X_R(e^{j\omega}) = \frac{1 - a\cos(\omega)}{1 - 2a\cos(\omega) + a^2} = X_R(e^{-j\omega}), \qquad \text{(even)} \qquad (4.128a)$$

$$X_I(e^{j\omega}) = \frac{-a\sin(\omega)}{1 - 2a\cos(\omega) + a^2} = -X_I(e^{-j\omega}), \qquad \text{(odd)} \qquad (4.128b)$$

$$\left|X(e^{j\omega})\right| = \frac{1}{\sqrt{1 - 2a\cos(\omega) + a^2}} = \left|X(e^{-j\omega})\right|, \qquad \text{(even)} \qquad (4.128c)$$

$$\angle X(e^{j\omega}) = \tan^{-1}\frac{-a\sin(\omega)}{1 - a\cos(\omega)} = -\angle X(e^{-j\omega}). \qquad \text{(odd)} \qquad (4.128d)$$

These functions are plotted in Figure 4.27 for a lowpass sequence ($0 < a < 1$) and a highpass sequence ($-1 < a < 0$). ■
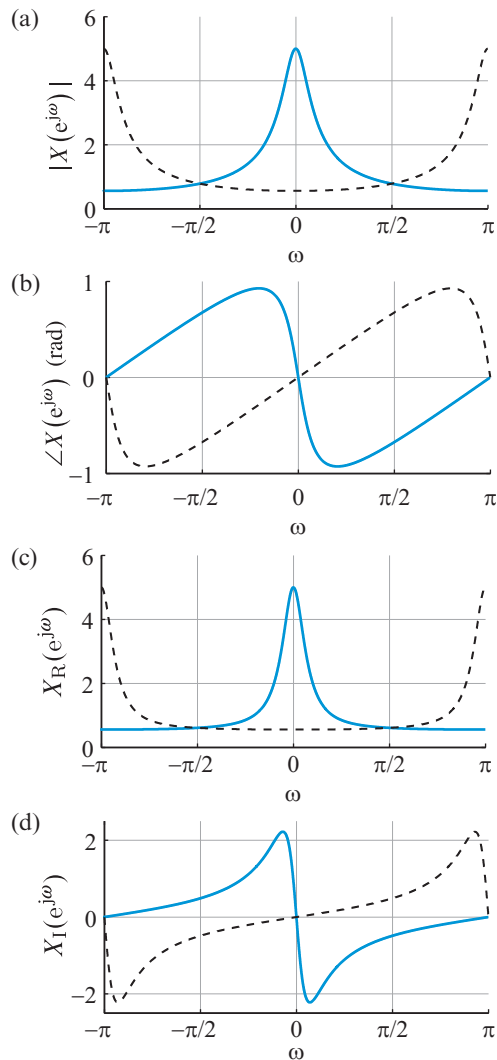
Figure 4.27 Plots of the magnitude (a), phase (b), real part (c), and imaginary part (d) of the DTFT for the sequence $x[n] = a^n u[n]$. The solid lines correspond to a lowpass sequence ($a = 0.8$) and the dashed lines to a highpass sequence ($a = -0.8$).

Example 4.13   Ideal lowpass sequence

Consider a sequence $x[n]$ with DTFT defined, over one period, by

$$X(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0. & \omega_c < |\omega| < \pi \end{cases} \tag{4.129}$$
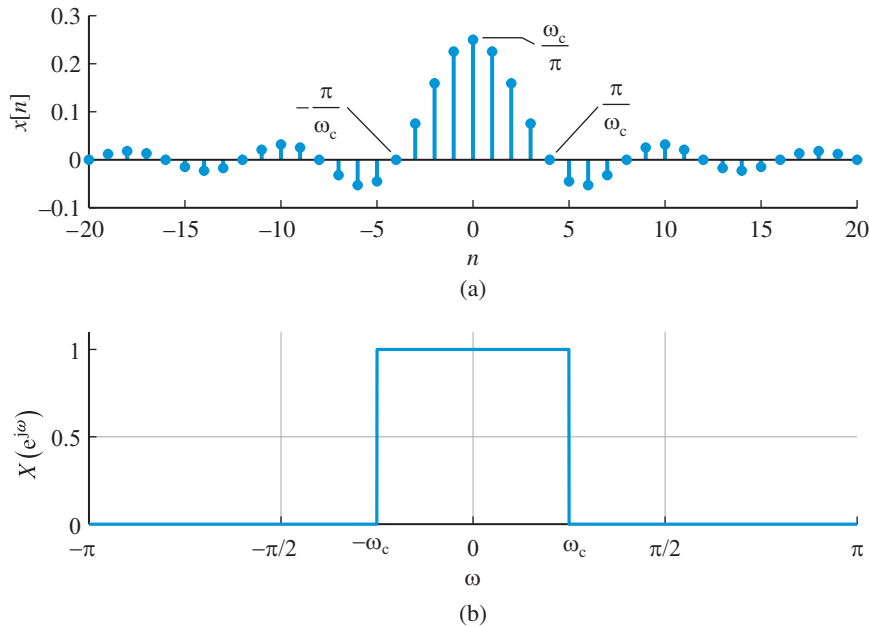
Figure 4.28 The "sinc" sequence (a) and its Fourier transform (b).

The sequence $x[n]$ can be obtained using the synthesis formula (4.88)

$$x[n] = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} e^{j\omega n} d\omega = \frac{1}{2\pi jn} e^{j\omega n}\Big|_{-\omega_c}^{\omega_c} = \frac{\sin(\omega_c n)}{\pi n}, \quad n \neq 0 \qquad (4.130)$$

For $n = 0$ we obtain $x[0] = 0/0$, which is undefined. Since n is integer, we cannot take the limit $n \to 0$ to determine $x[0]$ using l'Hôpital's rule. However, if we use the definition directly, we obtain

$$x[0] = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} d\omega = \frac{\omega_c}{\pi}. \qquad (4.131)$$

For convenience, we usually combine (4.130) and (4.131) into the single equation

$$x[n] = \frac{\omega_c}{\pi} \frac{\sin(\omega_c n)}{\omega_c n} = \frac{\sin(\omega_c n)}{\pi n}, \quad -\infty < n < \infty \qquad (4.132)$$

with the understanding that at $n = 0$, $x[n] = \omega_c/\pi$. As we explained in Section 4.3.2, the DTFT of $x[n]$ exists in the mean square sense. The sequence $x[n]$ and its DTFT $X(e^{j\omega})$ are shown in Figure 4.28. ∎

Example 4.14 Rectangular pulse sequence

Consider the sequence

$$x[n] = \begin{cases} A, & 0 \le n \le L - 1 \\ 0. & \text{otherwise} \end{cases} \tag{4.133}$$

which is illustrated in Figure 4.29(a). Since $x[n]$ is absolutely summable, its Fourier transform exists. Using the geometric summation formula, we obtain
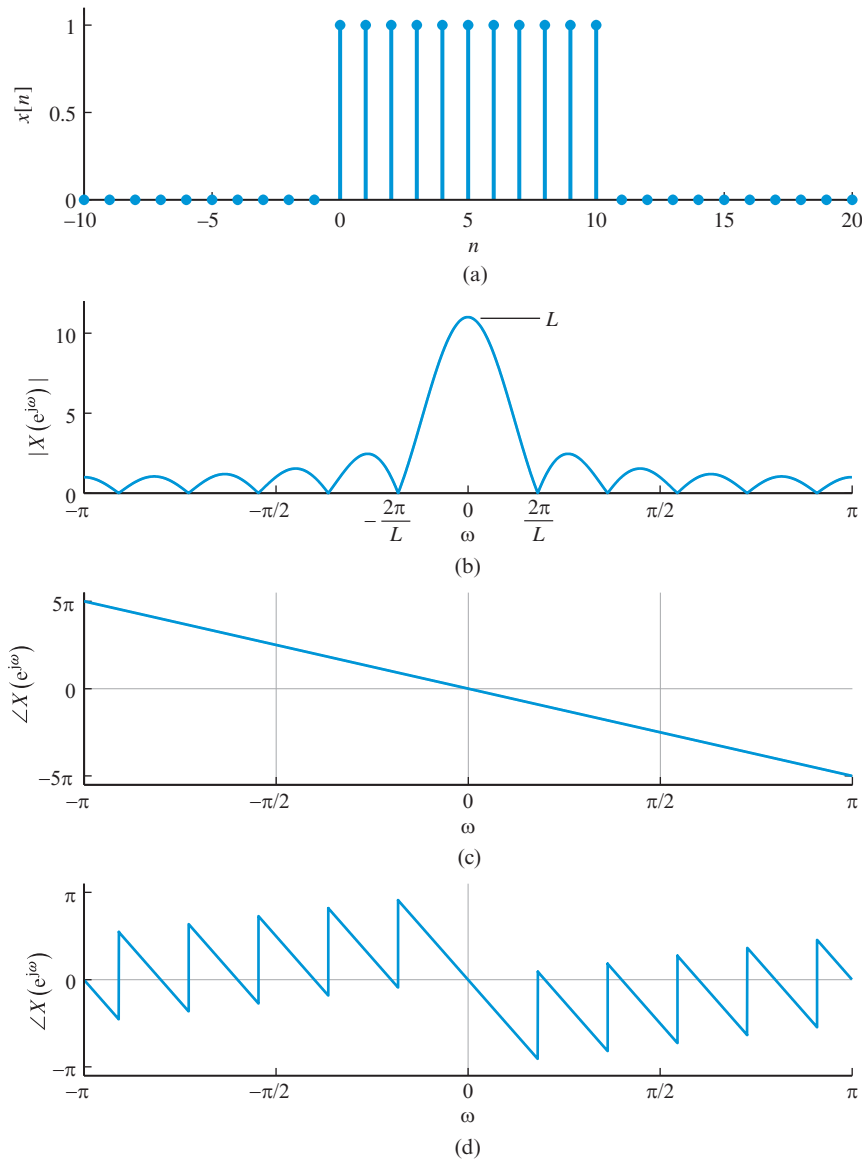
Figure 4.29 The rectangular pulse sequence and its DTFT $X(e^{j\omega})$. (a) Sequence $x[n]$, (b) magnitude $|X(e^{j\omega})|$ from (4.135), and (c) phase $\angle X(e^{j\omega})$ from (4.136). The plot in (d) shows the phase function computed using MATLAB function angle(X).

$$X(e^{j\omega}) = \sum_{n=0}^{L-1} Ae^{-j\omega n} = A\frac{1 - e^{-j\omega L}}{1 - e^{-j\omega}} = A\frac{e^{-j\omega L/2}(e^{j\omega L/2} - e^{-j\omega L/2})}{e^{-j\omega/2}(e^{j\omega/2} - e^{-j\omega/2})}$$

$$= Ae^{-j\omega(L-1)/2}\frac{\sin(\omega L/2)}{\sin(\omega/2)}. \tag{4.134}$$

For $\omega = 0$, (4.134) is indeterminate; however, the definition of DTFT yields $X(0) = AL$. The magnitude and phase of $X(e^{j\omega})$ are given by

$$|X(e^{j\omega})| = |A|\left|\frac{\sin(\omega L/2)}{\sin(\omega/2)}\right|, \tag{4.135}$$

$$\angle X(e^{j\omega}) = \angle A - \frac{\omega}{2}(L-1) + \angle\frac{\sin(\omega L/2)}{\sin(\omega/2)}, \tag{4.136}$$

where we should keep in mind that the phase of a real quantity is zero if the quantity is positive and $\pm\pi$ if the quantity is negative. The sign of π can be chosen in any way that assures the odd symmetry of $\angle X(e^{j\omega})$. This DTFT pair is illustrated in Figure 4.29(b)–(c). ∎

Principal value of angle The principal value of the angle of a complex number is defined to be the angle between $-\pi$ and $+\pi$ radians. The principal angle is typically computed using subroutine `angle`, which in MATLAB is implemented by the function `atan2` as follows

```
function p = angle(h)
p = atan2(imag(h), real(h));
```

The elements of `p` lie in the closed interval `[-pi,pi]`, where `pi` is the MATLAB floating-point representation of π. The `atan2` function uses the signs of real and imaginary parts to determine the specific quadrant of the unit circle where the angle lies.

The effects of this algorithm are illustrated in Figure 4.29(d), where the phase has been computed using the commands `X=freqz(x,1,om)` and `p=angle(X)`. Figure 4.29(c) shows the phase computed from the analytically derived formula (4.136). This continuous curve can be obtained from the curve in Figure 4.29(d) by adding multiples of π when absolute jumps between consecutive elements of `p` are greater than the default jump tolerance of π radians. In MATLAB this can be done using function `q=unwrap(p)`.

4.5.3 Operational properties

We now consider a number of properties of the DTFT. These properties provide additional insight into the relationship between a signal and its transform and can be used to simplify the evaluation of direct and inverse transforms. In practice, typically, it is more efficient to implement these operations in the time-domain. However, in some special cases, frequency-domain implementations are more efficient (see Chapter 7). Since the

4.5 Properties of the discrete-time Fourier transform

DTFT is a special case of the z-transform, all properties of the z-transform translate into similar properties for the Fourier transform.

Linearity The DTFT is a linear operator, that is,

$$a_1 x_1[n] + a_2 x_2[n] \xrightarrow{\text{DTFT}} a_1 X_1(\mathrm{e}^{\mathrm{j}\omega}) + a_2 X_2(\mathrm{e}^{\mathrm{j}\omega}), \tag{4.137}$$

which follows directly from the definition (4.87).

Time shifting If $x[n] \xleftarrow{\text{DTFT}} X(\mathrm{e}^{\mathrm{j}\omega})$, then

$$x[n-k] \xleftarrow{\text{DTFT}} \mathrm{e}^{-\mathrm{j}\omega k} X(\mathrm{e}^{\mathrm{j}\omega}). \tag{4.138}$$

To prove (4.138), we set $y[n] = x[n-k]$, substitute into the definition (4.87), and change the index of summation from n to $m = n - k$. Since

$$\mathcal{F}\{x[n-k]\} = |X(\mathrm{e}^{\mathrm{j}\omega})| \mathrm{e}^{-\mathrm{j}\left[\angle X(\mathrm{e}^{\mathrm{j}\omega}) - k\omega\right]}, \tag{4.139}$$

we see that the magnitude of the spectrum remains the same; only the phase spectrum is changed linearly by the factor $-k\omega$.

Frequency shifting According to this property

$$\mathrm{e}^{\mathrm{j}\omega_c n} x[n] \xleftarrow{\text{DTFT}} X(\mathrm{e}^{\mathrm{j}[\omega - \omega_c]}). \tag{4.140}$$

Indeed, if $y[n] = \mathrm{e}^{\mathrm{j}\omega_c n} x[n]$ we have

$$Y(\mathrm{e}^{\mathrm{j}\omega}) = \sum_{n=-\infty}^{\infty} \mathrm{e}^{\mathrm{j}\omega_c n} x[n] \mathrm{e}^{-\mathrm{j}\omega n} = \sum_{n=-\infty}^{\infty} x[n] \mathrm{e}^{-\mathrm{j}(\omega - \omega_c)n} = X(\mathrm{e}^{\mathrm{j}[\omega - \omega_c]}). \tag{4.141}$$

This property, which is also known as *amplitude modulation with exponential carrier* $c[n] = \mathrm{e}^{\mathrm{j}\omega_c n}$, is illustrated in Figure 4.30. For the spectrum $X\left[\mathrm{e}^{\mathrm{j}(\omega - \omega_c)}\right]$ to remain within the fundamental range, we should have $\omega_c + \omega_m \leq 2\pi$ or $\omega_c \leq 2\pi - \omega_m$. For $\omega_c = \pi$ we have, $y[n] = \mathrm{e}^{\mathrm{j}\pi n} x[n] = (-1)^n x[n]$, which can be achieved by flipping the sign of the odd-indexed samples (modulation by "odd-sample flipping").

Modulation The (amplitude) modulation with a sinusoidal carrier signal, is given by

$$x[n] \cos(\omega_c n) \xleftarrow{\text{DTFT}} \frac{1}{2} X(\mathrm{e}^{\mathrm{j}[\omega + \omega_c]}) + \frac{1}{2} X(\mathrm{e}^{\mathrm{j}[\omega - \omega_c]}). \tag{4.142}$$

This property, which involves double frequency shifting, is illustrated in Figure 4.31. To avoid overlap between low- and high-frequencies, we should have

$$\omega_c - \omega_m > 0 \quad \text{and} \quad (\omega_c + \omega_m) < (2\pi - \omega_c - \omega_m), \tag{4.143}$$
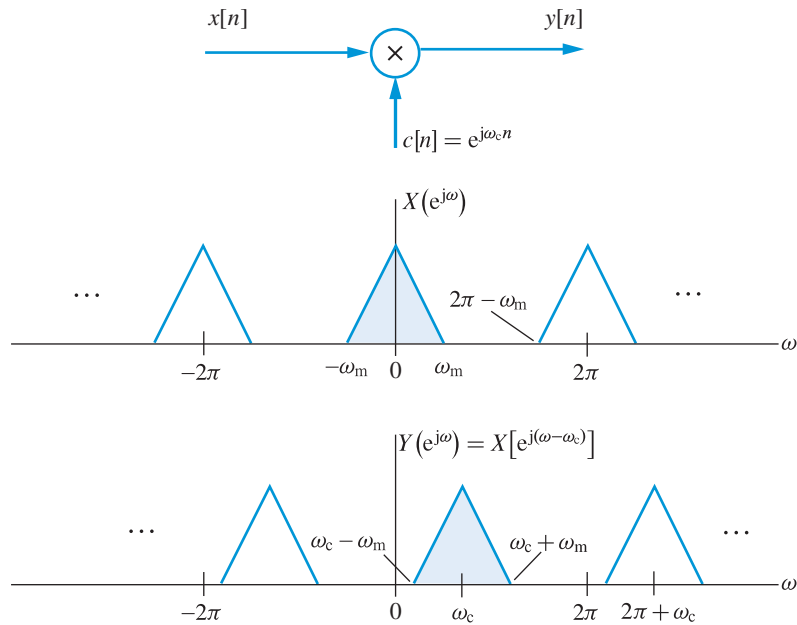
Figure 4.30 The frequency shifting property of the DTFT for $\omega_c > 0$. For $\omega_c < 0$, the spectrum is shifted to the left (at lower frequencies).
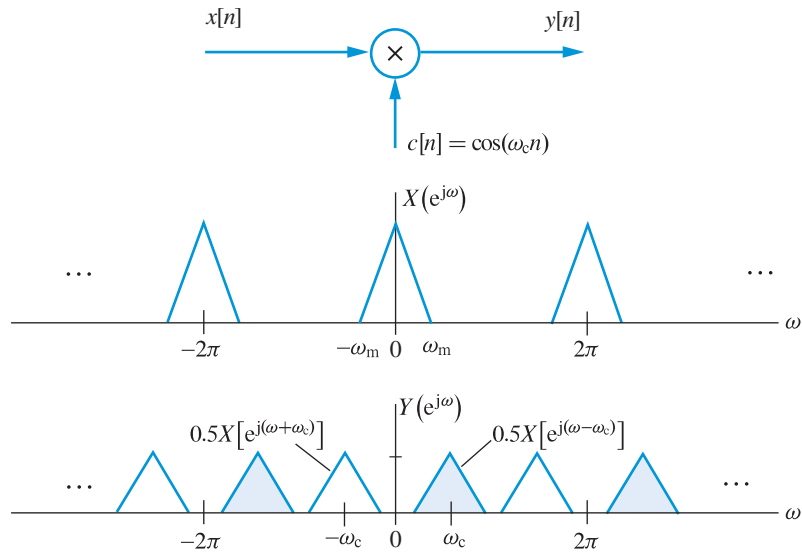
Figure 4.31 The modulation property of the DTFT using a real sinusoidal carrier.

or equivalently

$$\omega_m < \omega_c < \pi - \omega_m. \tag{4.144}$$

Relation (4.142) follows from (4.140) using Euler's identity $2\cos\omega_c n = e^{j\omega_c n} + e^{-j\omega_c n}$ and the frequency shifting theorem.

Differentiation in frequency Multiplying the value of each sample $x[n]$ by its index n, is equivalent to differentiating $X(e^{j\omega})$. More specifically,

$$nx[n] \xleftrightarrow{\text{DTFT}} -j\frac{dX(e^{j\omega})}{d\omega}, \tag{4.145}$$

which is obtained by differentiating both sides of (4.87).

Time reversal The time reversal or folding property is expressed as

$$x[-n] \xleftrightarrow{\text{DTFT}} X(e^{-j\omega}). \tag{4.146}$$

The proof is easily obtained by conjugating both sides of the definition of the DTFT (4.87). For a real sequence, we have

$$\mathcal{F}\{x[-n]\} = X(e^{-j\omega}) = |X(e^{-j\omega})|e^{j\angle X(e^{-j\omega})}$$

$$= |X(e^{j\omega})|e^{-j\angle X(e^{j\omega})}, \tag{4.147}$$

that is, "time reversal" is equivalent to "phase reversal." The shape of the magnitude spectrum depends only on the shape of the signal.

Conjugation of a complex sequence By the conjugation property,

$$x^*[n] \xleftrightarrow{\text{DTFT}} X^*(e^{-j\omega}). \tag{4.148}$$

The proof is easily obtained by conjugating both sides of the definition (4.87) of DTFT.

Convolution of sequences Convolving two sequences is equivalent to multiplying their Fourier transforms:

$$y[n] = x[n] * h[n] \xleftrightarrow{\text{DTFT}} Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega}). \tag{4.149}$$

This property is a consequence of linearity and time shifting properties. Indeed, applying successively the linearity and time shifting properties to the convolution summation

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k], \tag{4.150}$$

we obtain

$$Y(e^{j\omega}) = \sum_{k=-\infty}^{\infty} x[k]e^{-j\omega k}H(e^{j\omega}) = \left(\sum_{k=-\infty}^{\infty} x[k]e^{-j\omega k}\right)H(e^{j\omega})$$

$$= X(e^{j\omega})H(e^{j\omega}). \tag{4.151}$$

The convolution property, which is illustrated graphically in Figure 4.32, plays a very important role in the analysis of LTI systems.
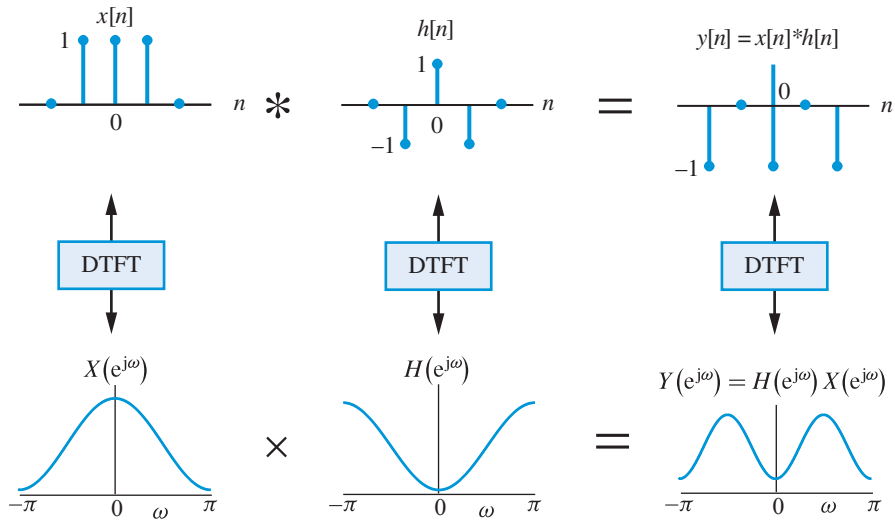
Figure 4.32 The convolution theorem of the DTFT.

Multiplication of sequences The DTFT of the product of two sequences is

$$s[n] = x[n]w[n] \xleftrightarrow{\text{DTFT}} S(e^{j\omega}) = \frac{1}{2\pi} \int_{2\pi} X(e^{j\theta})W\big[e^{j(\omega-\theta)}\big]d\theta. \qquad (4.152)$$

The proof is as follows:

$$S(e^{j\omega}) = \sum_{n=-\infty}^{\infty} s[n]e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x[n]w[n]e^{-j\omega n}$$

$$= \sum_{n=-\infty}^{\infty} \overbrace{\left[\frac{1}{2\pi}\int_{2\pi} X(e^{j\theta})e^{j\theta n}d\theta\right]}^{x[n]} w[n]e^{-j\omega n}$$

$$= \frac{1}{2\pi}\int_{2\pi} X(e^{j\theta})\underbrace{\left[\sum_{n=-\infty}^{\infty} w[n]e^{-j(\omega-\theta)n}\right]}_{W[e^{j(\omega-\theta)}]} d\theta$$

$$= \frac{1}{2\pi}\int_{2\pi} X(e^{j\theta})W\big[e^{j(\omega-\theta)}\big]d\theta.$$

The last integral, which can be evaluated over any interval of length 2π, generates what is known as the periodic convolution of $X(e^{j\omega})$ and $W(e^{j\omega})$. This property, which is also known as the *windowing theorem*, is widely used in spectral analysis and filter design.

Parseval's theorem According to this theorem, we have

$$\sum_{n=-\infty}^{\infty} x_1[n]x_2^*[n] = \frac{1}{2\pi}\int_{2\pi} X_1(e^{j\omega})X_2^*(e^{j\omega})d\omega. \qquad (4.153)$$

4.5 Properties of the discrete-time Fourier transform

Starting with the right hand side of (4.153) we have

$$\frac{1}{2\pi} \int_{2\pi} X_1(e^{j\omega}) X_2^*(e^{j\omega}) d\omega = \frac{1}{2\pi} \int_{2\pi} \left[\sum_{n=-\infty}^{\infty} x_1[n] e^{-j\omega n} \right] X_2^*(e^{j\omega}) d\omega$$

$$= \sum_{n=-\infty}^{\infty} x_1[n] \left[\frac{1}{2\pi} \int_{2\pi} X_2^*(e^{j\omega}) e^{-j\omega n} d\omega \right]$$

$$= \sum_{n=-\infty}^{\infty} x_1[n] \left[\frac{1}{2\pi} \int_{2\pi} X_2^*(e^{-j\omega}) e^{j\omega n} d\omega \right]$$

$$= \sum_{n=-\infty}^{\infty} x_1[n] x_2^*[n]. \quad \text{(using (4.148))}$$

For $x_1[n] = x_2[n] = x[n]$, we obtain Parseval's relation (4.94).

Summary of DTFT properties For easy reference, the operational properties of the DTFT are summarized in Table 4.4.

Table 4.4 Operational properties of the DTFT.

	Property	Sequence	Transform
		$x[n]$	$\mathcal{F}\{x[n]\}$
1.	Linearity	$a_1 x_1[n] + a_2 x_2[n]$	$a_1 X_1(e^{j\omega}) + a_2 X_2(e^{j\omega})$
2.	Time shifting	$x[n-k]$	$e^{-jk\omega} X(e^{j\omega})$
3.	Frequency shifting	$e^{j\omega_0 n} x[n]$	$X[e^{j(\omega-\omega_0)}]$
4.	Modulation	$x[n]\cos \omega_0 n$	$\frac{1}{2} X[e^{j(\omega+\omega_0)}] + \frac{1}{2} X[e^{j(\omega-\omega_0)}]$
5.	Folding	$x[-n]$	$X(e^{-j\omega})$
6.	Conjugation	$x^*[n]$	$X^*(e^{-j\omega})$
7.	Differentiation	$nx[n]$	$-j\dfrac{dX(e^{j\omega})}{d\omega}$
8.	Convolution	$x[n] * h[n]$	$X(e^{j\omega}) H(e^{j\omega})$
9.	Windowing	$x[n]w[n]$	$\dfrac{1}{2\pi} \int_{2\pi} X(e^{j\theta}) W\left[e^{j(\omega-\theta)}\right] d\theta$
10.	Parseval's theorem	$\displaystyle\sum_{n=-\infty}^{\infty} x_1[n]x_2^*[n] = \dfrac{1}{2\pi} \int_{2\pi} X_1(e^{j\omega}) X_2^*(e^{j\omega}) d\omega$	
11.	Parseval's relation	$\displaystyle\sum_{n=-\infty}^{\infty} |x[n]|^2 = \dfrac{1}{2\pi} \int_{2\pi} |X(e^{j\omega})|^2 d\omega$	

Correlation of signals

There are applications, like radar and digital communications, where we wish to measure the similarity between a signal of interest and a reference signal. The *correlation sequence* of two real-valued signals, say $x[n]$ and $y[n]$, is defined by

$$r_{xy}[\ell] \triangleq \sum_{n=-\infty}^{\infty} x[n]y[n-\ell]. \quad -\infty < \ell < \infty \tag{4.154}$$

The sequence $r_{xy}[\ell]$ exists, if at least one of the signals has finite energy. The sequence $y[n]$ is shifted to the right when the *lag* $\ell > 0$ and to the left when $\ell < 0$. To understand the meaning of correlation we first note that the energy E_z of the sequence $z[n] = ax[n] + y[n-\ell]$, which is nonnegative, can be expressed as

$$E_z = a^2 E_x + 2ar_{xy}[\ell] + E_y \geq 0. \tag{4.155}$$

This is a quadratic equation with coefficients E_x, $2r_{xy}[\ell]$, and E_y. The inequality in (4.155) is satisfied if the discriminant of the quadratic is nonpositive, that is, $4r_{xy}^2[\ell] - 4E_x E_y \leq 0$. Therefore, we have

$$-1 \leq \rho_{xy}[\ell] \triangleq \frac{r_{xy}[\ell]}{\sqrt{E_x}\sqrt{E_y}} \leq 1. \tag{4.156}$$

The sequence $\rho_{xy}[\ell]$, which is known as the normalized *correlation coefficient*, measures the similarity between the two sequences. If $x[n] = cy[n-n_0]$, $c > 0$, we obtain $\rho_{xy}[n_0] = 1$ (maximum correlation); in contrast, if $x[n] = -cy[n-n_0]$, $c > 0$, we obtain $\rho_{xy}[n_0] = -1$ (maximum negative correlation). If $\rho_{xy}[\ell] = 0$ for all lags, the two sequences are said to be uncorrelated. Computation of correlations and their interpretation are studied in Tutorial Problems 17–19.

Careful inspection of (4.154) reveals that the fundamental difference between convolution and correlation is that the sequence $y[n]$ is folded before the shifting operation. The absence of folding implies that

$$r_{xy}[\ell] = r_{yx}[-\ell], \tag{4.157}$$

that is, correlation does *not* have the commutative property. We can compute correlation using a function for the computation of convolution by first flipping the sequence $y[n]$, that is,

$$r_{xy}[\ell] = x[\ell] * y[-\ell]. \tag{4.158}$$

In MATLAB we can compute the correlation of two sequences using the function

$$\texttt{rxy=conv(x,flipud(y))}, \tag{4.159}$$

if `x` and `y` are column vectors. For `x` and `y` as row vectors we should use `fliplr(y)`. A development of a MATLAB function for the computation of correlation, which also provides timing information, is discussed in Tutorial Problem 19.

Taking the Fourier transform of (4.158) yields (see Problem 35)

$$r_{xy}[\ell] = x[\ell] * y[-\ell] \xrightarrow{\text{DTFT}} R_{xy}(\omega) = X(e^{j\omega})Y(e^{-j\omega}). \tag{4.160}$$

When $y[n] = x[n]$ we obtain the *autocorrelation* sequence $r_{xx}[\ell]$ or $r_x[\ell]$, for short. Since $x[n]$ is a real sequence, $X(-\omega) = X^*(\omega)$ and hence the Fourier transform of $r_x[\ell]$ is

$$r_x[\ell] = x[\ell] * x[-\ell] \xrightarrow{\text{DTFT}} R_x(\omega) = |X(e^{j\omega})|^2. \tag{4.161}$$

This Fourier transform pair is known as the Wiener–Khintchine theorem.

Example 4.15   Autocorrelation of exponential sequence

Let $x[n] = a^n u[n]$, $-1 < a < 1$. For $\ell > 0$, the product $x[n]u[n]x[n - \ell]u[n - \ell]$ is zero for $n < \ell$. Therefore, using the geometric summation formula, we have

$$r_x[\ell] = \sum_{n=\ell}^{\infty} x[n]x[n - \ell] = \sum_{n=\ell}^{\infty} a^n a^{n-\ell} = a^\ell(1 + a^2 + a^4 + \cdots)$$

$$= \frac{a^\ell}{1 - a^2}. \tag{4.162}$$

Since $r_x[\ell] = r_x[-\ell]$, the autocorrelation sequence for all values of ℓ is given by

$$r_x[\ell] = \frac{a^{|\ell|}}{1 - a^2}. \quad -1 < a < 1 \tag{4.163}$$

The Fourier transform of (4.163) is obtained using (4.161) and (4.127)

$$R_x(\omega) = X(e^{j\omega})X(e^{-j\omega}) = \frac{1}{1 - ae^{-j\omega}} \frac{1}{1 - ae^{j\omega}}$$

$$= \frac{1}{1 - 2a\cos(\omega) + a^2}. \tag{4.164}$$

Since $r_x[\ell]$ is real and even, its Fourier transform $R_x(\omega)$ is also real and even.   ■

For finite length sequences, correlation is a meaningful measure of similarity for small lags (smaller than 20 percent of the length); as the number of lags increases the number of samples used for the computation of correlation diminishes. Because correlation is primarily used for the analysis of signals corrupted by noise, an in depth discussion of correlation and its applications is provided in Chapters 13 and 14.

The correlation of periodic sequences is discussed in Chapter 7. The MATLAB function `c = xcorr(x,y)` returns the cross-correlation sequence in a length `2*N-1` vector, where `x` and `y` are length `N` vectors `(N>1)`. If `x` and `y` are not the same length, the shorter vector is zero-padded to the length of the longer vector. We avoid using this function because in practical applications we only need a small number of lags (see Section 14.2.1).

4.5.5   Signals with poles on the unit circle

The DTFT of a sequence $x[n]$ can be determined by evaluating its z-transform on the unit circle, provided that the ROC includes the unit circle (see Section 3.2). However, there are some useful aperiodic sequences with poles on the unit circle. For example, the z-transform of the unit step sequence

$$X(z) = \frac{1}{1 - z^{-1}}, \quad \text{ROC: } |z| > 1 \tag{4.165}$$

has a pole at $z = 1$. The DTFT is finite if $z = e^{j\omega} \neq 1$ or $\omega \neq 2\pi k$, k integer.

Similarly, the z-transform of the causal sinusoid $x[n] = \cos(\omega_0 n)u[n]$ is

$$X(z) = \frac{1 - (\cos \omega_0)z^{-1}}{1 - 2(\cos \omega_0)z^{-1} + z^{-2}}, \quad \text{ROC: } |z| > 1 \tag{4.166}$$

and has a pair of complex conjugate poles on the unit circle at $z = e^{\pm j\omega_0}$. The DTFT exist for $\omega \neq \pm\omega_0 + 2\pi k$.

The DTFT of sequences with poles on the unit circle can be formally defined for all values of ω by allowing Dirac impulse functions at the frequencies of the poles; however, this is not necessary for the needs of this text.

Learning summary

- The time domain and frequency domain representations of signals contain the same information in a different form. However, some signal characteristics and properties are better reflected in the frequency domain.
- The representation of a signal in the frequency domain (spectrum) consists of the amplitudes, frequencies, and phases of all sinusoidal components required to "build" the signal.
- The form of the formulas required to find the spectrum of a signal or synthesize a signal from its spectrum depends on whether:
  - the time variable is continuous or discrete;
  - the signal is periodic or nonperiodic.

  Therefore, there are four types of signal and related Fourier transform and series representations which are summarized in Figure 4.33.
- All Fourier representations share a set of properties that show how different characteristics of signals and how different operations upon signals are reflected in their spectra. The exact mathematical descriptions of these properties are different for each representation; however, the underlying concept is the same.
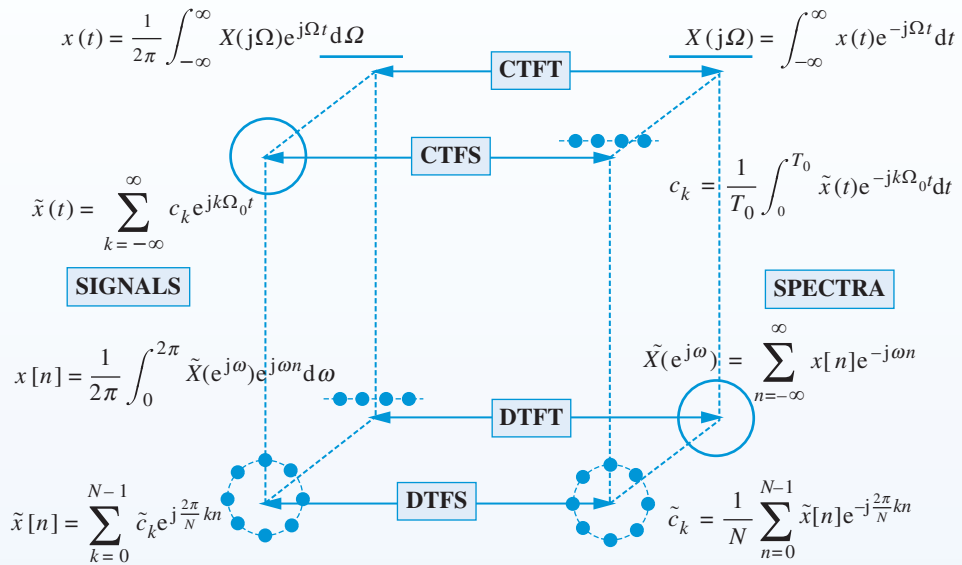
$$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega)e^{j\Omega t}d\Omega \qquad\qquad X(j\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}dt$$

CTFT

CTFS

$$\tilde{x}(t) = \sum_{k=-\infty}^{\infty} c_k e^{jk\Omega_0 t} \qquad\qquad c_k = \frac{1}{T_0} \int_0^{T_0} \tilde{x}(t)e^{-jk\Omega_0 t}dt$$

SIGNALS   **SPECTRA**

$$x[n] = \frac{1}{2\pi} \int_0^{2\pi} \tilde{X}(e^{j\omega})e^{j\omega n}d\omega \qquad\qquad \tilde{X}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

DTFT

DTFS

$$\tilde{x}[n] = \sum_{k=0}^{N-1} \tilde{c}_k e^{j\frac{2\pi}{N}kn} \qquad\qquad \tilde{c}_k = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[n]e^{-j\frac{2\pi}{N}kn}$$

Figure 4.33 Summary of four Fourier representations.

TERMS AND CONCEPTS

Amplitude spectrum A graph of the Fourier series coefficients or transform as a function of frequency when these quantities are real-valued.

Analog frequency Represents a number of occurrences of a repeating event per unit time. For sinusoidal signals, the linear frequency, F, is measured in cycles per second (or Hz) while the angular (or radian) frequency, $\Omega = 2\pi F$, is measured in radians per second.

Autocorrelation sequence A sequence defined by $r_x[\ell] = \sum_{n=-\infty}^{\infty} x[n]y[n-\ell]$ that measures a degree of similarity between samples of a real-valued sequence $x[n]$ at a lag ℓ.

Continuous-Time Fourier Series (CTFS) Expresses a continuous-time periodic signal $x(t)$ as a sum of scaled complex exponentials (or sinusoids) at harmonics kF_0 of the fundamental frequency F_0 of the signal. The scaling factors are called Fourier series coefficients c_k.

Continuous-Time Fourier Transform (CTFT) Expresses a continuous-time aperiodic

signal $x(t)$ as an integral of scaled complex exponentials (or sinusoids) of all frequencies. The scaling factor is denoted by $X(j2\pi F)$.

Correlation coefficient A sequence denoted by $\rho_{xy}[\ell]$ which is a normalized correlation between samples of two real-valued sequences $x[n]$ and $y[n]$ at a lag ℓ and measures similarity between the two.

Correlation sequence A sequence defined by $r_{xy}[\ell] = \sum_{n=-\infty}^{\infty} x[n]y[n-\ell]$ that measures similarity between samples of two real-valued sequences $x[n]$ and $y[n]$ at a lag ℓ.

Dirichlet conditions Requirements on the signals that are needed to determine Fourier series or transform of continuous- or discrete-time signals.

Dirichlet's function A periodic sinc function denoted by $D_L(x)$ and defined as $\frac{\sin(xL/2)}{L\sin(x/2)}$. Its value is 1 at $x = 0$.

Discrete-Time Fourier Series (DTFS) Expresses a discrete-time periodic signal $x[n]$ as a finite sum of scaled complex exponentials (or sinusoids) at harmonics k/N

of the fundamental frequency $1/N$ of the signal. The scaling factors are called Fourier series coefficients, c_k, which themselves form a periodic sequence.

Discrete-Time Fourier Transform (DTFT) Expresses a discrete-time aperiodic signal $x(t)$ as an integral of scaled complex exponentials (or sinusoids) of all frequencies. The scaling factor is denoted by $X(\omega)$.

Energy density spectrum A graph of $|X(j2\pi F)|^2$ or $|X(j\Omega)|^2$ as a function of frequency. It is a continuous spectrum.

Fundamental frequency Defined for periodic signals, it is the reciprocal of fundamental period. For continuous-time periodic signals it is denoted by $F_0 = 1/T_0$ in cycles per second, while for discrete-time signals it is denoted by $f_0 = 1/N$ in cycles per sample.

Fundamental harmonic The complex exponential (or sinusoid) associated with the fundamental period in set of harmonically-related complex exponentials.

Fundamental period Defined for periodic signals, it is the smallest period with respect to which a periodic signal repeats itself. For continuous-time periodic signals the fundamental period is T_0 in seconds, while for discrete-time signals the fundamental period is N in samples.

Harmonic frequencies or Harmonics Frequencies that are integer multiples of the fundamental frequency.

Harmonically-related complex exponentials A set of complex exponential signals with frequencies that are integer multiples of the fundamental frequency.

Magnitude spectrum A graph of the magnitude of the Fourier series coefficients or transform as a function of frequency.

Normalized frequency Defined for discrete-time sinusoids, it represents a number of occurrences of a repeating event per sample. For sinusoidal signals, the linear normalized frequency, f, is measured in cycles per sample while the normalized angular (or radian) frequency, $\omega = 2\pi f$, is measured in radians per sample.

Orthogonality property Defined for harmonically-related complex exponentials. For continuous-time complex exponentials it is given by

$$\frac{1}{T_0} \int_{T_0} e^{jk\Omega_0 t} e^{-jm\Omega_0 t} dt = \delta[k-m],$$

and for discrete-time complex exponentials it is given by

$$\frac{1}{N} \sum_{n=<N>} e^{j\frac{2\pi}{N}kn} e^{-j\frac{2\pi}{N}mn} = \delta[k-m].$$

Phase spectrum A graph of the phase of the Fourier series coefficients or transform as a function of frequency.

Power spectrum A graph of $|c_k|^2$ as a function of harmonic frequency. It is a line spectrum.

Sinc function Denoted by $\text{sinc}(x)$ and defined as $\frac{\sin(\pi x)}{\pi x}$. Its value is 1 at $x = 0$.

MATLAB functions and scripts

Name	Description	Page
angle	Computes angle of a complex number	180
diric	Computes the digital sinc function	162
dtfs*	Computes the DTFS	162
dtft12*	Computes the DTFT of $x[n]$, $N_1 \leq n \leq N_2$	168
fft	Computes the DTFS	162
freqz	Computes the DTFT of finite duration sequences	168
idtfs*	Computes the inverse DTFS	162
ifft	Computes the inverse DTFS	162
sinc	Computes the sinc function	147
unwrap	Computes a "continuous" phase from principal values	180

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

- A detailed treatment of continuous-time and discrete-time Fourier series and transforms, at the same level as in this book, is given in Oppenheim *et al.* (1997) and Lathi (2005).
- The standard references for Fourier transforms from an electrical engineering perspective are Bracewell (2000) and Papoulis (1962).
- A mathematical treatment of Fourier series and transforms is given in Walker (1988) and Kammler (2000).

Review questions

1. Describe the eigenvalue–eigenfunction concept for the LTI systems and explain why it is important.
2. Explain the set of harmonically-related complex exponentials (both continuous- and discrete-time). What is the fundamental harmonic term?
3. Describe the orthogonality property satisfied by the elements from the set of harmonically-related complex exponentials and its importance to Fourier analysis.
4. Enumerate the important properties of the continuous-time sinusoidal signals.
5. Describe various "frequency" variables and units used in Fourier analysis.
6. Enumerate the important properties of the discrete-time sinusoidal signals.
7. Define and explain the CTFS representation of continuous-time periodic signals using analysis and synthesis equations.
8. What are the Dirichlet conditions for the existence of the CTFS.
9. Describe the Gibbs phenomenon and explain why it occurs in the synthesis of signals using sinusoids.

10. Define and explain the CTFT representation of continuous-time aperiodic signals using analysis and synthesis equations.

11. What are the Dirichlet conditions for the existence of the CTFT.

12. Define and explain the DTFS representation of discrete-time periodic signals using analysis and synthesis equations.

13. What are the Dirichlet conditions for the existence of the DTFS.

14. Define and explain the DTFT representation of discrete-time aperiodic signals using analysis and synthesis equations.

15. What are the Dirichlet conditions for the existence of the DTFT.

16. Explain the various line spectra (magnitude, phase, amplitude, and power) for both continuous- and discrete-time periodic signals.

17. Explain the various continuous spectra (magnitude, phase, amplitude, and energy-density) for both continuous- and discrete-time aperiodic signals.

18. Give four different Parseval's relations and state what each signifies?

19. How is DTFT related to the z-transform of the same discrete-time signal?

20. If the z-transform of a discrete-time signal exists then its DTFT also exists. Do you agree or disagree? Explain.

21. If the DTFT of a discrete-time signal exists then its z-transform also exists. Do you agree or disagree? Explain.

22. Explain the concepts of correlation, autocorrelation, and correlation coefficients of discrete-time signals.

Problems

Tutorial problems

1. Let $x_1(t)$ and $x_2(t)$ be periodic signals with fundamental periods T_1 and T_2, respectively. Under what conditions is the sum $x(t) = x_1(t) + x_2(t)$ periodic, and what is its fundamental period T if it is periodic?

2. Determine whether or not each of the following signals is periodic. If a signal is periodic, determine its fundamental period:
   (a) $x_1(t) = \sin(\pi t/3) + \cos(\pi t/4)$,
   (b) $x_2(t) = \sin(10\pi t) + \sin(\sqrt{2}t)$,
   (c) $x_3[n] = \cos(n/5)$,
   (d) $x_4[n] = \cos(\pi n/3) + \sin(\pi n/4)$,
   (e) $x_5(t) = [\cos(2\pi t/3) + 2\sin(16\pi t/3)]\sin\pi t$.

3. Using MATLAB, plot each of the following functions over the indicated time interval and verify that the area under the function is zero:
   (a) $x_1(t) = 2\cos(10\pi t) \times 3\cos(20\pi t)$, $-0.2 \le t \le 0.2$,
   (b) $x_2(t) = 3\sin(0.2\pi t) \times 5\cos(2\pi t)$, $0 \le t \le 20$,
   (c) $x_3(t) = 5\cos(5\pi t) \times 4\sin(10\pi t)$, $0 \le t \le 2$,
   (d) $x_4(t) = 4\sin(100\pi t) \times 2\cos(400\pi t)$, $0 \le t \le 0.01$.

4. Let $x(t) = 2 + 4\sin(3\pi t) + 6\cos(8\pi t + \pi/3)$. This is a periodic signal.
   (a) Determine the average power P_{av} in $x(t)$.
   (b) Determine the fundamental frequency Ω_0 of $x(t)$.
   (c) Compute the CTFS coefficients c_k and express them in the magnitude-phase format. Plot the magnitude, phase, and power spectra as a function of $k\Omega_0$.
   (d) Determine the average power of $x(t)$ from the frequency domain and verify that it equals P_{av} in part (a) above.

5. Determine the CTFS representation of the periodic full-wave rectified signal $x(t) = |\cos(10\pi t)|$. Plot the magnitude and phase spectra for $-10 \le k \le 10$ as a function of $k\Omega_0$.

6. Prove the orthogonality property (4.7) and use it to prove Parseval's relation (4.27).

7. Let $h(t)$ and $x(t)$ be two periodic signals with the same fundamental period T_0 and Fourier coefficients a_k and b_k, respectively. Show that the Fourier coefficients c_k of $y(t) = h(t)x(t)$ are given by the convolution sum

$$c_k = \sum_{\ell=-\infty}^{\infty} a_\ell b_{k-\ell}.$$

8. Consider the continuous-time aperiodic signal $x(t)$ and the periodic signal $\tilde{x}(t)$ defined by

$$x(t) = \begin{cases} e^{-t}, & -1 < t < 1 \\ 0, & \text{otherwise} \end{cases} \quad \text{and} \quad \tilde{x}(t) = \sum_{\ell=-\infty}^{\infty} x(t - \ell 2).$$

   (a) Compute the Fourier transform $X(j2\pi F)$ of $x(t)$ and the Fourier series coefficients c_k of $\tilde{x}(t)$.
   (b) Using the results in (a) verify the relationship $c_k = X(j2\pi k/T_0)/T_0$.
   (c) Plot $|X(j2\pi F)|$ and $|c_k|$ on one plot and $\angle X(j2\pi F)$ and $\angle c_k$ on another plot to illustrate the result in part (b).

9. An aperiodic signal $x(t) = 2\mathrm{sinc}(2t)$ is multiplied by a periodic rectangular pulse train $s(t)$ with fundamental frequency $F_0 = 4$ Hz and the fundamental rectangular pulse given by

$$p(t) = \begin{cases} 1, & -1/80 \le t \le 1/80 \\ 0, & \text{otherwise} \end{cases}$$

   to obtain a signal $x_s(t) = x(t)s(t)$.
   (a) Compute and plot the CTFT $X(j2\pi F)$ over the $|F| \le 80$ range in Hz.
   (b) Compute and plot the CTFS coefficients c_k over the $|F| \le 80$ range in Hz.
   (c) Compute $X_s(j2\pi F)$ using (4.62) and plot it over the $|F| \le 80$ range in Hz.

10. In this problem we illustrate the numerical evaluation of DTFS using MATLAB.
   (a) Write a function `c=dtfs0(x)` which computes the DTFS coefficients (4.67) of a periodic signal.
   (b) Write a function `x=idtfs0(c)` which computes the inverse DTFS (4.63).
   (c) Verify that your functions are working correctly by replicating the results in Example 4.9.

11. Determine and plot the magnitude and phase spectra of the following periodic sequences:

(a) $x_1[n] = \sin[2\pi(3/10)n]$,
(b) $x_2[n] = \{1, 2, -1, 0, -1, 2\}, 0 \le n \le 5$ (one period),
(c) $x_3[n] = 1 - \sin(\pi n/4), 0 \le n \le 3$ (one period),
(d) $x_4[n] = 1 - \sin(\pi n/4), 0 \le n \le 11$ (one period),
(e) $x_5[n] = \{1, 1, 0, 1, 1, 1, 0, 1\}, 0 \le n \le 7$ (one period),
(f) $x_6[n] = 1$ for all n. (Hint: Treat the sequence as periodic.)

12. Determine the DTFT and plot its magnitude and phase for the following sequences:
(a) $x_1[n] = u[n]$, (b) $x_2[n] = \cos(\omega_0 n)u[n]$, $\omega_0 = \pi/3$.

13. Determine and plot the magnitude and phase spectra of the following signals:
(a) $x_1[n] = (1/2)^{|n|} \cos(\pi(n-1)/8)$,
(b) $x_2[n] = n(u[n+3] - u[n-4])$,
(c) $x_3[n] = (2 - n/2)(u[n+4] - u[n-5])$.

14. Determine the sequence $x[n]$ corresponding to each of the following Fourier transforms:
(a) $X_1(e^{j\omega}) = \cos^2(\omega) + \sin^2(3\omega)$,
(b) $X_2(e^{j\omega}) = 0, 0 \le |\omega| \le \omega_c$ and $X_2(e^{j\omega}) = 1, \omega_c < |\omega| \le \pi$
(c) $X_3(e^{j\omega}) = 1 - 2|\omega|/\pi, 0 \le |\omega| \le \pi/2$ and $X_3(e^{j\omega}) = 0, \pi/2 < |\omega| \le \pi$
(d) With $\Delta\omega > 0$ and $\omega_c > \Delta\omega/2$, $X_4(e^{j\omega})$ is given by

$$X_4(e^{j\omega}) = \begin{cases} 1, & \omega_c - \frac{\Delta\omega}{2} \le |\omega| \le \omega_c + \frac{\Delta\omega}{2} \le \pi \\ 0. & \text{otherwise} \end{cases}$$

15. Given a sequence $x[n]$ with Fourier transform $X(\omega)$, determine the Fourier transform of the following sequences in terms of $X(\omega)$:
(a) $x_1[n] = x[1+n] + x[-1-n]$,
(b) $x_2[n] = (x[n] + x^*[n])/2$,
(c) $x_3[n] = (1-n)^2 x[n]$.

16. The signal $x[n] = \{-1, 2, -3, 2, -1\}, -2 \le n \le 2$ has Fourier transform $X(e^{j\omega})$. Find the following quantities without explicitly computing $X(e^{j\omega})$:
(a) $X(e^{j0})$, (b) $\angle X(e^{j\omega})$, (c) $\int_{-\pi}^{\pi} X(e^{j\omega})d\omega$,
(d) $X(e^{j\pi})$, (e) $\int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega$.

17. Let $x[n] = [1, 2, \overset{\uparrow}{3}, 2, 1]$ and $y[n] = [2, 1, \overset{\uparrow}{0}, -1, -2]$.

(a) Using the scanning operation described in Chapter 2, determine and plot the correlation $r_{xy}[l]$ between the two signals.
(b) Determine and plot the correlation coefficient $\rho_{xy}[l]$.
(c) Comment on the degree of similarity between the two signals.

18. Let $x[n] = (0.9)^n u[n]$. Determine correlation $r_{xy}[l]$ and correlation coefficient $\rho_{xy}[l]$ for the following cases:
(a) $y[n] = x[n]$,
(b) $y[n] = x[-n]$,
(c) $y[n] = x[n+5]$.
Comment on the results in each case.

19. Develop a MATLAB function $[rxy, l] = ccrs(x, nx, y, ny)$ that computes correlation rxy between two finite length signals x and y defined over nx and ny intervals, respectively. Verify your function using signals given in Problem 17.

Basic problems

20. Let $x_1[n]$ and $x_2[n]$ be periodic sequences with fundamental periods N_1 and N_2, respectively. Under what conditions is the sum $x[n] = x_1[n] + x_2[n]$ periodic, and what is its fundamental period N if it is periodic?

21. Determine whether or not each of the following signals is periodic. If a signal is periodic, determine its fundamental period:
 (a) $x_1(t) = 2\cos(3\pi t) + 3\sin(4t)$,
 (b) $x_2[n] = 4\cos(0.1\pi n)$,
 (c) $x_3(t) = 3\sin(3000\pi t) + 5\sin(2000\pi t)$,
 (d) $x_4[n] = 2\cos(n/11) + 5\sin(n/31)$,
 (e) $x_5[n] = [\cos(\pi n/5) + 2\sin(\pi n/6)]\sin(\pi n/2)$.

22. Consider the continuous-time signal $x(t) = \cos(15\pi t)$, $-\infty < t < \infty$.
 (a) Determine the sampling period T such that the sequence $x[n] = x(nT)$ is periodic.
 (b) Find the fundamental period of the sequence $x[n]$ if $T = 0.1$ s.

23. Determine the Fourier series coefficients of the triangular pulse train with a single period defined by $x(t) = A(1 - 2|t|/T_0)$, $|t| < T_0/2$.
 (a) Plot the magnitude and phase spectra of $x(t)$ for $A = 1$ and $T_0 = 1$.
 (b) Compute the partial sums (4.33) to reproduce the plots in Figure 4.11.

24. Determine and plot the magnitude and phase spectra of the following signals:
 (a) $x_1(t) = (1 - t^2)[u(t) - u(t - 1)]$,
 (b) $x_2(t) = e^{-3|t|}\sin 2\pi t$,
 (c) $x_3(t) = \frac{\sin \pi t}{\pi t}\frac{\sin 2\pi t}{\pi t}$.

25. For the periodic pulse train sequence discussed in Example 4.8, write a MATLAB script to compute the partial sum

$$\hat{x}_M[n] = \sum_{k=-M}^{M} a_k e^{j(2\pi/N)kn}.$$

 (a) For $L = 2$ and $N = 9$, plot the sequence $\hat{x}_M[n]$ for $M = 1, 2, 3, 4$. Is the sequence $\hat{x}_M[n]$ real? Why?
 (b) Repeat (a) for $N = 10$ by first changing the lower limit of the summation from $-M$ to $-M + 1$ (why?).

26. Determine and plot the magnitude and phase spectra of the following periodic sequences:
 (a) $x_1[n] = 4\cos(1.2\pi n + 60°) + 6\sin(0.4\pi n - 30°)$,
 (b) $x_2[n] = |\cos(0.25\pi n)|, 0 \le n \le 3$ (one period)
 (c) $x_3[n] = \{\underset{\uparrow}{1}, 1, 0, 1, 1, 1, 0, 1\}$, (one period)
 (d) $x_4[n] = 1 - \sin(\pi n/4), 0 \le n \le 11$ (one period)
 (e) $x_5[n] = \{\underset{\uparrow}{1}, -2, 1, 0, -1, 2, -1\}$. (one period)

27. Given that $x[n]$ is a periodic sequence with fundamental period N and Fourier coefficients a_k, determine the Fourier coefficients of the following sequences:
 (a) $x[n - n_0]$,
 (b) $x[n] - x[n - 1]$,
 (c) $(-1)^n x[n]$ (N even),
 (d) $(-1)^n x[n]$ (N odd).

28. Let $x[n]$ be a periodic sequence with fundamental period N and Fourier coefficients a_k.
    (a) Express the Fourier coefficients b_k of $y[n] = |x[n]|^2$ in terms of a_k.
    (b) If a_k are real, can we claim that b_k are real as well?

29. Let $h[n]$ and $x[n]$ be periodic sequences with fundamental period N and Fourier coefficients a_k and b_k, respectively.
    (a) Show that the Fourier coefficients c_k of $y[n] = h[n]x[n]$ are given by

$$c_k = \sum_{\ell=0}^{N-1} a_\ell b_{k-\ell} = \sum_{\ell=0}^{N-1} b_\ell a_{k-\ell}.$$

    (b) Verify the result in (a) using the periodic sequences $(N = 8)$

$$h[n] = \sin(3\pi n/4),$$

$$\text{and} \quad x[n] = \{1, 1, 1, 1, 0, 0, 0, 0\}.$$

30. Determine and plot the DTFT magnitude and phase spectra of the following signals:
    (a) $x_1[n] = (1/3)^n u[n-1]$,
    (b) $x_2[n] = (1/4)^n \cos(\pi n/4) u[n-2]$,
    (c) $x_3[n] = \text{sinc}(2\pi n/8) * \text{sinc}\{2\pi(n-4)/8\}$,
    (d) $x_4[n] = \sin(0.1\pi n)(u[n] - u[n-10])$,
    (e) $x_5[n] = \text{sinc}^2(\pi n/4)$.

31. Determine the sequence $x[n]$ corresponding to each of the following Fourier transforms:
    (a) $X(e^{j\omega}) = \delta(\omega) - \delta(\omega - \pi/2) - \delta(\omega + \pi/2)$,
    (b) $X(e^{j\omega}) = 1, 0 \le |\omega| \le 0.2\pi$ and $X(e^{j\omega}) = 0, 0.2\pi < |\omega| \le \pi$
    (c) $X(e^{j\omega}) = 2|\omega|/\pi, 0 \le |\omega| \le \pi/2$ and $X(e^{j\omega}) = 0, \pi/2 < |\omega| \le \pi$
    (d) With $\Delta\omega > 0$ and $\omega_c > \Delta\omega/2$, $X(e^{j\omega})$ is given by

$$X(e^{j\omega}) = \begin{cases} 0, & \omega_c - \frac{\Delta\omega}{2} \le |\omega| \le \omega_c + \frac{\Delta\omega}{2} \\ 1. & \text{otherwise} \end{cases}$$

32. Given a sequence $x[n]$ with Fourier transform $X(e^{j\omega})$, determine the Fourier transform of the following sequences in terms of $X(e^{j\omega})$:
    (a) $x_1[n] = 2x[n+2] + 3x[3-n]$,
    (b) $x_2[n] = (1 + x[n]) \cos(0.2\pi n + \pi/6)$,
    (c) $x_3[n] = 2e^{j0.5\pi(n-2)}x[n+2]$,
    (d) $x_4[n] = (x[n] - x^*[-n])/2$,
    (e) $x_5[n] = j^n x[n+1] + j^{-n} x[n-1]$.

33. Given a sequence with Fourier transform $X(e^{j\omega}) = 1/(1 + 0.8^{-j\omega})$, determine the Fourier transform of the following signals:
    (a) $x_1[n] = e^{j\pi n/2} x[n+2]$,
    (b) $x_2[n] = x[n] \cos(0.4\pi n)$,

(c) $x_3[n] = x[n] * x[-n]$,

(d) $x_4[n] = x[2n]$,

(e) $x_5[n] = x[n]$, $n =$ even and $x_5[n] = 0$, $n =$ odd.

34. Let $x[n]$ be a purely imaginary signal, that is, $x[n] = 0 + jx_I[n]$.

(a) Develop the DTFT analysis and synthesis equations using (4.106) through (4.109). Comment on their symmetry properties.

(b) Assume that $x_I[n]$ has an even symmetry. Develop the DTFT analysis and synthesis equations and comment on their symmetry properties.

(c) Finally, assume that $x_I[n]$ has an odd symmetry. Develop the DTFT analysis and synthesis equations and comment on their symmetry properties.

35. Let $x[n]$ and $y[n]$ be two finite-energy signals. The correlation, $r_{xy}[l]$, between the two signals is defined in (4.154) and the signal autocorrelation, $r_x[l]$ is obtained when $y[n] = x[n]$.

(a) Show that the "cross" spectral density function $R_{xy}(\omega)$ is given by (4.160).

(b) Show that the "auto" spectral density function $R_x(\omega)$ is given by (4.161).

36. Signal $x[n] = \sin(0.2\pi n)$, $-200 \le n \le 200$, when sent over a channel is delayed and contaminated by noise. It is observed as $y[n] = x[n - D] + w[n]$ where D is an amount of delay in samples and $w[n]$ is a Gaussian sequence with mean 0 and variance 0.1.

(a) Compute and plot the correlation $r_{xy}[l]$ between the $x[n]$ and $y[n]$ For $D = 10, 20$, and 50.

(b) Can you determine delay D from the observation of $r_{xy}[l]$?

Assessment problems

37. Write a MATLAB program to generate and plot the signals given in Figure 4.3.

38. Determine whether or not each of the following signals is periodic. If a signal is periodic, determine its fundamental period:

(a) $x_1(t) = |\sin(7\pi t)| \cos(11\pi t)$,

(b) $x_2(t) = \sin(\sqrt{2}t) + \cos(2\sqrt{2}t)$,

(c) $x_3(t) = \frac{1}{3}\{\sin(t/11) + \cos(t/79) + \sin(t/31)\}$,

(d) $x_4[n] = e^{j\pi n/7} + e^{j\pi n/11}$,

(e) $x_5[n] = |\cos(0.1\pi n)| + \sin(2\pi n/11)$.

39. Use the geometric summation formula to prove the orthogonality property (4.22). Provide a geometric interpretation by treating the samples of $s_k[n]$, $k = 0, 1, \ldots, N - 1$ as the components of an N-dimensional vector.

40. Write a MATLAB program to generate and plot the signals shown in Figure 4.12. Experiment with different values of m to appreciate the nature of Gibbs' phenomenon. Note: You can zoom on the discontinuities to see more clearly the behavior of oscillations.

41. Use the orthogonality property (4.22) to prove Parseval's relation (4.69).

42. Write a MATLAB script to compute and plot the Dirichlet function (4.80) for $L = 6$ and $D = 7$. What is the fundamental period in each case?

43. Show that for $K = N$, we can recover the N samples of $x[n]$ from the N samples of $X(e^{j\omega_k})$ by solving a linear system of equations. Use MATLAB to demonstrate this result with the signal $x[0] = 1, x[1] = 2, x[2] = 3$, and $x[n] = 0$ otherwise.

44. Determine and plot the magnitude and phase spectra of the following periodic sequences:

    (a) $x_1[n] = \{1, \underset{\uparrow}{2}, 3, 3, 3, 2, 1\}$, (one period)

    (b) $x_2[n] = |\sin(0.2\pi n)|$, $-5 \le n \le 4$ (one period)

    (c) $x_3[n] = e^{j2\pi n/7} + e^{j\pi n/3} + e^{j\pi n/7}$,

    (d) $x_4[n] = \{\underset{\uparrow}{1}, 2, 3, 4, 5, 6, 7, 8\}$, $0 \le n \le 7$ (one period)

    (e) $x_5[n] = (-1)^n$ for all n.

45. Given that $x[n]$ is a periodic sequence with fundamental period N and Fourier coefficients a_k, determine the Fourier coefficients of the following sequences in terms of a_k:

    (a) $x[n+1] + 2x[n] + x[n-1]$,

    (b) $e^{-j6\pi n/N} x[n-2]$,

    (c) $3\cos(2\pi 5n/N)x[-n]$,

    (d) $x[n] + x^*[-n]$.

46. Prove Parseval's theorem (4.153) using the multiplication and conjugation properties.

47. Let the system function be

$$H(z) = \frac{1 - z^{-2}}{1 - 0.9\sqrt{2}z^{-1} + 0.81z^{-2}}.$$

    (a) Provide a surface plot of the magnitude of $H(z)$ over $\{-2 \le \mathcal{R}e(z) \le 2\} \cap \{-2 \le \mathcal{I}m(z) \le 2\}$ region. On this surface plot superimpose the magnitude response $|H(e^{j\omega})|$ and adjust the view angle so that your plot looks similar to the top left plot in Figure 4.26.

    (b) Using the `pol2cart` function truncate the surface plot in (a) above to obtain a plot similar to the top right plot in Figure 4.26.

    (b) Provide zero-pole and magnitude response plots of the system function as in Figure 4.26.

48. Determine and plot the magnitude and phase spectra of the following signals:

    (a) $x_1[n] = 3(0.9)^n u[n]$,

    (b) $x_2[n] = 2(-0.8)^{n+2} u[n-2]$,

    (c) $x_3[n] = (n+2)(-0.7)^{n-1} u[n-2]$,

    (d) $x_4[n] = 5(-0.8)^n \cos(0.1\pi n)u[n]$,

    (e) $x_5[n] = (0.7)^{|n|}(u[n+10] - u[n-11])$.

49. Determine sequences corresponding to each of the following Fourier transforms:

    (a) $X_1(e^{j\omega}) = 2 + 3\cos(\omega) + 4\cos(3\omega)$,

    (b) $X_2(e^{j\omega}) = [1 + 5\cos(2\omega) + 8\cos(4\omega)]e^{-j3\omega}$,

    (c) $X_3(e^{j\omega}) = je^{-j4\omega}[2 + 3\cos(\omega) + \cos(2\omega)]$,

    (d) $X_4(e^{j\omega}) = \begin{cases} 2, & 0 \le |\omega| \le \pi/8 \\ 1, & \pi/8 \le |\omega| \le 3\pi/4 \\ 0, & 3\pi/4 \le |\omega| \le \pi \end{cases}$

    (e) $X_5(e^{j\omega}) = \omega e^{j(\pi/2 - 5\omega)}$.

50. Consider a periodic sequence $\tilde{x}[n]$ with fundamental period N and Fourier coefficients a_k. Define a sequence $x[n] = \tilde{x}[n](u[n-n_0] - u[n-n_0-N])$ with Fourier transform $X(e^{j\omega})$.

(a) Show that for any value of n_0 we have $a_k = (1/N)X(e^{j2\pi k/N})$.

(b) Use the sequence $x[n] = u[n] - u[n-5]$ to verify the formula in (a).

51. Given a sequence $x[n]$ with Fourier transform $X(e^{j\omega})$, determine the Fourier transform of the following sequences in terms of $X(e^{j\omega})$:

(a) $x_1[n] = \frac{1}{6}\sum_{k=-2}^{2} |k| x[n-k]$,

(b) $x_2[n] = [(0.9)^n \cos(0.1\pi n)] * x[n-2]$,

(c) $x_3[n] = nx[n-1] + n^2 x[n-2]$,

(d) $x_4[n] = (x[n] - jx^*[-n])/2$,

(e) $x_5[n] = [(-0.7)^n \sin(0.4\pi n)] * x[n+2]$.

52. Use Parseval's theorem to compute the following summation

$$S = \sum_{n=-\infty}^{\infty} \frac{\sin(\pi n/4)}{2\pi n} \frac{\sin(\pi n/6)}{5\pi n}.$$

53. Using the frequency-shifting property of the DTFT, show that the real part of DTFT of the sinusoidal pulse $x[n] = (\cos\omega_0 n)(u[n] - u[n-M])$ is given by

$$X(e^{j\omega}) = \frac{1}{2}\cos\left\{\frac{(\omega-\omega_0)(M-1)}{2}\right\}\left[\frac{\sin\{(\omega-\omega_0)M/2\}}{\sin\{(\omega-\omega_0)/2\}}\right] + \frac{1}{2}\cos\left\{\frac{(\omega+\omega_0)(M-1)}{2}\right\}\left[\frac{\sin\{(\omega+\omega_0)M/2\}}{\sin\{(\omega+\omega_0)/2\}}\right].$$

Compute and plot $X(e^{j\omega})$ for $\omega_0 = \pi/2$ and $N = 5, 15, 25, 100$. Use the plotting interval of $[-\pi, \pi]$. Comment on your results.

54. The signal $x[n] = \{1, -2, 3, -4, \overset{\uparrow}{0}, 4, -3, 2, -1\}$, has Fourier transform $X(e^{j\omega})$. Find the following quantities without explicitly computing $X(e^{j\omega})$:

(a) $X(e^{j0})$,   (b) $\angle X(e^{j\omega})$,   (c) $\int_{-\pi}^{\pi} X(e^{j\omega})d\Omega$,

(d) $X(e^{j\pi})$,   (e) $\int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\Omega$.

55. In a concert hall signal, an echo is generated due to reflection from walls and ceiling. In a simplified model, the audio signal $y[n]$ received by a listener is modeled using $y[n] = x[n] + a x[n-D]$, where $x[n]$ is the original signal, D is the amount of delay in samples, and $0 < a < 1$ is the echo amplitude.

(a) Develop an expression for the autocorrelation $r_y(\ell)$ in terms of the autocorrelation of $r_x(\ell)$.

(b) Using $x[n] = \cos(0.1\pi n) + 0.8\cos(0.4\pi n)$, $a = 0.1$, and $D = 50$ generate $y[n]$ over $0 \le n \le 300$. Compute and plot $r_y(\ell)$ using MATLAB.

(c) Can you obtain a and D from the plot of $r_y(\ell)$ above?

Review problems

56. The MATLAB script `dtftprop.m` generates plots that illustrate various properties of DTFT. Run the script and explain the results obtained using relevant symmetry properties and theorems of the DTFT.

57. A continuous-time periodic signal $x(t)$, with period $T_0 = 1$ s, is given by $x(t) = (1 - 4|t|)/2$ over one period $-0.5 \le t \le 0.5$.

(a) Determine the CTFS coefficients $\{c_k\}$ for the above signal. You should notice that these coefficients are real-valued, symmetric, and with odd harmonics.

(b) The CTFS $\{c_k\}$ can be considered as a discrete-time signal. Compute and plot its DTFT $C(e^{j\omega})$. Use MATLAB to perform this operation and include a sufficient

number of (even and odd) CTFS coefficients. Compare your plot with the periodic signal $x(t)$ and comment.

(c) Using your results in (b) above develop a relationship between $x(t)$ and $C(e^{j\omega})$.

(d) Repeat part (c) by considering only the non-zero odd harmonics (you may need to shift CTFS by one harmonic). Explain your result.

5 Transform analysis of LTI systems

In Chapter 2 we discussed representation and analysis of LTI systems in the time-domain using the convolution summation and difference equations. In Chapter 3 we developed a representation and analysis of LTI systems using the *z*-transform. In this chapter, we use Fourier representation of signals in terms of complex exponentials and the pole-zero representation of the system function to characterize and analyze the effect of LTI systems on the input signals. The fundamental tool is the frequency response function of a system and the close relationship of its shape to the location of poles and zeros of the system function. Although the emphasis is on discrete-time systems, the last section explains how the same concepts can be used to analyze continuous-time LTI systems.

Study objectives

After studying this chapter you should be able to:

- Determine the steady-state response of LTI systems to sinusoidal, complex exponential, periodic, and aperiodic signals using the frequency response function.
- Understand the effects of ideal and practical LTI systems upon the input signal in terms of the shape of magnitude, phase, and group-delay responses.
- Understand how the locations of poles and zeros of the system function determine the shape of magnitude, phase, and group-delay responses of an LTI system.
- Develop and use algorithms for the computation of magnitude, phase, and group-delay responses of LTI systems described by linear constant-coefficient difference equations.
- Understand the important types of allpass and minimum-phase systems and their use in theoretical investigations and practical applications.

5.1 Sinusoidal response of LTI systems

In Section 3.1, we showed that the response of a LTI system to an everlasting exponential excitation is another everlasting exponential, that is,

$$x[n] = z^n \overset{\mathcal{H}}{\longmapsto} y[n] = H(z)z^n, \text{ all } n \tag{5.1}$$

where

$$H(z) \triangleq \sum_{k=-\infty}^{\infty} h[k]z^{-k} \tag{5.2}$$

is the system function, that is, the z-transform of the impulse response $h[n]$.

Eigenfunctions of LTI systems If the system is stable, the ROC of $H(z)$ contains the unit circle. In this case, we can evaluate (5.1) and (5.2) for $z = \mathrm{e}^{\mathrm{j}\omega}$. The result is

$$x[n] = \mathrm{e}^{\mathrm{j}\omega n} \overset{\mathcal{H}}{\longmapsto} y[n] = H(\mathrm{e}^{\mathrm{j}\omega})\mathrm{e}^{\mathrm{j}\omega n}, \text{ all } n \tag{5.3}$$

where

$$H(\mathrm{e}^{\mathrm{j}\omega}) \triangleq H(z)|_{z=\mathrm{e}^{\mathrm{j}\omega}} = \sum_{k=-\infty}^{\infty} h[k]\mathrm{e}^{-\mathrm{j}\omega k} \tag{5.4}$$

is the Fourier transform of the impulse response sequence. The system function $H(z)$ of a stable system, evaluated on the unit circle $z = \mathrm{e}^{\mathrm{j}\omega}$ and viewed as a function of ω, is known as the *frequency response* function of the system. From (5.3) we see that the complex exponentials $\mathrm{e}^{\mathrm{j}\omega n}$, $-\infty < n < \infty$, are eigenfunctions of LTI systems. The constant $H(\mathrm{e}^{\mathrm{j}\omega})$ for a specific value of ω is then the eigenvalue associated with the eigenfunction $\mathrm{e}^{\mathrm{j}\omega n}$.

The complex exponentials are the *only* eigenfunctions of LTI systems. Thus,

$$y[n] = H(\mathrm{e}^{\mathrm{j}\omega})x[n] \text{ if and only if } x[n] = \mathrm{e}^{\mathrm{j}\omega n}, \text{ all } n. \tag{5.5}$$

This property is meaningless for any other signal, including one sided or finite length complex exponentials. For the property (5.5) to be valid, the frequency response $H(\mathrm{e}^{\mathrm{j}\omega})$ must be well defined and finite. This is feasible only for stable systems; the frequency response is meaningless for unstable systems.

The frequency response is a complex function that can be expressed in either polar or rectangular form

$$H(\mathrm{e}^{\mathrm{j}\omega}) = |H(\mathrm{e}^{\mathrm{j}\omega})|\mathrm{e}^{\mathrm{j}\angle H(\mathrm{e}^{\mathrm{j}\omega})} = H_{\mathrm{R}}(\mathrm{e}^{\mathrm{j}\omega}) + \mathrm{j}H_{\mathrm{I}}(\mathrm{e}^{\mathrm{j}\omega}). \tag{5.6}$$

However, only the polar form reveals the physical meaning of the frequency response function. Indeed, using (5.3), the linearity property, and the polar notation, we obtain

$$x[n] = A\mathrm{e}^{\mathrm{j}(\omega n + \phi)} \overset{\mathcal{H}}{\longmapsto} y[n] = A|H(\mathrm{e}^{\mathrm{j}\omega})|\mathrm{e}^{\mathrm{j}[\omega n + \phi + \angle H(\mathrm{e}^{\mathrm{j}\omega})]}. \tag{5.7}$$

Therefore, *the response of a stable LTI system to a complex exponential sequence is a complex exponential sequence with the same frequency; only the amplitude and phase are changed by the system.*

Sinusoidal response of real LTI systems Suppose next that the input is a real sinusoidal sequence

$$x[n] = A_x \cos(\omega n + \phi_x) = \frac{A_x}{2} e^{j\phi_x} e^{j\omega n} + \frac{A_x}{2} e^{-j\phi_x} e^{-j\omega n}. \tag{5.8}$$

From (5.7), the response to the complex exponential $x_1[n] = \frac{A_x}{2} e^{j\phi_x} e^{j\omega n}$ is

$$y_1[n] = |H(e^{j\omega})| \frac{A_x}{2} e^{j\phi_x} e^{j[\omega n + \angle H(e^{j\omega})]}. \tag{5.9}$$

Similarly, the response to the complex exponential $x_2[n] = \frac{A_x}{2} e^{-j\phi_x} e^{-j\omega n}$ is

$$y_2[n] = |H(e^{-j\omega})| \frac{A_x}{2} e^{-j\phi_x} e^{j[-\omega n + \angle H(e^{-j\omega})]}. \tag{5.10}$$

Using the principle of superposition, we can easily see that

$$y[n] = \frac{A_x}{2} |H(e^{j\omega})| e^{j[\omega n + \phi_x + \angle H(e^{j\omega})]} + \frac{A_x}{2} |H(e^{-j\omega})| e^{j[-\omega n - \phi_x + \angle H(e^{-j\omega})]}. \tag{5.11}$$

If we assume that the impulse response $h[n]$ is real-valued, we have $|H(e^{-j\omega})| = |H(e^{j\omega})|$ and $\angle H(e^{-j\omega}) = -\angle H(e^{j\omega})$. Hence, (5.11) can be written as

$$y[n] = A_x |H(e^{j\omega})| \cos\left[\omega n + \phi_x + \angle H(e^{j\omega})\right]. \tag{5.12}$$

Therefore, we obtain the following *unique* property of LTI systems

$$x[n] = A_x \cos(\omega n + \phi_x) \xrightarrow{\mathcal{H}} y[n] = A_y \cos(\omega n + \phi_y), \tag{5.13}$$

where

$$A_y = |H(e^{j\omega})| A_x, \quad \phi_y = \angle H(e^{j\omega}) + \phi_x. \tag{5.14}$$

In conclusion, all an LTI system can do to a sinusoidal input is to scale its amplitude and change its phase; its frequency remains the *same*. If a system changes the frequency of a

sinusoidal input, it has to be nonlinear or time-varying. This property provides a convenient test to check whether a system is linear and time-invariant.

Gain and phase responses Since $A_y = |H(e^{j\omega})|A_x$, at frequency ω, the quantity $|H(e^{j\omega})|$ is known as the *magnitude response* or *gain* of the system. By the same token, since $\phi_y = \angle H(e^{j\omega}) + \phi_x$, $\angle H(e^{j\omega})$ is called the *phase response* of the system. Plots of $|H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ versus ω show at a glance how a system changes the amplitude and phase of input sinusoids at various frequencies. Therefore, $H(e^{j\omega})$ is known as the *frequency response* function of the system. When $|H(e^{j\omega})|$ is small at a frequency $\omega = \omega_0$, the component at this frequency is essentially removed, that is, "filtered out," from the input signal. For this reason, LTI systems are often called filters. However, it is more appropriate to use the term filter for LTI systems designed to remove some frequency components from the input signal. These ideas are illustrated in the following example.

Example 5.1   Illustration of frequency response function

Consider a stable system described by the first-order difference equation

$$y[n] = ay[n-1] + bx[n]. \quad -1 < a < 1 \tag{5.15}$$

To determine the frequency response function, we can assume a solution of the form of (5.3), substitute into the difference equation (5.15), and then solve for $H(e^{j\omega})$. Indeed, we have

$$H(e^{j\omega})e^{j\omega n} = aH(e^{j\omega})e^{j\omega(n-1)} + be^{j\omega n}. \tag{5.16}$$

Solving for $H(e^{j\omega})$, we obtain the formula

$$H(e^{j\omega}) = \frac{b}{1 - ae^{-j\omega}}. \tag{5.17}$$

Since $1 - ae^{-j\omega} = (1 - a\cos\omega) + ja\sin\omega$, it follows that

$$\left|1 - ae^{-j\omega}\right| = \sqrt{(1 - a\cos\omega)^2 + (a\sin\omega)^2} = \sqrt{1 + a^2 - 2a\cos\omega},$$

$$\angle(1 - ae^{-j\omega}) = \tan^{-1}\left[\frac{a\sin\omega}{1 - a\cos\omega}\right].$$

Therefore, the magnitude and phase responses are

$$|H(e^{j\omega})| = \frac{|b|}{\sqrt{1 - 2a\cos\omega + a^2}}, \tag{5.18}$$

$$\angle H(e^{j\omega}) = \angle b - \tan^{-1}\frac{a\sin\omega}{1 - a\cos\omega}. \tag{5.19}$$

It is customary to choose b so that the maximum of $|H(e^{j\omega})|$ equals one. If $a > 0$, the denominator of $|H(e^{j\omega})|$ attains its minimum at $\omega = 0$. Therefore, we require that
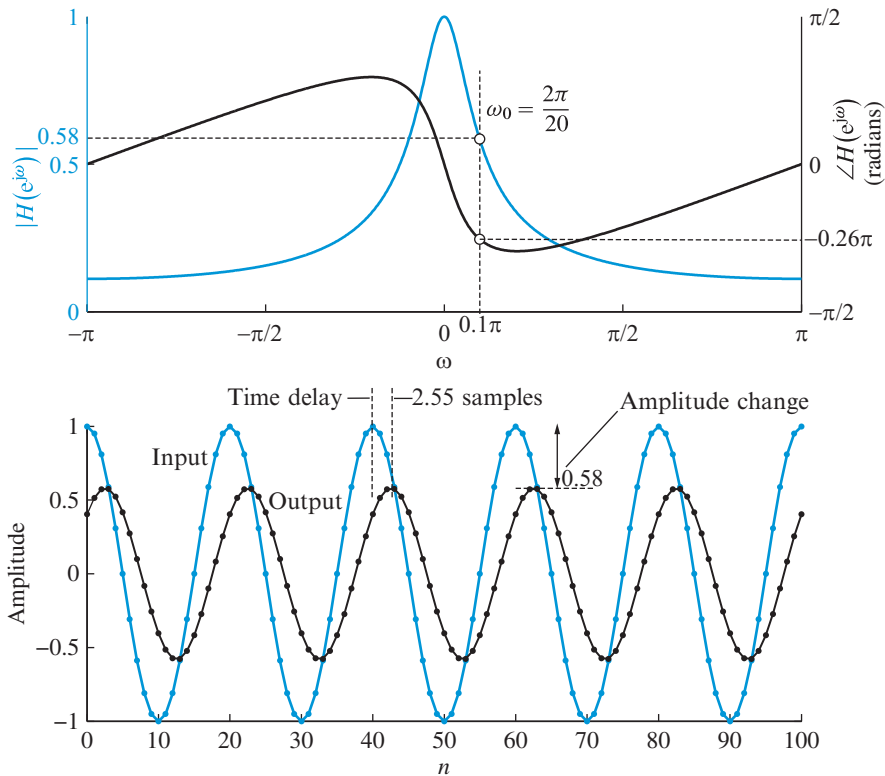
Figure 5.1 Magnitude and phase response functions and input–output signals for the LTI system defined by (5.15). The higher frequency suffers more attenuation than the lower frequency (lowpass filter).

$|H(e^{j0})| = |b|/(1-a) = 1$. This yields $b = \pm(1-a)$. If $a < 0$, the maximum of $|H(e^{j\omega})|$ occurs at $\omega = \pi$. By requiring that $|H(e^{j\pi})| = |b|/(1+a) = 1$, we obtain $b = \pm(1+a)$. Both cases can be satisfied by choosing

$$b = 1 - |a|, \tag{5.20}$$

which implies $|b| = 1 - |a|$ and $\angle b = 0$ because $-1 < a < 1$.

Figure 5.1 shows plots of magnitude and phase response functions for $a = 0.8$ and an input–output pair for the frequency $\omega = 2\pi/20$. We can clearly see that sinusoidal inputs with frequencies close to $\omega = 0$ pass with small attenuation; in contrast, sinusoids with frequencies close to $\omega = \pi$ are severely attenuated. Since for $a > 0$, $|H(e^{j\omega})|_{max}/|H(e^{j\omega})|_{min} = (1+a)/(1-a)$, the peak of the magnitude response becomes narrower as a approaches one. From the magnitude and phase response plots in Figure 5.1, the normalized gain at $\omega = 2\pi/20$ is about 0.58 while the phase shift is about -0.26π radians (or $-0.26\pi/\omega = -2.55$ samples). These values are evident from the input–output plots in Figure 5.1. ■

Example 5.2   Response to a linear FM signal

The procedure illustrated in Figure 5.1 provides the magnitude and phase response at a single frequency ω. If we repeat this process for various values of ω, we can compute the frequency response at any frequency interval of interest with the desired resolution.

However, it is possible to evaluate the magnitude response at several frequencies at once by using an input signal known as a *linear FM pulse*. The linear FM pulse is a sinusoidal sequence but with a frequency that grows linearly with time. To understand this concept, we recall that a constant frequency sinusoid can be considered as the real part of a complex rotating phasor

$$x(t) = A\cos(\Omega_0 t + \phi) = \mathcal{R}e\left\{Ae^{j(\Omega_0 t + \phi)}\right\}. \tag{5.21}$$

The total angle $\theta(t) = \Omega_0 t + \phi = 2\pi F_0 t + \phi$ changes linearly with time. The time derivative of the angle, which is the phasor's instantaneous rate of rotation in cycles per second, is equal to the constant frequency F_0, that is,

$$F_i(t) = \frac{1}{2\pi}\frac{d\theta(t)}{dt} = F_0. \tag{5.22}$$

Suppose now that the phase changes with time according to $\theta(t) = 2\pi F_0 t + \pi\beta t^2$. Then the instantaneous rate of rotation is given by

$$F_i(t) = \frac{1}{2\pi}\frac{d\theta(t)}{dt} = F_0 + \beta t. \tag{5.23}$$

We call $F_i(t)$ the *instantaneous frequency* of $x(t)$. The constant β in (5.23) provides the rate of frequency change. Thus, if $\beta = B/\tau$, the instantaneous frequency of the continuous-time signal

$$x(t) = A\cos(2\pi F_0 t + \pi\beta t^2), \quad 0 \le t \le \tau \tag{5.24}$$

increases from F_0 to $F_1 \triangleq F_0 + B$ Hz at a constant rate. Because this linear FM signal produces an audible sound similar to a siren or a chirp, it is also known as a *chirp signal* or simply a *chirp*. If we set $F_0 = 0$, sample $x(t)$ at a rate of $F_s = 1/T$, and choose τ so that $\tau = NT$, we obtain a discrete-time chirp signal

$$x[n] = x(nT) = A\cos(\pi\beta n^2 T^2) = A\cos(\pi\beta_d n^2), \quad 0 \le n \le N \tag{5.25}$$

where $\beta_d \triangleq (B/F_s)/N$. Since $f_1 = F_1/F_s = B/F_s$ is the maximum attainable frequency in cycles per sampling interval, the quantity β_d is referred to as the rate of frequency change per sampling interval.

Figure 5.2 shows the response $y[n]$ of the system (5.15) with $a = 0.8$ to a chirp signal $x[n]$ with $A = 1$, $B = 10$ Hz, $\tau = 10$ s, and $F_s = 100$ Hz. The magnitude response of the system is shown from zero to B Hz, which is the frequency range spanned by the input chirp. Since this frequency sweep takes place at the time interval from zero to τ seconds,
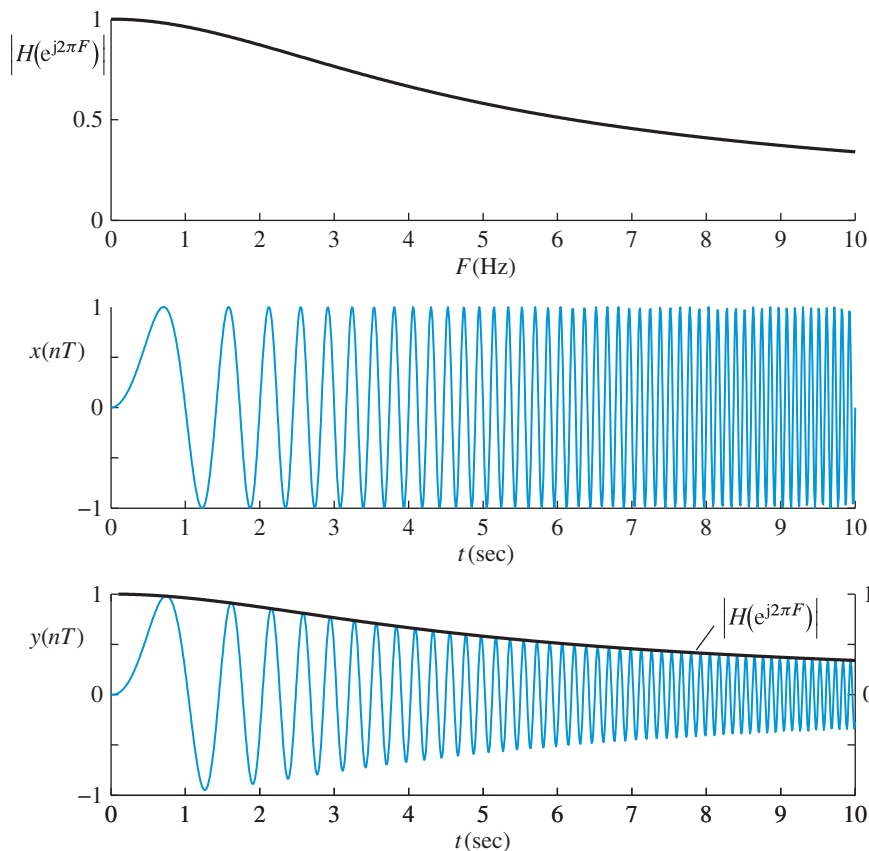
Figure 5.2 Evaluation of the magnitude response of an LTI system using a linear FM (chirp) input signal.

we superimpose the magnitude response on the output signal using a dual axis plot. This is possible because the magnitude response is normalized from zero to one and the maximum amplitude of the chirp is equal to one. We note that the magnitude response coincides with the envelope of the output signal, that is, the amplitude of the input chirp is attenuated according to the value of the instantaneous frequency specified by (5.23). More details are provided in Tutorial Problem 3. Chirp signals are used in radar systems and seismic exploration.   ∎

Continuous and principal phase functions When we deal with the complex exponential function, there are two key observations to bear in mind:

1. The determination of phase function has an intrinsic ambiguity because

$$H(e^{j\omega}) = |H(e^{j\omega})|e^{j\angle H(e^{j\omega})} = |H(e^{j\omega})|e^{j[\angle H(e^{j\omega})+2m\pi]} \tag{5.26}$$

for any integer m. This is consistent with the fact that a sinusoidal signal shifted a multiple number of periods is indistinguishable from the original signal.

2. Numerical algorithms compute the principal value of the phase, which is always within the following range

$$-\pi < \text{ARG}\left[H(e^{j\omega})\right] \leq \pi. \qquad (5.27)$$

If the phase response exceeds the limits in (5.27), the function $\text{ARG}\left[H(e^{j\omega})\right]$ is discontinuous. The discontinuities introduced by (5.27) are jumps of $2m\pi$ radians, where m is an integer. These observations are explained in Example 5.3.

Example 5.3   Phase functions

For example, the frequency response of the system $H(z) = [(1 + z^{-1})/2]^6$ is $H(e^{j\omega}) = \cos^6(\omega/2)e^{-j3\omega}$. Therefore, its phase response, $\Psi(\omega) = -3\omega$, varies continuously from 0 to -6π as ω changes from 0 to 2π. However, if we evaluate $\angle H(e^{j\omega})$ using the MATLAB function angle (see Section 4.5.2 on page 173) we obtain the piecewise linear curve with jumps of 2π at $\omega = \pi/3$, 4π at $\omega = \pi$, and 6π at $\omega = 5\pi/3$ (see Tutorial Problem 6). The symbol $\angle H(e^{j\omega})$ is used to denote the phase response function of a system, in general. We shall reserve the notation $\Psi(\omega)$ for the continuous or unwrapped phase function. However, the principal value of phase is sufficient for most practical applications. ∎

Steady-state and transient response The eigenfunction property (5.3) holds if the input sequence $x[n]$ is a complex exponential sequence that exists over the entire interval $-\infty < n < \infty$. However, in practice every input starts at a finite time. To see the implications of this restriction, consider a complex exponential starting at time $n = 0$, that is,

$$x[n] = e^{j\omega n}u[n]. \qquad (5.28)$$

The response of a causal system ($h[n] = 0, n < 0$) to the input (5.28) is

$$
\begin{aligned}
y[n] &= \sum_{k=0}^{n} h[k]x[n-k] = \sum_{k=0}^{n} h[k]e^{j\omega(n-k)} \\
&= \left(\sum_{k=0}^{\infty} h[k]e^{-j\omega k}\right)e^{j\omega n} - \left(\sum_{k=n+1}^{\infty} h[k]e^{-j\omega k}\right)e^{j\omega n} \\
&= \underbrace{H(e^{j\omega})e^{j\omega n}}_{y_{ss}[n]} - \underbrace{\left(\sum_{k=n+1}^{\infty} h[k]e^{-j\omega k}\right)e^{j\omega n}}_{y_{tr}[n]}.
\end{aligned}
\qquad (5.29)
$$

The term $y_{tr}[n]$ is known as the transient response (see Section 2.10). If the system is stable, we have

$$|y_{tr}[n]| \leq \sum_{k=n+1}^{\infty} |h[k]| \leq \sum_{k=0}^{\infty} |h[k]| < \infty,$$

which shows that the transient response becomes progressively smaller as $n \to \infty$ because fewer and smaller samples of the impulse response are included in the summation. For an

FIR system with $h[n] = 0$ for $n > M$, the transient response vanishes for $n > M$. Therefore, for large values of n the transient response of a stable system decays towards zero leaving only the steady-state response, that is,

$$\lim_{n \to \infty} y[n] = H(e^{j\omega})e^{j\omega n} = y_{ss}[n]. \tag{5.30}$$

Therefore, in practice, the eigenfunction property (5.3) holds after the transient response has diminished. A simple illustration of the difference between transient and steady-state response is provided in Example 5.4. We emphasize that in most signal processing applications, we are mainly interested in the steady-state response of a system.

Example 5.4   Steady-state and transient responses

Consider a causal and stable system described by the impulse response $h[n] = 0.8^n u[n]$. We will compute and plot the response $y[n]$ of the system to the input $x[n] = \cos(0.05\pi n)u[n]$.

Using the z-transform approach and Table 3.2, we have

$$H(z) = \mathcal{Z}\{h[n]\} = \frac{1}{1 - 0.8z^{-1}}, \quad |z| > 0.8$$

$$X(z) = \mathcal{Z}\{x[n]\} = \frac{1 - \cos(0.05\pi)z^{-1}}{1 - 2\cos(0.05\pi)z^{-1} + z^{-2}}. \quad |z| > 1$$

Hence the z-transform of the response $y[n]$ is given by

$$Y(z) = H(z)X(z) = \frac{1 - \cos(0.05\pi)z^{-1}}{(1 - 0.8z^{-1})[1 - 2\cos(0.05\pi)z^{-1} + z^{-2}]}$$

$$= \frac{\frac{0.8[0.8 - \cos(0.05\pi)]}{0.8^2 - 2(0.8)\cos(0.05\pi) + 1}}{1 - 0.8z^{-1}} + \frac{\frac{e^{j0.05\pi}}{2(e^{j0.05\pi} - 0.8)}}{1 - e^{j0.05\pi}z^{-1}} + \frac{\frac{e^{-j0.05\pi}}{2(e^{-j0.05\pi} - 0.8)}}{1 - e^{-j0.05\pi}z^{-1}}$$

$$= \frac{-2.5151}{1 - 0.8z^{-1}} + \frac{\frac{1}{2}H(z)|_{z=e^{j0.05\pi}}}{1 - e^{j0.05\pi}z^{-1}} + \frac{\frac{1}{2}H(z)|_{z=e^{-j0.05\pi}}}{1 - e^{-j0.05\pi}z^{-1}}, \quad |z| > 1$$

$$= \frac{-2.5151}{1 - 0.8z^{-1}} + \frac{\frac{1}{2}H(e^{j0.05\pi})}{1 - e^{j0.05\pi}z^{-1}} + \frac{\frac{1}{2}H(e^{-j0.05\pi})}{1 - e^{-j0.05\pi}z^{-1}}. \quad |z| > 1$$

After inverse transformation, we have

$$y[n] = -2.5151(0.8)^n u[n] + \frac{1}{2}H(e^{j0.05\pi})e^{j0.05\pi n}u[n]$$

$$+ \frac{1}{2}H(e^{-j0.05\pi})e^{-j0.05\pi n}u[n]$$

$$= -2.5151(0.8)^n u[n] + \mathcal{R}e\left\{|H(e^{j0.05\pi})|e^{j\angle H(e^{j0.05\pi})}\right\}u[n]$$

$$= -2.5151(0.8)^n u[n] + |H(e^{j0.05\pi})|\cos[0.05\pi n + \angle H(e^{j0.05\pi})]u[n]$$

$$= \underbrace{-2.5151(0.8)^n u[n]}_{y_{tr}[n]} + \underbrace{4.0928\cos(0.05\pi n - 0.5377)u[n]}_{y_{ss}[n]}.$$
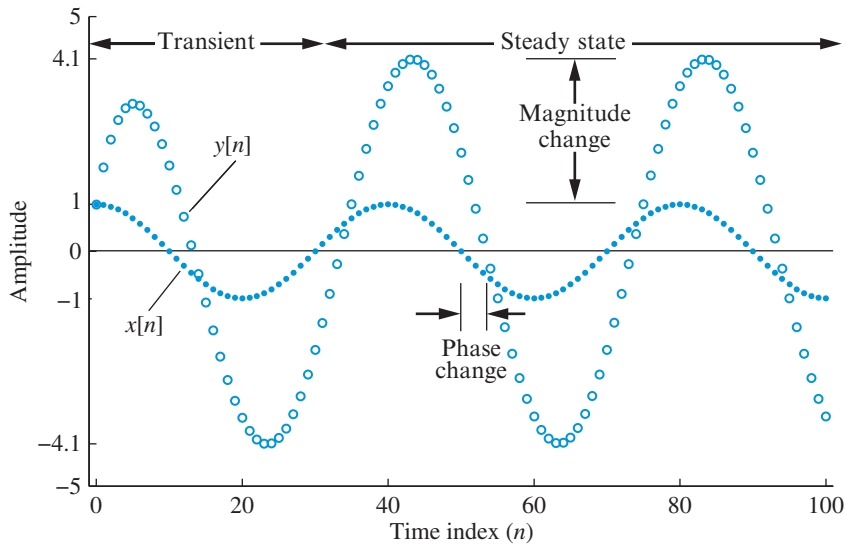
Figure 5.3 Transient and steady-state responses for sinusoidal excitation $x[n] = \cos(0.05\pi n)$.

As expected, the transient response decays over time and the steady-state response continues as a scaled and time-shifted sinusoidal signal. This response can be computed in MATLAB in one of two ways: (a) by generating long $h[n]$ and $x[n]$ sequences and then using the `y=conv(h,x)` function or (b) first converting $h[n]$ into a difference equation, which for this example is $y[n] = 0.8y[n-1] + x[n]$ and then using the `y=filter(1,[1,-0.8],x)` function. The resulting input–output signal pair is shown in Figure 5.3 which clearly shows the transient and steady-state parts of the output $y[n]$ including the magnitude gain and the phase shift in the response.   ■

5.2   Response of LTI systems in the frequency domain

Since every signal can be represented by a superposition of sinusoidal components, as we discussed in Chapter 4, the frequency response provides a simple and intuitive way to determine and understand what an LTI system does to the input signal sequence. Furthermore, the frequency response leads to a simple relationship between the spectra of input and output signals of LTI systems. The form of this relationship depends on whether the input sequence is periodic or aperiodic.

5.2.1   Response to periodic inputs

Consider a periodic input $x[n] = x[n + N]$ with fundamental period N. The sequence $x[n]$ can be expressed as a sum of complex exponentials using the IDTFS

$$x[n] = \sum_{k=0}^{N-1} c_k^{(x)} e^{j \frac{2\pi}{N} kn}. \tag{5.31}$$

Using the eigenfunction property (5.3) and the linearity property, we have

$$e^{j\frac{2\pi}{N}kn} \overset{\mathcal{H}}{\longmapsto} H(e^{j\frac{2\pi}{N}k})e^{j\frac{2\pi}{N}kn},$$

$$x[n] = \sum_{k=0}^{N-1} c_k^{(x)} e^{j\frac{2\pi}{N}kn} \overset{\mathcal{H}}{\longmapsto} \sum_{k=0}^{N-1} c_k^{(x)} H(e^{j\frac{2\pi}{N}k})e^{j\frac{2\pi}{N}kn} = y[n].$$

From the last equation, we deduce that the output sequence is periodic with Fourier coefficients $c_k^{(y)}$ given by

$$c_k^{(y)} = H(e^{j\frac{2\pi}{N}k})c_k^{(x)}, \quad -\infty < k < \infty. \tag{5.32}$$

Therefore, *the response of an LTI system to a periodic input sequence is a periodic sequence with the same fundamental period*. This should not be a surprise because LTI systems *cannot* alter the frequencies of the input signals; they can only change their amplitude and phase. From (5.32), we have

$$|c_k^{(y)}| = |H(e^{j\frac{2\pi}{N}k})||c_k^{(x)}|, \tag{5.33}$$

$$\angle c_k^{(y)} = \angle H(e^{j\frac{2\pi}{N}k}) + \angle c_k^{(x)}. \tag{5.34}$$

These relations are essentially equations (5.14) applied to each frequency component of the periodic input signal.

Using (5.33) and Parseval's theorem (4.69), we find that the power of the output sequence is

$$P_y = \frac{1}{N} \sum_{n=0}^{N-1} |y[n]|^2 = \sum_{k=0}^{N-1} |c_k^{(y)}|^2 = \sum_{k=0}^{N-1} |H(e^{j\frac{2\pi}{N}k})|^2 |c_k^{(x)}|^2. \tag{5.35}$$

The following example illustrates the use and meaning of (5.32) in the implementation and analysis of LTI systems with periodic inputs.

Example 5.5   Zero-state and steady-state responses
Consider the first-order system described in Example 5.1

$$y[n] = ay[n-1] + (1 - |a|)x[n], \quad y[-1] = 0. \tag{5.36}$$

The system is excited by a periodic sequence, with fundamental period $N = 10$, given by

$$x[n] = \begin{cases} 1, & 0 \le 0 < 6 \\ 0. & 6 \le n < 10 \end{cases} \tag{5.37}$$

We will compute the output of the system in the time-domain using the difference equation (5.36) as well as in the frequency-domain using (5.32), for $a = 0.7$ and $a = 0.9$, and compare the resulting output signals.

In the time-domain the output is computed using the difference equation (5.36), which is implemented by the MATLAB function `filter`, with zero initial conditions. The result is the zero-state response $y_{zs}[n]$ of the system.

In the frequency-domain, based on (5.32), we use the following procedure:

1. Use the function `dtfs` to compute the DTFS

$$c_k^{(x)} = \frac{1}{N} \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}, \quad 0 \le k \le N - 1$$

2. Compute gain values $H(e^{j\frac{2\pi}{N}k}), 0 \le k \le N - 1$ using (5.17).
3. Compute the DTFS $c_k^{(y)} = H(e^{j\frac{2\pi}{N}k}) c_k^{(x)}, 0 \le k \le N - 1$
4. Use the function `idtfs` to compute the inverse DTFS

$$y_{ss}[n] = \sum_{k=0}^{N-1} c_k^{(y)} e^{j\frac{2\pi}{N}kn}. \quad 0 \le n \le N - 1$$

According to the eigenfunction property (5.3), this approach provides the steady-state response $y_{ss}[n]$ of the system.

Figures 5.4 and 5.5 show the input, zero-state response, steady-state response, and impulse response of the system. We first note that the impulse response becomes essentially zero after a certain index $n = M$. This value determines the "memory" of the system, because the system uses only the M most recent values of the input to determine the current value of the output. Basically, the system "forgets" after M samples. In Figure 5.4, $M \approx 10$; therefore, the transient response dies after 10 samples and $y_{zs}[n] = y_{ss}[n]$ for $n > 10$. The system in Figure 5.5 has longer memory ($M \approx 40$) and the transient response, as expected, lasts longer. Therefore, for all practical purposes, we can compute the response of a stable system to a periodic input either in the time-domain or in the frequency-domain. This idea is the cornerstone for the implementation of discrete-time systems in the frequency-domain (see Chapter 7). ∎

5.2.2   Response to aperiodic inputs

Aperiodic sequences can be expressed as a "continuous" superposition of complex exponentials, using the inverse DTFT, as follows

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega}) e^{j\omega n} d\omega = \lim_{\substack{\Delta\omega \to 0 \\ k\Delta\omega \to \omega}} \frac{1}{2\pi} \sum_k X(e^{jk\Delta\omega}) e^{j(k\Delta\omega)n} \Delta\omega. \tag{5.38}$$

Using the eigenfunction property (5.3) and the superposition principle of LTI systems, the response $y[n]$ to the input (5.38) is
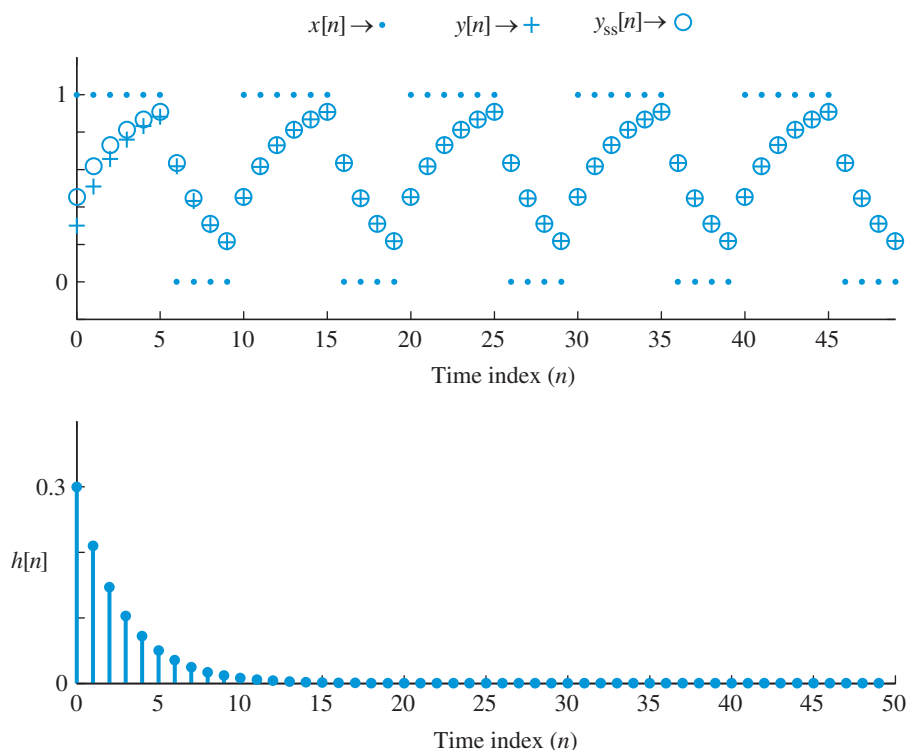
Figure 5.4 Zero-state and steady-state responses of a "short-memory" first-order system to a periodic pulse train sequence. The impulse response $h[n]$ is essentially zero for $n > 10$.

$$y[n] = \lim_{\substack{\Delta\omega \to 0 \\ k\Delta\omega \to \omega}} \frac{1}{2\pi} \sum_k H(e^{jk\Delta\omega})X(e^{jk\Delta\omega})\, e^{j(k\Delta\omega)n}\, \Delta\omega$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} H(e^{j\omega})X(e^{j\omega})\, e^{j\omega n}\, d\omega. \tag{5.39}$$

Therefore, we conclude that the Fourier transform of the output sequence is

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}). \tag{5.40}$$

This heuristic derivation parallels the approach for periodic sequences. A formal derivation is given by the convolution theorem (4.149). Also (5.40) can be obtained by evaluating (3.63) on the unit circle.

If we express the Fourier transforms in (5.40) in polar notation, we obtain

$$\left|Y(e^{j\omega})\right| = \left|H(e^{j\omega})\right|\left|X(e^{j\omega})\right|, \tag{5.41}$$

$$\angle Y(e^{j\omega}) = \angle H(e^{j\omega}) + \angle X(e^{j\omega}). \tag{5.42}$$
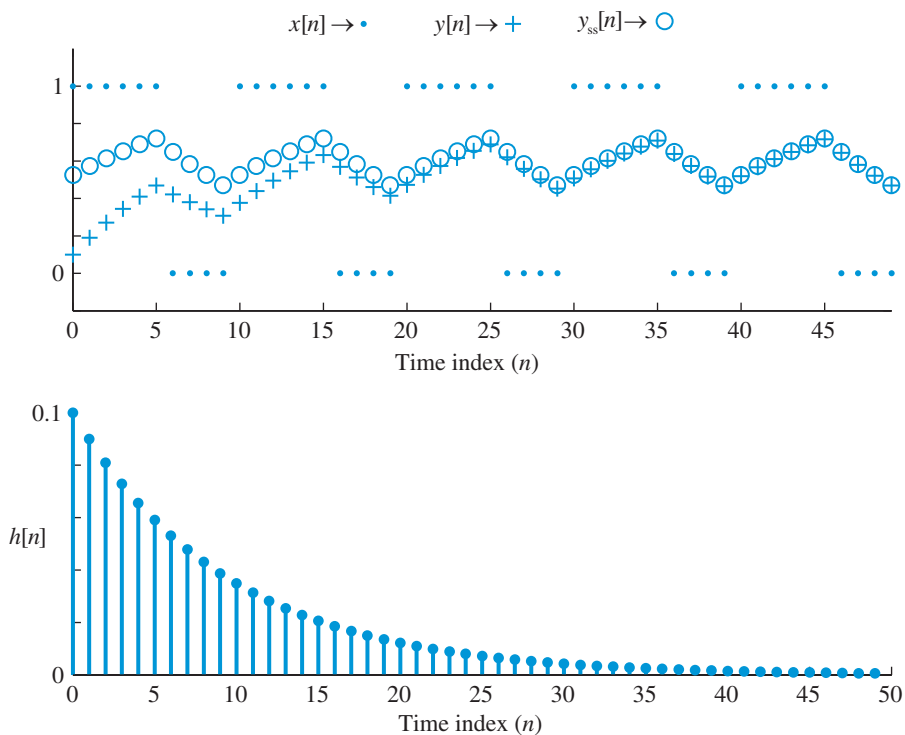
Figure 5.5 Zero-state and steady-state responses of a "long-memory" first-order system to a periodic pulse train sequence. The impulse response $h[n]$ is essentially zero for $n > 40$.

We note that (5.41) and (5.42) are essentially equations (5.14) applied to each frequency component of the aperiodic input signal.

From (5.32) and (5.40) we see that frequency components of the input are *suppressed* from the output if $|H(e^{j\omega})|$ is *small* at those frequencies. This property provides the basis for the design of frequency-selective filters.

From (5.40) and Parseval's theorem (4.94), the energy of the output sequence is

$$E_y = \sum_{n=-\infty}^{\infty} |y[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left|X(e^{j\omega})\right|^2 \left|H(e^{j\omega})\right|^2 d\omega. \tag{5.43}$$

In general, due to the continuity of ω we *cannot* use (5.40) to compute the output $y[n]$ from the input $x[n]$, as in Example 5.5. In Chapter 7 we will show that an exception is possible for the important case of FIR systems.

5.2.3   Energy or power gain

From (5.43) and (5.35) we see that $|H(e^{j\omega})|^2$ shows how the system transfers energy or power from the input signal to the output signal. To emphasize this interpretation, we often refer to $|H(e^{j\omega})|^2$ as the (energy or power) *gain* of the system. Since the gain may take very

large values, it is convenient to express the gain in a logarithmic unit, known as *decibel* (dB), using the formula

$$\text{Gain in dB} = |H(\text{e}^{\text{j}\omega})|_{\text{dB}} \triangleq 10 \log_{10} |H(\text{e}^{\text{j}\omega})|^2. \tag{5.44}$$

We note that zero dB corresponds to a value of $|H(\text{e}^{\text{j}\omega})| = 1$. If $|H(\text{e}^{\text{j}\omega})| = 2^m$, then $|H(\text{e}^{\text{j}\omega})|_{\text{dB}} \approx 6m$ dB, that is, each time we double the magnitude response we increase the gain by 6 dB. When $|H(\text{e}^{\text{j}\omega})| < 1$, instead of gain we have *attenuation*; in this case the gain $|H(\text{e}^{\text{j}\omega})|_{\text{dB}}$ is negative.

Another advantage of using logarithmic units is that the multiplicative relations (5.33) and (5.41) become additive, that is,

$$|c_k^{(y)}|_{\text{dB}} = \left|H(\text{e}^{\text{j}\frac{2\pi}{N}k})\right|_{\text{dB}} + |c_k^{(x)}|_{\text{dB}}, \tag{5.45}$$

$$\left|Y(\text{e}^{\text{j}\omega})\right|_{\text{dB}} = \left|H(\text{e}^{\text{j}\omega})\right|_{\text{dB}} + \left|X(\text{e}^{\text{j}\omega})\right|_{\text{dB}}. \tag{5.46}$$

Thus, the effects of both magnitude and phase responses become additive.

5.3   Distortion of signals passing through LTI systems

An LTI system changes the input signal $x[n]$ into an output signal $y[n]$. The nature of this change can be understood by examining the frequency response of the system. Indeed, the system changes the relative magnitudes and phases of the frequency components in an input signal in a way dictated by its frequency response function. These changes may be either desirable, that is, the input signal is modified in a useful way, or undesirable, that is, the input signal is subject to distortion. In this section, we formulate the conditions for systems with a distortionless response and discuss the types of distortion that result when these conditions are violated.

Distortionless response systems A system has distortionless response if the input signal $x[n]$ and the output signal $y[n]$ have the same "shape." This is possible if the input and output signals satisfy the condition

$$y[n] = Gx[n - n_{\text{d}}], \quad G > 0 \tag{5.47}$$

where G and n_{d} are constants. Taking the Fourier transform of both sides, we have

$$Y(\text{e}^{\text{j}\omega}) = G\text{e}^{-\text{j}\omega n_{\text{d}}}X(\text{e}^{\text{j}\omega}). \tag{5.48}$$

From (5.40) and (5.48), the frequency response function is

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = Ge^{-j\omega n_d}. \tag{5.49}$$

From this equation it follows that

$$|H(e^{j\omega})| = G, \tag{5.50}$$

$$\angle H(e^{j\omega}) = -\omega n_d. \tag{5.51}$$

This result shows that for a LTI system to have a distortionless response, the magnitude response $|H(e^{j\omega})|$ must be a constant and the phase response $\angle H(e^{j\omega})$ must be a linear function of ω with slope $-n_d$, where n_d is the delay of the output with respect to the input. We emphasize that the phase response should not only be a linear function of frequency, but it should also pass through the origin $\omega = 0$.

If the slope α of a linear-phase response function is not an integer n_d, that is, $H(e^{j\omega}) = Ge^{-j\omega\alpha}$, relation (5.47) has no formal meaning because we can only shift $x[n]$ by an integer number of samples. However, if $x[n] = x_c(nT)$ and $y[n] = y_c(nT)$, then $y_c(t) = Gx_c(t - \alpha T)$. The meaning of *fractional delay* is further discussed in Chapter 12.

Magnitude distortion We say that a system introduces *magnitude distortion* if

$$|H(e^{j\omega})| \neq G. \tag{5.52}$$

In words, the system distorts the input signal by changing the "correct proportion" of the input frequency components. Systems without magnitude distortion, that is, systems that satisfy (5.50), are known as *allpass* systems. Allpass systems have a "flat" magnitude response and their characteristics are completely determined by the phase response. While the frequency domain description of magnitude distortion is easy, its effects on the shape of the signal are far less obvious. To illustrate this point, consider the simple test signal

$$x[n] = \cos(\omega_0 n) - \tfrac{1}{3}\cos(3\omega_0 n) + \tfrac{1}{5}\cos(5\omega_0 n), \tag{5.53}$$

which is an approximation of a rectangular pulse train. Suppose now that a system $H_i(e^{j\omega})$ with input $x[n]$ produces an output signal $y_i[n]$ given by

$$y_i[n] = c_1\cos(\omega_0 n + \phi_1) + c_2\cos(3\omega_0 n + \phi_2) + c_3\cos(5\omega_0 n + \phi_3). \tag{5.54}$$

Figure 5.6(a) shows the signals $x[n]$, $y_1[n]$, and $y_2[n]$ obtained for $\omega_0 = 0.004\pi$ rads and the following amplitudes and phases:

Signal	c_1	c_2	c_3	ϕ_1	ϕ_2	ϕ_3	Amplitude
$x[n]$	1	$-1/3$	$1/5$	0	0	0	original
$y_1[n]$	1/4	$-1/3$	$1/5$	0	0	0	highpass
$y_2[n]$	1	$-1/6$	$1/10$	0	0	0	lowpass
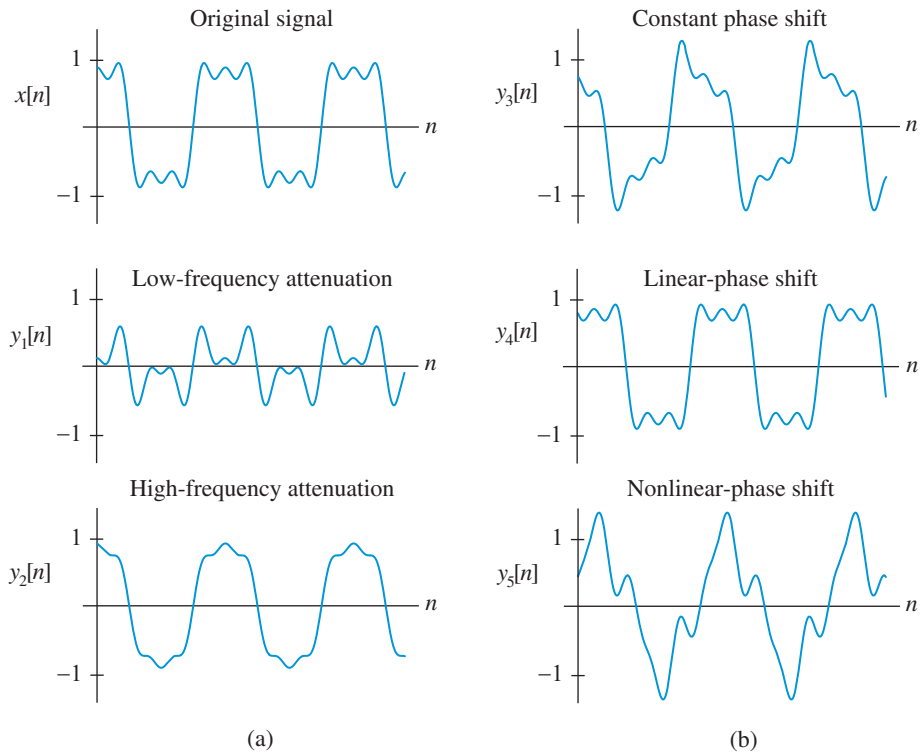
Figure 5.6 Magnitude (a) and phase (b) distortions. Clearly, it is difficult to distinguish the effects of magnitude and phase distortion.

We note that if a system attenuates the low-frequency component c_1 to $1/4$, the resulting signal $y_1[n]$ becomes "sharper." In contrast, attenuating the high-frequency components in $y_2[n]$ results in a "smoother" signal. However, we cannot predict the extent of sharpening or smoothing without computing the output signal.

Phase or delay distortion If the phase response is not a linear function of frequency, that is,

$$\angle H(e^{j\omega}) \neq -\omega n_d, \tag{5.55}$$

the resulting distortion is known as *phase* or *delay distortion*.

The phase response $\angle H(e^{j\omega})$ gives the phase shift (in radians) experienced by each sinusoidal component of the input signal. If we rewrite (5.12) as

$$y[n] = A_x |H(e^{j\omega})| \cos[\omega n + \phi_x + \angle H(e^{j\omega})] \tag{5.56}$$

$$= A_x |H(e^{j\omega})| \cos\left\{\omega\left[n + \frac{\phi_x}{\omega} + \frac{\angle H(e^{j\omega})}{\omega}\right]\right\}, \tag{5.57}$$

we note that the quantity $\angle H(\mathrm{e}^{\mathrm{j}\omega})/\omega$ shows the time shift (in number of sampling intervals) experienced by each sinusoidal component of the input signal. Therefore, sometimes it is more meaningful to use the *phase delay* defined by

$$\tau_{\mathrm{pd}}(\omega) \triangleq -\frac{\angle H(\mathrm{e}^{\mathrm{j}\omega})}{\omega}. \tag{5.58}$$

To illustrate the difference between constant phase shift and constant time delay, we consider again the signal (5.53). We now consider an allpass system that changes the input signal phase as shown in the list below.

Signal	c_1	c_2	c_3	ϕ_1	ϕ_2	ϕ_3	Phase shift
$x[n]$	1	$-1/3$	$1/5$	0	0	0	zero
$y_3[n]$	1	$-1/3$	$1/5$	$\pi/6$	$\pi/6$	$\pi/6$	constant
$y_4[n]$	1	$-1/3$	$1/5$	$-\pi/4$	$-3\pi/4$	$-5\pi/4$	linear
$y_5[n]$	1	$-1/3$	$1/5$	$-\pi/3$	$\pi/4$	$\pi/7$	nonlinear

These phase distorted signals are shown in Figure 5.6(b). We note that the constant phase shift in $y_3[n]$ causes distortion because each frequency component is delayed by a different amount. In contrast, the linear-phase shift in $y_4[n]$ does not cause any distortion because it results in a constant phase delay $\tau_{\mathrm{pd}}(\omega) = 62.5$ sampling intervals. The arbitrary nonlinear-phase shift in $y_5[n]$ results in a more drastic change of the input signal shape. In most cases magnitude and phase distortions are simultaneously present, and it is difficult if not impossible to separate their effects.

We conclude that for distortionless transmission it is not enough that the system amplifies (or attenuates) all frequency components equally. All these frequency components must also undergo an identical time delay in order to add up correctly. This demands a constant phase delay, that is, a phase shift proportional to frequency. Nonlinear-phase responses may lead to severe shape alterations.

Group delay A convenient way to check the linearity of phase response is to use the *group delay*, defined as the negative of the slope of the phase as follows:

$$\tau_{\mathrm{gd}}(\omega) \triangleq -\frac{\mathrm{d}\Psi(\omega)}{\mathrm{d}\omega}. \tag{5.59}$$

The derivative in this definition requires that the phase response is a continuous function of frequency. Therefore, to compute the group delay, we should use the unwrapped phase response $\Psi(\omega)$. The continuous phase can be obtained from the group delay by integration as

$$\Psi(\omega) = -\int_0^\omega \tau_{\mathrm{gd}}(\theta)\mathrm{d}\theta + \Psi(0). \tag{5.60}$$

For real systems, $\Psi(0) = 0$ because $\Psi(\omega)$ has odd symmetry. Phase responses which are linear in frequency correspond to constant phase delay and constant group delay; both delays are identical, and each may be interpreted as time delay. If the phase response is nonlinear, then the relative phase of each frequency component is delayed by a different amount resulting in severe shape distortions.

We note that both the linear-phase response $\angle H(e^{j\omega}) = -\omega n_d$ and the *generalized linear-phase* response

$$\angle H(e^{j\omega}) = \theta_0 - \omega n_d \tag{5.61}$$

have a constant group delay. Thus, constant group delay is a more relaxed condition than constant phase delay.

To better illustrate the difference between phase and group delay, consider a bandpass signal obtained by modulating a lowpass signal such as

$$x[n] = s[n] \cos \omega_c n, \tag{5.62}$$

where $s[n]$ is a lowpass signal with maximum frequency $\omega_m \ll \omega_c$ (see Section 4.5.3). If the phase response $\Psi(\omega)$ around $\omega = \omega_c$ is approximately linear, it can be expressed using a Taylor's series expansion by

$$\Psi(\omega) \approx \Psi(\omega_c) + \left.\frac{d\Psi(\omega)}{d\omega}\right|_{\omega=\omega_c} (\omega - \omega_c)$$

$$= -\tau_{pd}(\omega_c)\omega_c - \tau_{gd}(\omega_c)(\omega - \omega_c), \tag{5.63}$$

where we have used (5.58) and (5.59). Using equations (5.62) and (5.63), it can be shown that (see Tutorial Problem 12 and Papoulis 1977)

$$y[n] \approx \left| H(e^{j\omega_c}) \right| s[n - \tau_{gd}(\omega_c)] \cos\{\omega_c[n - \tau_{pd}(\omega_c)]\}. \tag{5.64}$$

From (5.64) we see that the group delay evaluated at the carrier frequency ω_c is the delay of the envelop $s[n]$ of the input and the phase delay is equal to the delay of the carrier. The name group delay comes because $\tau_{gd}(\omega_c)$ shows the delay of the "bundle" (group) of frequency components about ω_c. If (5.63) is not true, then the output is no longer given by (5.64). These concepts are illustrated in the following example.

Example 5.6   Magnitude and group delay distortions
Consider a filter with system function

$$H(z) = \frac{b_0}{\left[1 - 2r\cos(\omega_0)\, z^{-1} + r^2 z^{-2}\right]^K}. \tag{5.65}$$

Figure 5.7(a) shows the magnitude and group delay responses of this filter with $r = 0.9$, $\omega_0 = \pi/3$, and $K = 8$. The coefficient b_0 is chosen to assure a maximum gain of $0\,dB$. The input signal $x[n]$ consists of two consecutive narrowband Gaussian pulses followed by
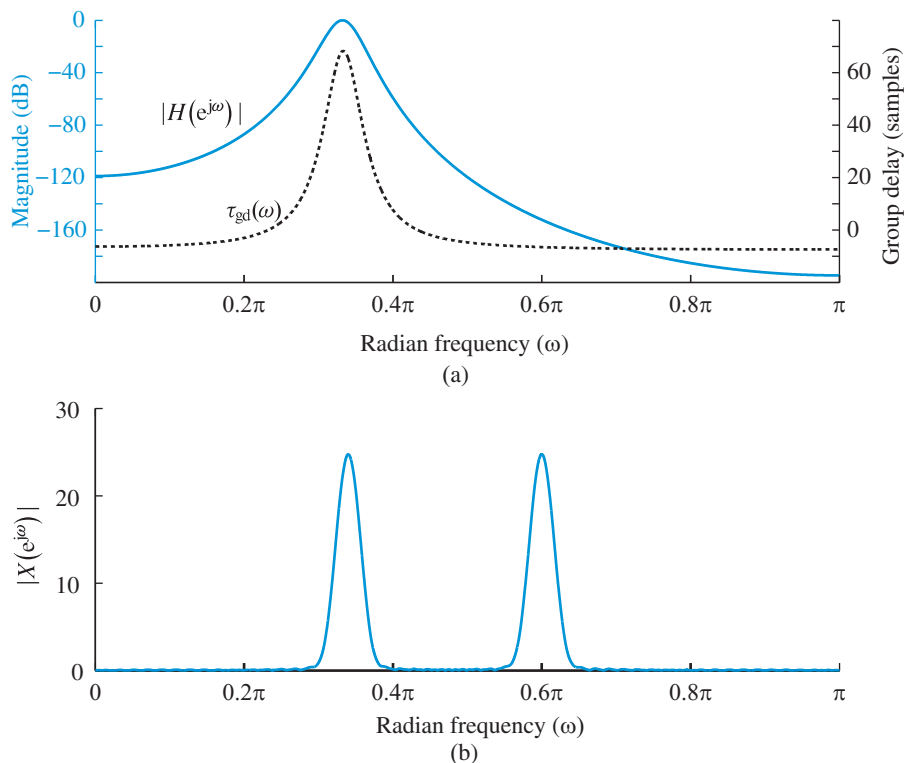
Figure 5.7 Magnitude and group-delay response (a), and spectrum (b), for the filter and bandpass input signal used in Example 5.6.

a trail of zeros. To create this signal, we first compute $N = 100$ samples of a Gaussian pulse

$$s(t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{1}{2}\frac{(t-\mu)^2}{\sigma^2}\right\}, \tag{5.66}$$

with $\mu = 0$ and $\sigma = 2$ in the range $-5 \le t \le 5$. These values are used to define a sequence $s[n]$, $0 \le n \le N-1$. The two modulated pulses are generated by $s[n]\cos(\omega_1 n)$ and $s[n]\cos(\omega_2 n)$, where $\omega_1 = 0.34\pi$, and $\omega_2 = 0.6\pi$. The spectrum of $x[n]$ is shown in Figure 5.7(b). The filter input and output signals are shown in Figure 5.8. The first pulse, which is centered at the passband of the filter, passes through with a group or envelope delay of about 50 samples. The attenuation and smearing of the envelope is due to the magnitude distortion of the filter. We note that the pulse centered at ω_2 is attenuated by more than 100 dB and it does not appear in the output. More details are given in Tutorial Problem 11.   ■

Interestingly enough, the human ear is insensitive to small or moderate delay distortion; thus, delay distortion is seldom a concern in voice and music storage and transmission. In contrast, the human eye is sensitive to phase distortion but it is relatively insensitive to

Input signal
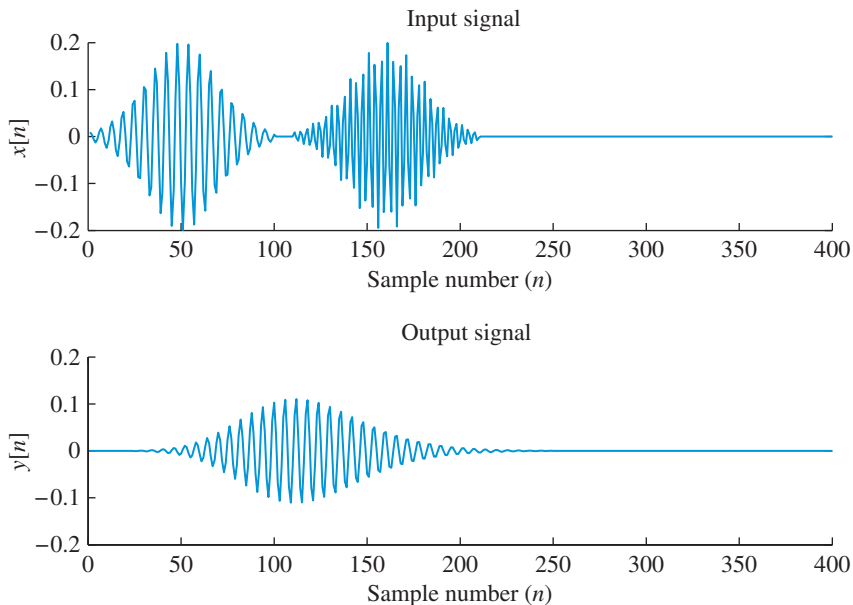
Output signal

Figure 5.8 Input and output signals for the filter in Example 5.6.

magnitude distortion. Finally, delay distortion can be critical in pulse transmission, where the shape of the transmitted pulses carries important information.

5.4 Ideal and practical filters

Systems that are designed to pass some frequency components without significant distortion while severely or completely eliminating others are known as *frequency-selective filters*. By definition, an *ideal frequency-selective filter* satisfies the requirements for distortionless response over one or more frequency bands and has zero response at the remaining frequencies. For example, an *ideal bandpass filter* (BPF) is defined by

$$H(e^{j\omega}) = \begin{cases} e^{-j\omega n_d}, & \omega_\ell \le |\omega| \le \omega_u \\ 0, & \text{otherwise} \end{cases} \tag{5.67}$$

where $n_d \ge 0$ and $0 \le \omega_\ell \le \omega_u \le \pi$. Since $H(e^{j\omega})$ is periodic with period 2π radians, we only specify and plot the frequency response over a single period. "Low-frequencies" are located around $\omega = 0$ and "high-frequencies" are close to $\omega = \pi$ radians. The parameters ω_ℓ and ω_u, which specify the end points of the *passband*, are called the lower and upper *cutoff* frequencies. The *bandwidth* of the filter, defined as the width of the passband at the positive part of the frequency axis, is given by

$$\Delta\omega = \omega_u - \omega_\ell. \tag{5.68}$$
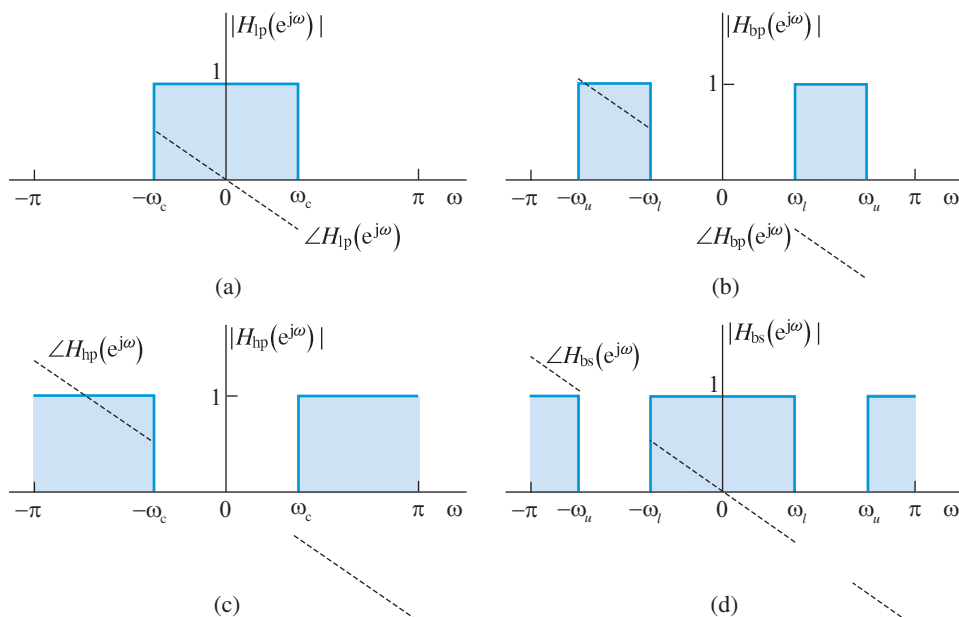
Figure 5.9 Ideal frequency-selective filters: (a) lowpass filter, (b) bandpass filter, (c) highpass filter, and (d) bandstop filter.

An ideal *lowpass* filter is defined by (5.67) with $\omega_\ell = 0$, whereas an ideal *highpass* filter has $\omega_u = \pi$. Ideal *bandstop* filters have a distortionless response over all frequencies except some *stopband*, $\omega_\ell \leq |\omega| \leq \omega_u$, where $H(e^{j\omega}) = 0$. We emphasize that the phase response $\angle H(e^{j\omega})$ is required to be linear only in the passband; there is no need for it to be defined elsewhere because the response of the filter is zero. Figure 5.9 shows the frequency responses of four types of ideal filter.

To understand the implications of the "steep" transition from passband to stopband in ideal filters, we consider an ideal lowpass filter with frequency response

$$H_{\text{lp}}(e^{j\omega}) = \begin{cases} e^{-j\omega n_d}, & |\omega| < \omega_c \\ 0. & \omega_c < |\omega| \leq \pi \end{cases} \tag{5.69}$$

The impulse response corresponding to (5.69) is given by (see Example 4.13)

$$h_{\text{lp}}[n] = \frac{\sin \omega_c (n - n_d)}{\pi (n - n_d)}. \tag{5.70}$$

The impulse response and the step response of the ideal lowpass filter are illustrated in Figure 5.10 for $n_d = 0$. We note that $h_{\text{lp}}[n]$ extends from $-\infty$ to ∞; therefore we *cannot* compute the output of the ideal lowpass filter using a convolution sum. The impulse response $h_{\text{lp}}[n]$ has a DTFT $H_{\text{lp}}(e^{j\omega})$ because it has finite energy. However, it should be noted that $h_{\text{lp}}[n]$ is *not* absolutely summable, that is,

$$\sum_{n=-\infty}^{\infty} |h_{\text{lp}}[n]| = \infty. \tag{5.71}$$
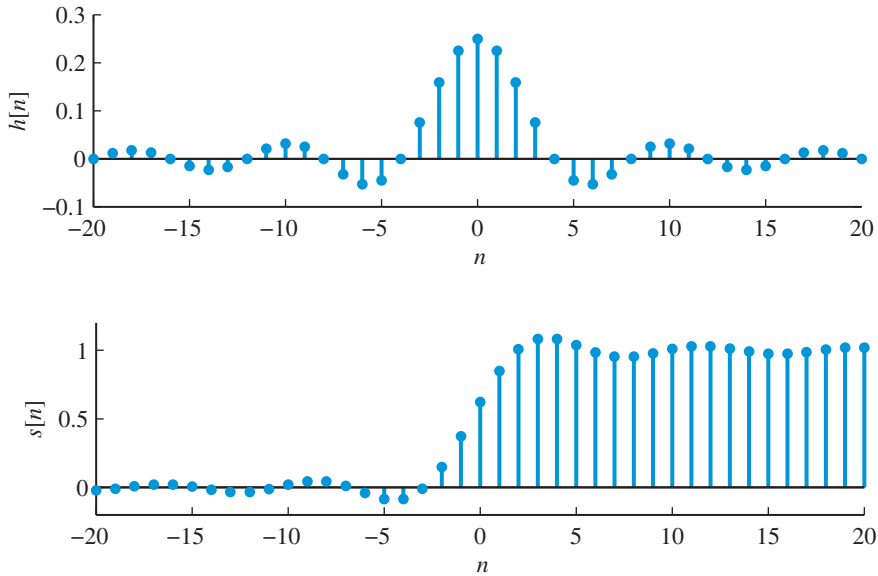
Figure 5.10 Impulse and step response sequences of the ideal lowpass filter.

Therefore, the ideal lowpass filter is unstable. Furthermore, since $r^{-n}h_{\mathrm{lp}}[n]$ is not absolutely summable for any value of r, the sequence $h_{\mathrm{lp}}[n]$ does *not* have a z-transform. Since only systems with a rational system function can be computed recursively, we deduce that we *cannot* compute the output of the ideal lowpass filter either recursively or nonrecursively. In conclusion, *the ideal lowpass filter is unstable and practically unrealizable*.

The impulse response of the ideal bandpass filter can be obtained by modulating the impulse response of an ideal lowpass filter with $\omega_{\mathrm{c}} = (\omega_{\mathrm{u}} - \omega_{\ell})/2 = \Delta\omega/2$ using a carrier with frequency $\omega_0 = (\omega_{\mathrm{u}} + \omega_{\ell})/2$. The result is

$$h_{\mathrm{bp}}[n] = 2\frac{\sin\omega_{\mathrm{c}}(n - n_{\mathrm{d}})}{\pi(n - n_{\mathrm{d}})}\cos\omega_0 n. \tag{5.72}$$

The impulse responses of the ideal highpass and bandstop filters are given by

$$h_{\mathrm{hp}}[n] = \delta[n] - h_{\mathrm{lp}}[n], \tag{5.73}$$

$$h_{\mathrm{bs}}[n] = \delta[n] - h_{\mathrm{bp}}[n], \tag{5.74}$$

because $H_{\mathrm{hp}}(\mathrm{e}^{\mathrm{j}\omega}) = 1 - H_{\mathrm{lp}}(\mathrm{e}^{\mathrm{j}\omega})$ and $H_{\mathrm{bs}}(\mathrm{e}^{\mathrm{j}\omega}) = 1 - H_{\mathrm{bp}}(\mathrm{e}^{\mathrm{j}\omega})$. Therefore, all ideal filters are unstable and unrealizable. Since all ideal filters can be expressed in terms of (5.69), we refer to $H_{\mathrm{lp}}(\mathrm{e}^{\mathrm{j}\omega})$ as the ideal lowpass prototype filter.

Ideal filters are used in the early stages of a design process to specify the modules in a signal processing system. However, since they are not realizable in practice, they must be approximated by *practical* or *nonideal* filters. This is usually done by minimizing some approximation error between the nonideal filter and a prototype ideal filter.
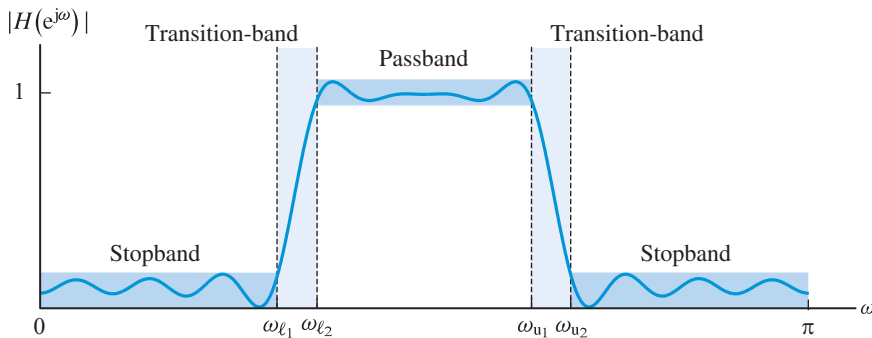
Figure 5.11 Typical characteristics of a practical bandpass filter.

The design of practical filters that approach ideal behavior is the subject of Chapters 10 and 11. To understand the nature of the approximations required to obtain a practical filter from an ideal filter, we note that we can obtain a causal FIR filter by truncating the impulse response of the ideal lowpass filter as follows

$$\hat{h}_{lp}[n] = \begin{cases} \dfrac{\sin \omega_c (n - n_d)}{\pi (n - n_d)}, & 0 \le n \le M - 1 \\ 0. & \text{otherwise} \end{cases} \tag{5.75}$$

As the delay n_d and the length M of $\hat{h}_{lp}[n]$ increase, the resulting filter $\hat{H}_{lp}(e^{j\omega})$ will be a better approximation of the ideal lowpass filter.

A natural question arising at this point is how to evaluate the quality of a practical filter. Figure 5.11 shows the magnitude response of a typical practical bandpass filter. Compared to the ideal bandpass filter in Figure 5.9, we observe a passband where $|H(e^{j\omega})|$ fluctuates about one and stopbands where $|H(e^{j\omega})|$ fluctuates close to zero. Between the passband and stopbands are *transition* bands, where the filter neither passes nor rejects the input frequency components. A good filter should have only a small ripple in the passband, high attenuation in the stopband, and very narrow transition bands. In some applications, the specifications of phase characteristics or time-domain characteristics (for example, the overshoot of the step response) are also important. These issues, which are of fundamental significance in filter design, are further investigated in Chapters 10 and 11.

5.5 Frequency response for rational system functions

In Section 3.6, we demonstrated that all LTI systems of practical interest are described by a difference equation of the form

$$y[n] = -\sum_{k=1}^{N} a_k y[n - k] + \sum_{k=0}^{M} b_k x[n - k], \tag{5.76}$$

and have a rational system function

$$H(z) = \frac{\displaystyle\sum_{k=0}^{M} b_k z^{-k}}{1 + \displaystyle\sum_{k=1}^{N} a_k z^{-k}} = \frac{B(z)}{A(z)}. \tag{5.77}$$

For a stable system, the system function converges on the unit circle. Therefore, from (5.4) and (5.77) we obtain

$$H(e^{j\omega}) = \left.\frac{B(z)}{A(z)}\right|_{z=e^{j\omega}} = \frac{\displaystyle\sum_{k=0}^{M} b_k e^{-j\omega k}}{1 + \displaystyle\sum_{k=1}^{N} a_k e^{-j\omega k}}, \tag{5.78}$$

which expresses $H(e^{j\omega})$ as a ratio of two polynomials in the variable $e^{-j\omega}$.

The frequency response function given in (5.78) can also be expressed in terms of poles and zeros as follows:

$$H(e^{j\omega}) = b_0 \left.\frac{\displaystyle\prod_{k=1}^{M}(1 - z_k z^{-1})}{\displaystyle\prod_{k=1}^{N}(1 - p_k z^{-1})}\right|_{z=e^{j\omega}} = b_0 \frac{\displaystyle\prod_{k=1}^{M}(1 - z_k e^{-j\omega})}{\displaystyle\prod_{k=1}^{N}(1 - p_k e^{-j\omega})}, \tag{5.79}$$

where $\{z_1, z_2, \ldots, z_M\}$ are the zeros and $\{p_1, p_2, \ldots, p_N\}$ are the poles of the system. From (5.79), it follows that the magnitude, phase, and group-delay responses are given by

$$|H(e^{j\omega})| = |b_0| \prod_{k=1}^{M} \left|1 - z_k e^{-j\omega}\right| \Big/ \prod_{k=1}^{N} \left|1 - p_k e^{-j\omega}\right|, \tag{5.80}$$

$$\angle H(e^{j\omega}) = \angle b_0 + \sum_{k=1}^{M} \angle\left(1 - z_k e^{-j\omega}\right) - \sum_{k=1}^{N} \angle\left(1 - p_k e^{-j\omega}\right), \tag{5.81}$$

$$\tau_{\text{gd}}(\omega) = \sum_{k=1}^{M} \frac{d}{d\omega}\left[\angle\left(1 - z_k e^{-j\omega}\right)\right] - \sum_{k=1}^{N} \frac{d}{d\omega}\left[\angle\left(1 - p_k e^{-j\omega}\right)\right], \tag{5.82}$$

where the derivatives in (5.82) are evaluated using the continuous (unwrapped) phase response function. Each of these first-order terms can be expressed in polar notation as $C(\omega) = \left(1 - \alpha e^{j\beta} e^{-j\omega}\right)$. Then, we can easily show that

$$C(\omega) = \left(1 - \alpha e^{j\beta} e^{-j\omega}\right) = 1 - \alpha\cos(\omega - \beta) + j\alpha\sin(\omega - \beta), \qquad (5.83)$$

$$|C(\omega)|^2 = C(\omega)C^*(\omega) = \left(1 - \alpha e^{j\beta} e^{-j\omega}\right)\left(1 - \alpha e^{-j\beta} e^{j\omega}\right)$$

$$= 1 + \alpha^2 - 2\alpha\cos(\omega - \beta), \qquad (5.84)$$

$$\angle C(\omega) = \tan^{-1}\left(\frac{\mathcal{R}e\{C(\omega)\}}{\mathcal{I}m\{C(\omega)\}}\right) = \tan^{-1}\left(\frac{\alpha\sin(\omega - \beta)}{1 - \alpha\cos(\omega - \beta)}\right), \qquad (5.85)$$

$$\tau_{\text{gd}}(\omega) = -\frac{d\Psi(\omega)}{d\omega} = \frac{\alpha^2 - \alpha\cos(\omega - \beta)}{1 + \alpha^2 - 2\alpha\cos(\omega - \beta)}. \qquad (5.86)$$

Expressing the zeros and poles in polar notation as $z_k = q_k e^{j\theta_k}$ and $p_k = r_k e^{j\phi_k}$ and using (5.84)–(5.86), we obtain

$$|H(e^{j\omega})| = |b_0|\left[\frac{\prod\limits_{k=1}^{M}\sqrt{1 + q_k^2 - 2q_k\cos(\omega - \theta_k)}}{\prod\limits_{k=1}^{N}\sqrt{1 + r_k^2 - 2r_k\cos(\omega - \phi_k)}}\right], \qquad (5.87)$$

$$\angle H(e^{j\omega}) = \angle b_0 + \sum_{k=1}^{M}\tan^{-1}\left(\frac{q_k\sin(\omega - \theta_k)}{1 - q_k\cos(\omega - \theta_k)}\right)$$

$$- \sum_{k=1}^{N}\tan^{-1}\left(\frac{r_k\sin(\omega - \phi_k)}{1 - r_k\cos(\omega - \phi_k)}\right), \qquad (5.88)$$

$$\tau_{\text{gd}}(\omega) = \sum_{k=1}^{N}\frac{r_k^2 - r_k\cos(\omega - \phi_k)}{1 + r_k^2 - 2r_k\cos(\omega - \phi_k)} - \sum_{k=1}^{M}\frac{q_k^2 - q_k\cos(\omega - \theta_k)}{1 + q_k^2 - 2q_k\cos(\omega - \theta_k)}. \qquad (5.89)$$

The significance of (5.87)–(5.89) is that they explicitly show the influence of each individual pole or zero on the magnitude, phase, and group-delay responses of the system.

Computation of frequency response MATLAB provides function `freqz` to compute $H(e^{j\omega})$ from the coefficients a_k and b_k over an equally spaced grid in the frequency variable ω. This is done by evaluating the DTFTs of the sequences b_k and a_k at $\omega = 2\pi k/K$, $0 \le k \le K - 1$ using the `fft` function (see Section 8.6):

$$H(e^{j\omega})\big|_{\omega = \frac{2\pi k}{K}} = \frac{\text{DTFT}\{b_k\}\big|_{\omega = \frac{2\pi k}{K}}}{\text{DTFT}\{a_k\}\big|_{\omega = \frac{2\pi k}{K}}} = \text{fft(b,K)./fft(a,K)}. \qquad (5.90)$$

The functions `abs` and `angle` are then used to extract the magnitude and phase responses. We recall that function `angle` computes the principal value of phase. The basic functionality of `freqz` is illustrated by the MATLAB function

5.5 Frequency response for rational system functions

```
function [H, omega]=freqz0(b,a);
% Computation of frequency response function
K=1024;
H=fft(b,K)./fft(a,K); % 0 <= omega < 2*pi
omega=2*pi*(0:K-1)/K;
% H=H(1:K/2+1); % 0 <= omega <= pi
% omega=2*pi*(0:K/2)/K;
% k=[K/2+1:K 1:K/2];
% H=H(k); % -pi < omega <= pi
% omega=2*pi*(-K/2+1:K/2)/K;
```

Figure 5.12 Computation of frequency response function. Function `freqz0` demonstrates the basic algorithm used by MATLAB function `freqz`.

$$[\text{H,omega}]=\text{freqz0(b,a)}, \tag{5.91}$$

shown in Figure 5.12. The lines of code show how we can compute $H(e^{j\omega})$ in the range $0 \le \omega \le \pi$ or in the range $-\pi < \omega \le \pi$. The latter case, which is useful to emphasize symmetries about $\omega = 0$, exploits the periodicity of $H(e^{j\omega})$ to append the first half of the period at the end of the period.

Computation of group delay If we express the frequency response in polar coordinates and take its complex logarithm, we have

$$H(e^{j\omega}) = H_R(\omega) + jH_I(\omega) = G(\omega)e^{j\Psi(\omega)}, \tag{5.92}$$

$$\tilde{H}(\omega) \triangleq \ln H(e^{j\omega}) = \ln G(\omega) + j\Psi(\omega). \tag{5.93}$$

Differentiating both sides of (5.93) yields

$$\tilde{H}'(\omega) = \frac{H'(e^{j\omega})}{H(e^{j\omega})} = \frac{G'(\omega)}{G(\omega)} + j\Psi'(\omega), \tag{5.94}$$

where the prime denotes derivative with respect to ω. Therefore,

$$\tau_{\text{gd}}(\omega) = -\Psi'(\omega) = -\mathcal{I}m\{\tilde{H}'(\omega)\} = -\mathcal{I}m\left\{\frac{H'(e^{j\omega})}{H(e^{j\omega})}\right\}. \tag{5.95}$$

The derivative of $H(e^{j\omega})$ is determined from (4.145) as the DTFT of the sequence $nh[n]$,

$$H_n(e^{j\omega}) = \text{DTFT}\{nh[n]\} = jH'(e^{j\omega}). \tag{5.96}$$

Hence

$$\tau_{\text{gd}}(\omega) = -\mathcal{I}m\left\{\frac{H'(e^{j\omega})}{H(e^{j\omega})}\right\} = \mathcal{I}m\left\{j\frac{H_n(e^{j\omega})}{H(e^{j\omega})}\right\} = \mathcal{R}e\left\{\frac{H_n(e^{j\omega})}{H(e^{j\omega})}\right\}. \tag{5.97}$$

```
function [gd,omega]=grpdelay0(b,a)
% Computation of group delay
K=1024;
h=impz(b,a,K);
n=(0:K-1)';
Hn=fft(n.*h,K);
H=fft(h,K);
ind0=find(abs(H)<10*eps);
gd=real(Hn./H);
gd(ind0)=NaN;
omega=2*pi*(0:K-1)/K;
% gd=gd(1:K/2+1); % 0 <= omega <= pi
% omega=2*pi*(0:K/2)/K;
% k=[K/2+1:K 1:K/2];
% gd=gd(k); % -pi < omega <= pi
% omega=2*pi*(-K/2+1:K/2)/K;
```

Figure 5.13 Computation of group-delay function using (5.96) and (5.97).

This approach is implemented by MATLAB function

$$[gd,omega]=grpdelay0(b,a), \tag{5.98}$$

shown in Figure 5.13. If the system has zeros on the unit circle, $H(e^{j\omega}) = 0$ at the corresponding frequencies. Since input components at these frequencies are "filtered-out" by the system, their phase is indeterminable. For plotting purposes we can set these values to NaN. However, if we wish to use the function $\tau_{gd}(\omega)$, we can replace each NaN with the mean of two adjacent values. MATLAB function grpdelay uses (5.96) and (5.97) for FIR systems and (5.89) for systems with rational system functions. We note that grpdelay0 is sufficient for most practical purposes.

Figure 5.15 shows the pole-zero plot, the magnitude response, the phase response (principal value and continuous function), and the group delay of the system

$$H(z) = \frac{1 + 1.655z^{-1} + 1.655z^{-2} + z^{-3}}{1 - 1.57z^{-1} + 1.264z^{-2} - 0.4z^{-3}}, \tag{5.99}$$

evaluated using the functions freqz, angle, and grpdelay. The continuous or unwrapped phase $\Psi(\omega)$ is evaluated from the group delay using simple trapezoidal integration. This approach is implemented with the MATLAB function contphase, which is shown in Figure 5.14. To understand the phase response discontinuities in Figure 5.15, we recall that the principal phase value jumps by a multiple of 2π when $|\Psi(\omega)| > \pi$. This explains the 2π jumps at the first and last discontinuities. The remaining three discontinuities of size π result from sign reversals due to the real zero at $\omega = \pi$ and the complex conjugate zeros at $\omega = \pm 3\pi/5$.

```
function cph=contphase(grd,om)
% Computation of continuous phase function
% from equidistant values of group delay
N=length(om);
dom=om(2)-om(1);
p(1)=0;
for k=2:N
    p(k)=p(k-1)+dom*(grd(k-1)+grd(k))/2;
end
cph=-p;
```

Figure 5.14 Computation of continuous phase function by integrating the group delay.
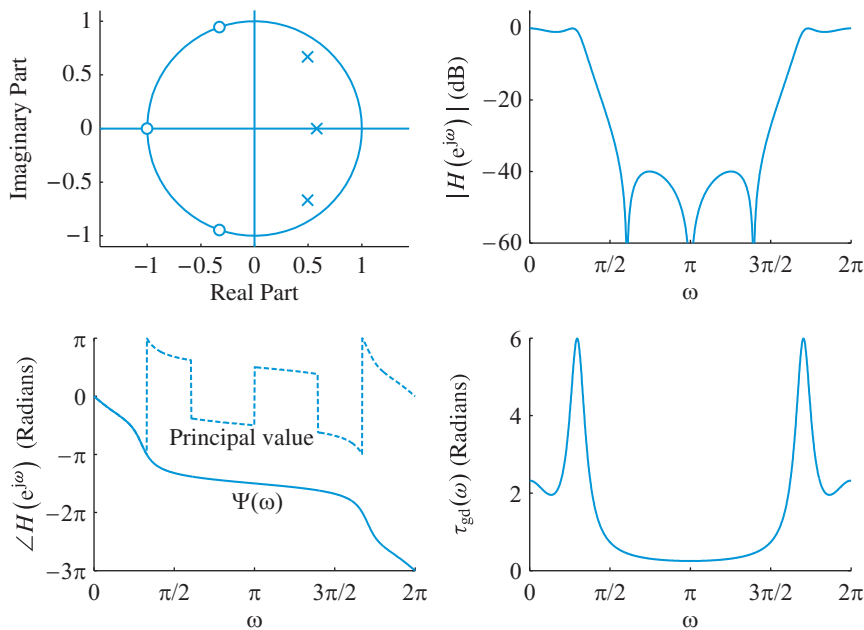
Figure 5.15 Pole-zero plot, magnitude response, phase response (principal value and continuous phase functions), and group delay of the system (5.99).

Interactive filter visualization tool The MATLAB filter visualization tool function `fvtool(b,a)` provides a convenient utility to evaluate and display the magnitude, phase, phase delay, group delay, impulse response, step response, pole-zero pattern, and coefficients of any system with a rational system function. The functionality of this tool is illustrated in Figure 5.16 using the system function (5.99). This utility uses the functions `phasez` and `phasedelay` to determine the phase response (in rads) and the phase delay (in samples) of the system.
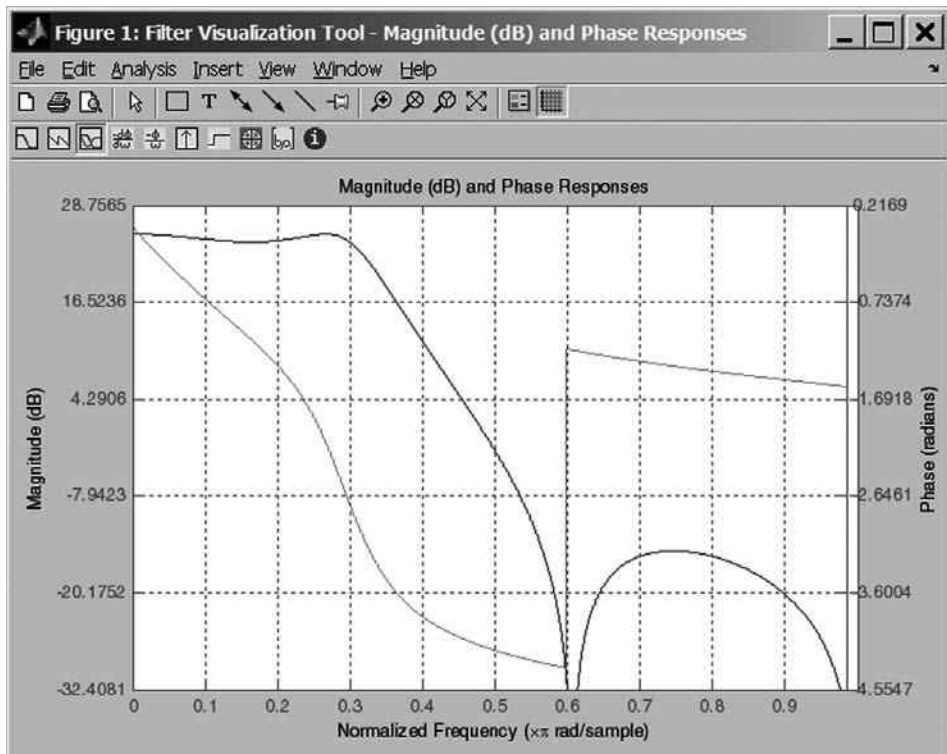
Figure 5.16 MATLAB filter visualization tool function `fvtool`.

Practical recommendations In practice, we usually need to compute and plot the magnitude, phase, and group-delay responses of a filter. If we have the Signal Processing Toolbox, we can call the functions `freqz` and `grpdelay` as follows:

```
% Typical use of freqz and grpdelay functions
om=linspace(-pi,pi,1000);
b=1; a=[1 -0.8];
H=freqz(b,a,om);
% grp delay is measured in samples
tau=grpdelay(b,a,om);
subplot(3,1,1), plot(om/pi,abs(H));
% angles are measured in units of pi rads
subplot(3,1,2), plot(om/pi,angle(H)/pi);
subplot(3,1,3), plot(om/pi,tau);
```

Although these functions can be called in many different ways, the suggested approach is easy to remember and can be used to compute the frequency response at any set of frequencies. The functions `freqz0` and `grpdelay0` can be easily modified to compute the frequency response and group delay at different frequency ranges.

5.6   Dependence of frequency response on poles and zeros

The shape of the frequency response is determined by the impulse response or the coefficients of the difference equation. However, we *cannot* guess the shape of $|H(e^{j\omega})|$ and $\angle H(e^{j\omega})|$ by inspecting the values of $h[n]$ or $\{a_k, b_k\}$. In this section, we show that there is a strong dependence of the shape of the frequency response on the location of poles and zeros of the system. We can use this dependence to (a) obtain a simple and intuitive procedure for determining quickly the magnitude and phase response, and (b) to gain physical insight into the filtering characteristics of LTI systems.

5.6.1   Geometrical evaluation of $H(e^{j\omega})$ from poles and zeros

We start by noting that (5.79) can be equivalently written as

$$H(e^{j\omega}) = b_0 \left[\frac{\prod_{k=1}^{M}\left(1 - z_k e^{-j\omega}\right)}{\prod_{k=1}^{N}\left(1 - p_k e^{-j\omega}\right)} \right] = b_0 e^{j\omega(N-M)} \left[\frac{\prod_{k=1}^{M}\left(e^{j\omega} - z_k\right)}{\prod_{k=1}^{N}\left(e^{j\omega} - p_k\right)} \right]. \tag{5.100}$$

This equation consists of factors of the form $\left(e^{j\omega} - z_k\right)$ and $\left(e^{j\omega} - p_k\right)$. The factor $\left(e^{j\omega} - z_k\right)$ is a complex number represented by a vector $\overrightarrow{Z_k Z}$ drawn from the point z_k (zero) to the point $z = e^{j\omega}$ in the complex plane, as illustrated in Figure 5.17. This complex number can be written in polar form as follows:
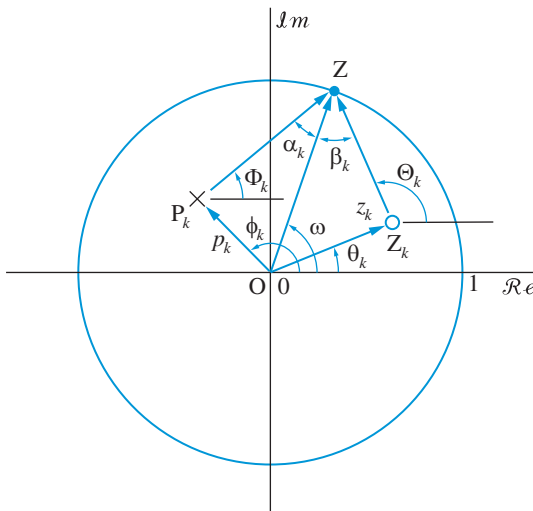
Figure 5.17 The quantities required to compute the magnitude and phase response of a system from the location of its poles and zeros.

$$\left(e^{j\omega} - z_k\right) = \overrightarrow{Z_k Z} = Q_k e^{j\Theta_k},\tag{5.101}$$

where Q_k is the distance from the zero z_k to the point $e^{j\omega}$, and Θ_k is the angle of the vector $\overrightarrow{Z_k Z}$ with the (horizontal) positive real axis. Similarly, see Figure 5.17, the factor $\left(e^{j\omega} - p_k\right)$ is a complex number that can be expressed as

$$\left(e^{j\omega} - p_k\right) = \overrightarrow{P_k Z} = R_k e^{j\Phi_k},\tag{5.102}$$

where R_k is the distance from the pole p_k to the point $e^{j\omega}$ and Φ_k the angle of $\overrightarrow{P_k Z}$ with the positive real axis. Substituting (5.101) and (5.102) into (5.100) yields

$$H(e^{j\omega}) = |b_0| \frac{\prod_{k=1}^{M} Q_k(\omega)}{\prod_{k=1}^{N} R_k(\omega)}$$

$$\times \exp\left[\angle b_0 + \omega(N - M) + \sum_{k=1}^{M} \Theta_k(\omega) - \sum_{k=1}^{N} \Phi_k(\omega)\right].\tag{5.103}$$

We note that $\angle b_0 = \pi$ rads when $b_0 < 0$, because $e^{j\pi} = -1$, and $\angle b_0 = 0$ for $b_0 \geq 0$, because $e^{j0} = 1$. We have expressed $Q, R, \Theta,$ and Φ as functions of ω, to emphasize their dependence on frequency.

The magnitude and phase responses are easily obtained from (5.103) as

$$|H(e^{j\omega})| = |b_0| \frac{\prod_{k=1}^{M} Q_k(\omega)}{\prod_{k=1}^{N} R_k(\omega)},$$

$$\angle H(e^{j\omega}) = \angle b_0 + \omega(N - M) + \sum_{k=1}^{M} \Theta_k(\omega) - \sum_{k=1}^{N} \Phi_k(\omega),\tag{5.104}$$

where ω is the angle of the point $z = e^{j\omega}$ with the positive real axis and

$$Q_k(\omega) = \text{distance of } k\text{th zero from } z = e^{j\omega},$$

$$R_k(\omega) = \text{distance of } k\text{th pole from } z = e^{j\omega},$$

$$\Theta_k(\omega) = \text{angle of } k\text{th zero with the real axis,}$$

$$\Phi_k(\omega) = \text{angle of } k\text{th pole with the real axis.}$$

Therefore, *the magnitude response at a certain frequency ω is given by the product of the lengths of the vectors drawn from the zeros to $z = e^{j\omega}$ divided by the product of the lengths of vectors drawn from the poles to $z = e^{j\omega}$.* Similarly, the phase response is obtained by subtracting from the sum of angles of zeros the sum of angles of poles. All angles are determined with respect to the positive real axis. Using this geometrical procedure, we

can determine $H(e^{j\omega})$ for every value of ω or equivalently any location of the point $e^{j\omega}$ on the unit circle.

5.6.2   Significance of poles and zeros

To understand the effect of poles and zeros on the magnitude and phase responses, we separately consider the case of a single pole and a single zero.

Gain enhancement by a pole   Consider a pole $p_k = r_k e^{j\phi_k}$, as illustrated in Figure 5.18. To find the magnitude response $|H(e^{j\omega})|$ for a certain value of ω, we connect the pole (point P_k) to the tip of vector $z = e^{j\omega}$ (point Z on the unit circle). If the length of this line is $R_k(\omega)$, then

$$|H(e^{j\omega})| = \frac{\kappa}{(\overline{P_k Z})} = \frac{\kappa}{R_k(\omega)}, \tag{5.105}$$

where overbar denotes the length of a vector. The exact value of constant κ is not important at this point. The line segment $P_k Z$ takes its minimum value $1 - r_k$ at $\omega = \phi_k$, and its maximum value $1 + r_k$ at $\omega = \phi_k + \pi$. Therefore, the length $\overline{P_k Z}$ increases progressively as ω increases from ϕ_k to $\phi_k + \pi$ and then decreases continuously until ω approaches the value ϕ_k. Then, according to (5.105), $|H(e^{j\omega})|$ decreases as ω goes from ϕ_k to $\phi_k + \pi$ and then progressively increases as ω moves closer to ϕ_k (see Figure 5.18). We conclude that a pole $p_k = r_k e^{j\phi_k}$ results in a frequency-selective response that enhances the gain around $\omega = \phi_k$ (angle of the pole) and attenuates the gain as we move away from ϕ_k. The dynamic range of the magnitude response

$$\frac{|H(e^{j\omega})|_{\max}}{|H(e^{j\omega})|_{\min}} = \frac{1 + r_k}{1 - r_k} \tag{5.106}$$

(a)   (b)
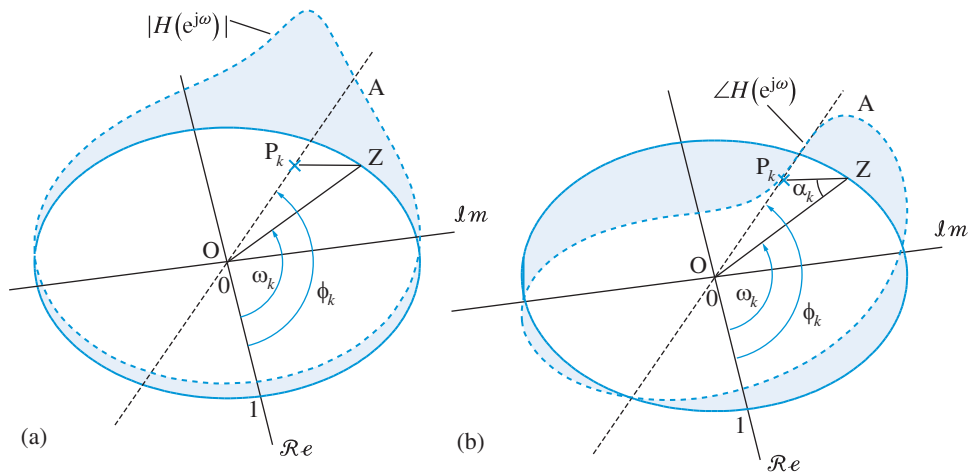
Figure 5.18 Geometrical computation of magnitude (a), and phase (b), responses for the case of a single pole.

increases as the pole is moved closer to the unit circle. As a result, the peak of $H(e^{j\omega})$ at $\omega = \phi_k$ becomes sharper as the pole approaches the unit circle. The maximum gain $|H(\phi_k)|$ goes to infinity as the pole moves on the unit circle. However, this should be avoided, because causal LTI systems with poles on or outside the unit circle are unstable.

To evaluate the phase response for a single pole, we first recall from (5.104) that $\angle H(e^{j\omega}) = \omega - \Phi_k(\omega)$. Since the angles of the triangle OP_kZ sum to π, we have $\alpha_k + (\phi_k - \omega) + (\Phi_k + \pi - \phi_k) = \pi$ or $\omega - \Phi_k = \alpha_k$. Hence

$$\angle H(e^{j\omega}) = \omega - \Phi_k(\omega) = \alpha_k. \qquad (5.107)$$

Moving the point Z around the unit circle, we see that $\angle H(e^{j\omega})$ becomes zero and changes signs at $\omega = \phi_k$ and $\omega = \pi + \phi_k$ (see Figure 5.18). However, it is not easy to obtain a reasonably accurate shape for the phase response with this approach. We note that these sign changes and the use of principal value are the cause of the discontinuities observed in numerical evaluation of phase response.

The geometrical interpretation provides a nice way to illustrate the symmetry properties of magnitude and phase responses. Careful inspection of Figure 5.18 shows that, in general, we have

$$|H(e^{j\omega})| = \frac{\kappa}{(P_1Z_1)} \neq \frac{\kappa}{(P_1Z_2)} = \left|H(e^{-j\omega})\right|, \qquad (5.108)$$

$$\angle H(e^{j\omega}) = \alpha_1 \neq \alpha_2 = -\angle H(e^{-j\omega}). \qquad (5.109)$$

However, if the pole P_k moves on the real axis we have $\overline{P_1Z_1} = \overline{P_1Z_2}$ and $\alpha_1 = \alpha_2$. Therefore, $|H(e^{j\omega})| = |H(e^{-j\omega})|$ (even) and $\angle H(e^{j\omega}) = -\angle H(e^{-j\omega})$ (odd), as expected for systems with real coefficients (see Table 4.3). Another way to enforce this symmetry is to place a complex conjugate pole at $p_k^* = r_k e^{-j\phi_k}$, as shown in Figure 5.19. In this case, due to symmetry of poles about the real axis, we always have

$$|H(e^{j\omega})| = \frac{\kappa}{(P_1Z)(P_2Z)} = \frac{\kappa}{(P_1\bar{Z})(P_2\bar{Z})} = \left|H(e^{-j\omega})\right|, \qquad (5.110)$$

where Z and $\bar{Z}$ are points on the unit circle at frequencies ω and $-\omega$.

Gain suppression by a zero Suppose now that the pole in Figure 5.18 is replaced by a zero $z_k = q_k e^{j\theta_k}$. The magnitude response at a given frequency ω is given by

$$|H(e^{j\omega})| = \kappa\left(\overline{Z_kZ}\right) = \kappa Q_k(\omega), \qquad (5.111)$$

that is, it is proportional to the distance of the zero from the point on the unit circle corresponding to $z = e^{j\omega}$. We can easily see that $|H(e^{j\omega})|$ dips sharply at $\omega = \theta_k$ and increases as point Z moves away from Z_k. The size and sharpness of the dip increase as the zero approaches the unit circle. The minimum gain is $\left|H(e^{j\theta_k})\right| = 0$ when the zero falls on
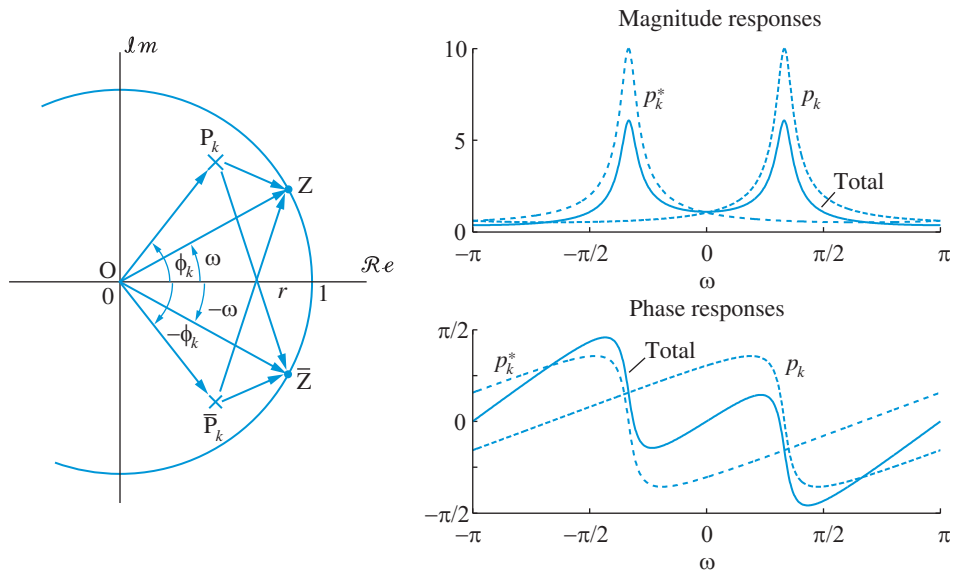
Figure 5.19 Geometrical explanation of the shape of magnitude and phase responses generated by single and complex conjugate poles at $p_k = 0.9e^{\pm j\pi/3}$. Only pole-zero patterns with mirror symmetry about the real axis, that is, real or complex-conjugate poles and zeros, result in magnitude responses with even symmetry and phase responses with odd symmetry.

the unit circle; hence, the system fully suppresses sinusoidal components with frequency $\omega = \theta_k$. Thus, we conclude that zeros have the opposite effect of poles. From (5.104) and the triangle OZZ_k in Figure 5.17, we can show that the phase response for a single zero is

$$\angle H(e^{j\omega}) = -\omega + \Theta_k(\omega) = \beta_k, \tag{5.112}$$

and changes sign at $\omega = \theta_k$ and $\omega = \theta_k + \pi$. Adding a complex-conjugate zero, to assure a system with real coefficients, makes the magnitude response function symmetric about $\omega = 0$. This can be proven using arguments similar to those used for complex conjugate poles (see Problem 32).

Zeros outside the unit circle Although the poles of causal and stable systems should be inside the unit circle, their zeros can be anywhere in the z-plane. Moving a zero outside the unit circle, without changing its angle, has an interesting effect on its phase response. This effect is illustrated using the geometrical construction in Figure 5.20. The phase response of the zero inside the unit circle is equal to $\alpha_{in}(\omega) = \Theta_{in}(\omega) - \omega$ and continuously changes sign at $\omega = 0$. In contrast, for a zero outside the unit circle, the phase response changes from $-\pi$ at $\omega = 0 - \epsilon$ to π at $\omega = 0 + \epsilon$, where ϵ is an arbitrarily small positive number. Furthermore, since $\Theta_{out}(\omega) \geq \Theta_{in}(\omega)$ we have $\alpha_{out}(\omega) \geq \alpha_{in}(\omega)$. Thus, zeros outside the unit circle introduce larger phase shifts than zeros inside the unit circle. This topic is further discussed in Section 5.10. For a zero, $z = re^{j\theta}$, not on the real-line we can simply
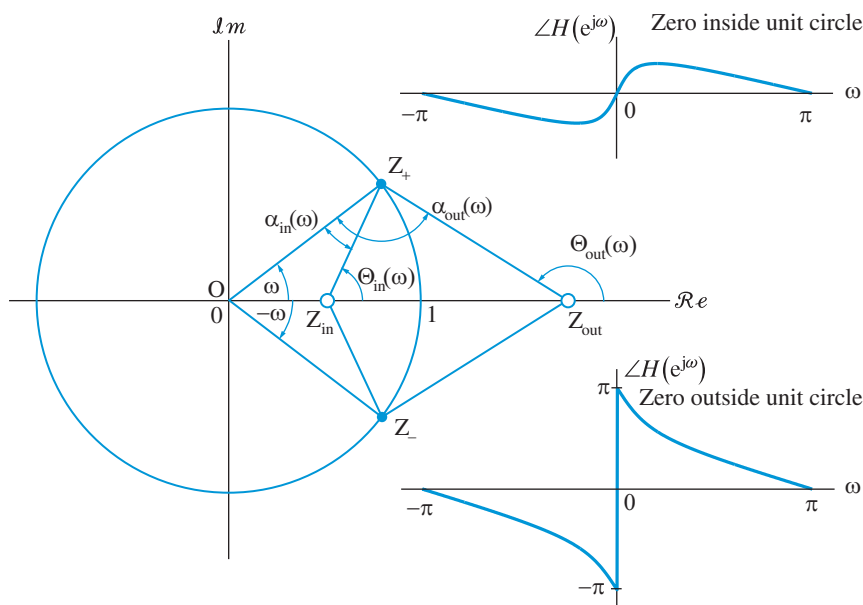
Figure 5.20 Geometrical evaluation of phase response of a zero inside the unit circle and a zero outside the unit circle.

rotate (see Figure 5.18) the curves in Figure 5.20 by θ and add a constant to obtain the new phase responses.

Visual interpretation The terms poles and zeros and their effect on the magnitude response make sense when we plot $|H(z)|$ as a function of z. The result is a surface, whose distance above the z-plane is equal to the magnitude of $H(z)$. The zeros are the points where the surface dips down to touch the z-plane. The poles are points where the surface looks like a flexible rubber membrane pushed underneath by a thin rod (or a pole). The resulting peaks become sharper, thinner, and higher as we approach the pole. At the precise location of the pole, $H(z) = \infty$; however, we do not plot this point to retain a finite scale in the graph. These interpretations are illustrated in Figure 5.21.

Time-domain, frequency-domain, and z-domain We shall now review and put into perspective the three domains used for the analysis of LTI systems and the key tools pertinent to each one of them. Discrete-time signals and systems "exist" in the time-domain, that is, they are functions of the time index n. Any realizable LTI system is defined by the difference equation

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k], \tag{5.113}$$
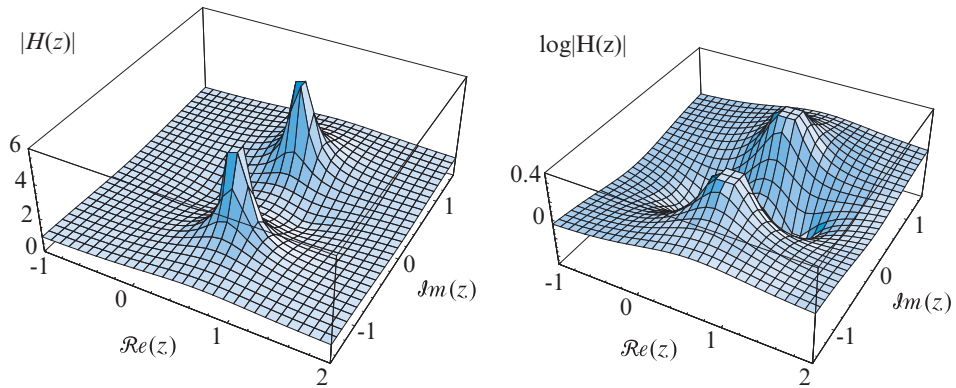
Figure 5.21 The meaning of poles and zeros as peaks and valleys of the z-transform magnitude surface. There are two zeros at $z_1 = 0$, $z_2 = 1$ and two poles at $p_{1,2} = 0.9\mathrm{e}^{\pm\mathrm{j}\pi/4}$. The logarithmic plot shows better the effect of zeros on the response surface.

which provides the basic algorithm for the computation of the output sequence $y[n]$ from the input sequence $x[n]$. In the z-domain, the system is described by the system function, which is obtained from the coefficients of the difference equations,

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = b_0 \frac{\prod_{k=1}^{M}\left(1 - z_k z^{-1}\right)}{\prod_{k=1}^{N}\left(1 - p_k z^{-1}\right)}. \tag{5.114}$$

Finally, if the system is stable, evaluation of $H(z)$ on the unit circle yields the frequency response function $H(\mathrm{e}^{\mathrm{j}\omega}) = H(z)|_{z=\mathrm{e}^{\mathrm{j}\omega}}$. The shape of the impulse response $h[n]$ and the magnitude response $|H(\mathrm{e}^{\mathrm{j}\omega})|$ (that is, the shape of the passbands and stopbands) are strongly dependent on the location of poles and zeros with respect to the unit circle (see Section 3.7 and Figures 5.18, 5.19). The three domains, n, ω, and z, and their relationships are illustrated in Figure 5.22. We note that all three representations are completely specified by the system coefficients $\{a_k, b_k\}$. Therefore, they represent information in different but complementary ways and provide additional insight into the effects of the system on the input signals.

5.7   Design of simple filters by pole-zero placement

From the previous discussion we conclude that there is an intuitive strong dependence between the locations of poles and zeros and the frequency response of a system. This relationship can be used to design simple filters with desired frequency response
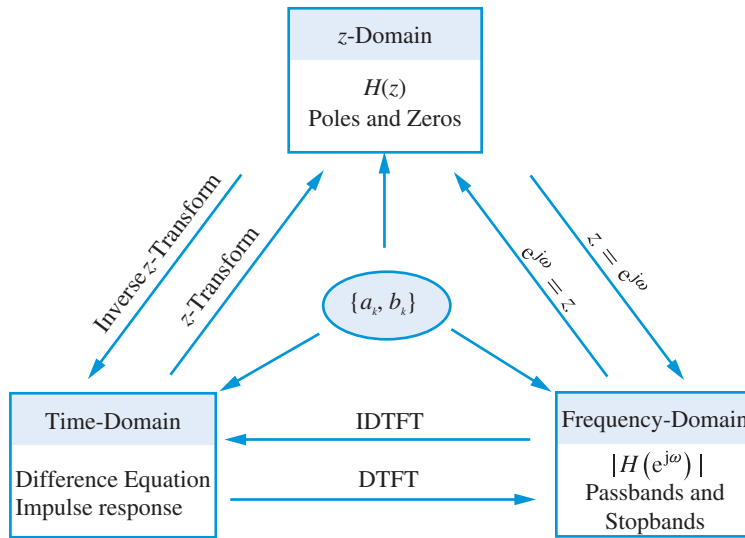
Figure 5.22 Relationship between the time, frequency, and z-transform domains. The frequency-domain description exists for stable systems only.

characteristics by interactively placing poles and zeros on the z-plane. This procedure is based on the following guidelines:

- To suppress a frequency component at $\omega = \omega_0$, we should place a zero at angle $\theta = \omega_0$ on the unit circle. Indeed, $\overline{Z_k Z} = 0$ implies that $H(e^{j\omega_0}) = 0$.
- To enhance or amplify a frequency component at $\omega = \omega_0$, we should place a pole at angle $\phi = \omega_0$ close but inside the unit circle. Indeed, as $\overline{P_k Z}$ becomes very small, the magnitude response $\left| H(e^{j\omega_0}) \right| \propto 1/\overline{P_k Z}$ becomes very large.
- Complex poles or zeros should appear in complex conjugate pairs to assure that the system has real coefficients. This stems from the fact that the frequency ω is defined as the angle with respect to the positive real axis. Therefore, all symmetries should be defined with respect to the real axis.
- Poles or zeros at the origin do not influence the magnitude response because their distance from any point on the unit circle is unity. However, a pole (or zero) at the origin adds (or subtracts) a linear phase of ω rads to the phase response. We often introduce poles and zeros at $z = 0$ to assure that $N = M$.

In this section we use these guidelines to design some simple filters that are useful in practical applications. Furthermore, these examples help to appreciate how the location of poles and zeros affects the magnitude response of a filter.

5.7.1   Discrete-time resonators

Consider a system with two complex conjugate poles at $p_{1,2} = re^{\pm j\phi}$:

$$H(z) = \frac{b_0}{\left(1 - re^{j\phi}z^{-1}\right)\left(1 - re^{-j\phi}z^{-1}\right)} = \frac{b_0}{1 - (2r\cos\phi)z^{-1} + r^2 z^{-2}}, \qquad (5.115)$$

where $0 < r < 1$ and $0 \le \phi \le \pi$. The input-output relationship is given by

$$y[n] = (2r\cos\phi)y[n-1] - r^2y[n-2] + b_0x[n]. \tag{5.116}$$

Using partial fraction expansion, we can show that the impulse response is

$$h[n] = b_0r^n\frac{\sin[(n+1)\phi]}{\sin\phi}u[n], \tag{5.117}$$

which is a damped sinusoid of frequency ϕ (see Figure 3.11).

Figure 5.23 shows the pole-zero pattern, magnitude response, phase response, and group delay for $p_{1,2} = 0.9e^{\pm j\pi/3}$. These quantities can be determined from (5.87), (5.88), and (5.89) for $M = 0$, $N = 2$, $r_1 = r_2 = r$, and $\phi_1 = -\phi_2 = \phi$. The shape of $|H(e^{j\omega})|$ can be easily justified geometrically from the pole-zero pattern. The peak of the magnitude response can be shown to be located at a frequency ω_c, given by

$$\cos\omega_c = [(1+r^2)/(2r)]\cos\phi. \tag{5.118}$$

Since $1 + r^2 > 2r$ for $r < 1$, and we have $\cos\omega_c > \cos\phi$, the peak is lower than the pole frequency for $0 < \phi < \pi/2$ and higher than the pole frequency for $\pi/2 < \phi < \pi$. As expected, for poles close to the unit circle the peak occurs at $\omega_c \approx \phi$. From (5.115) we
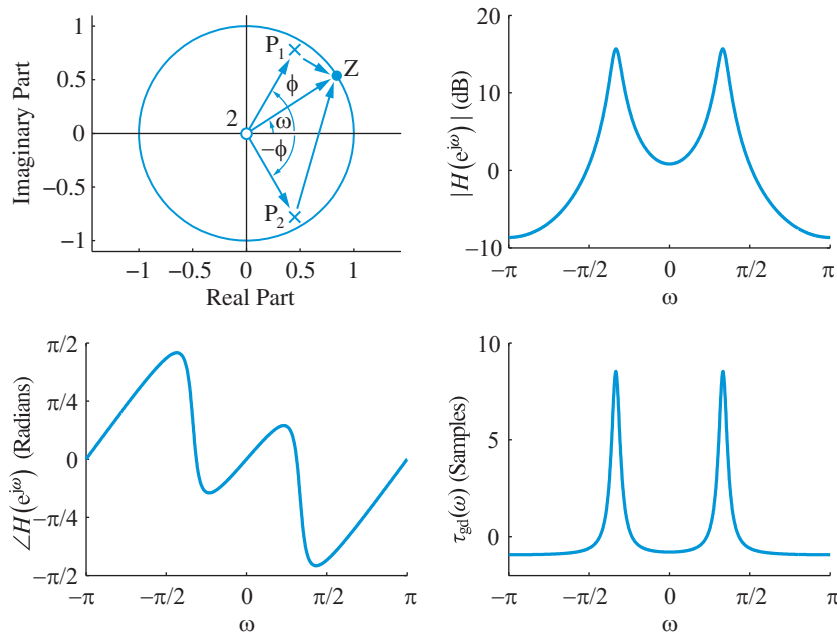
Figure 5.23 Typical pole-zero pattern, magnitude response, phase response, and group delay for a discrete-time resonator with $r = 0.9$ and $\phi = \pi/3$ radians.

obtain $|H(e^{j\phi})| = b_0/(1-r)\sqrt{1+r^2-2r\cos 2\phi}$. Thus, we can normalize the gain of the filter by choosing

$$b_0 = (1-r)\sqrt{1+r^2-2r\cos 2\phi}. \tag{5.119}$$

The width of the peak is determined by the 3-dB bandwidth, which is calculated by determining the two nearest frequencies ω_1 and ω_2 around ω_c where $|H(e^{j\omega})|$ is equal to $(1/\sqrt{2})|H(e^{j\omega_c})|$. The 3-dB bandwidth can be approximated by

$$\Delta\omega \approx 2(1-r), \quad r \lesssim 1 \tag{5.120}$$

which shows that the peak becomes sharper as the poles approach the unit circle.

The system (5.116) is known as a discrete-time *resonator* because it has a large magnitude response, that is, it "resonates" in the vicinity of the pole locations. Since a resonator is essentially a bandpass filter, its bandpass characteristics can be improved by attenuating the low and high frequencies by placing a zero at $z = 1$ and a zero at $z = -1$. The result is a resonator with system function

$$H(z) = b_0 \frac{1 - z^{-2}}{1 - (2r\cos\phi)z^{-1} + r^2 z^{-2}}. \tag{5.121}$$

The frequency response characteristics of this system are discussed in Tutorial Problem 14.

Discrete-time sinusoidal oscillators If the poles of the resonator (5.116) are placed on the unit circle, that is, if we set $r = 1$, and b_0 is set to $A\sin\phi$, we obtain

$$h[n] = A\sin[(n+1)\phi]u[n]. \tag{5.122}$$

Therefore, we can generate a sinusoidal signal using the difference equation

$$y[n] = (2\cos\phi)y[n-1] - y[n-2] + b_0\delta[n], \quad y[-1] = y[-2] = 0. \tag{5.123}$$

The frequency of oscillation ϕ is determined by the angle of the poles. This system, which is a limiting form of a resonator, is known as a *sinusoidal oscillator*. The system is marginally stable because the poles are on the unit circle.

5.7.2   Notch filters

An FIR system with two complex conjugate zeros is defined by

$$H(z) = b_0[1 - (2r\cos\phi)z^{-1} + r^2 z^{-2}]. \tag{5.124}$$

which is the reciprocal of (5.115), except for the b_0 factor. Thus, the frequency response plots for this FIR system are horizontally-flipped from those in Figure 5.23 (see Tutorial Problem 15). The magnitude response of this filter contains two sharp dips at $\omega = \pm\phi$.
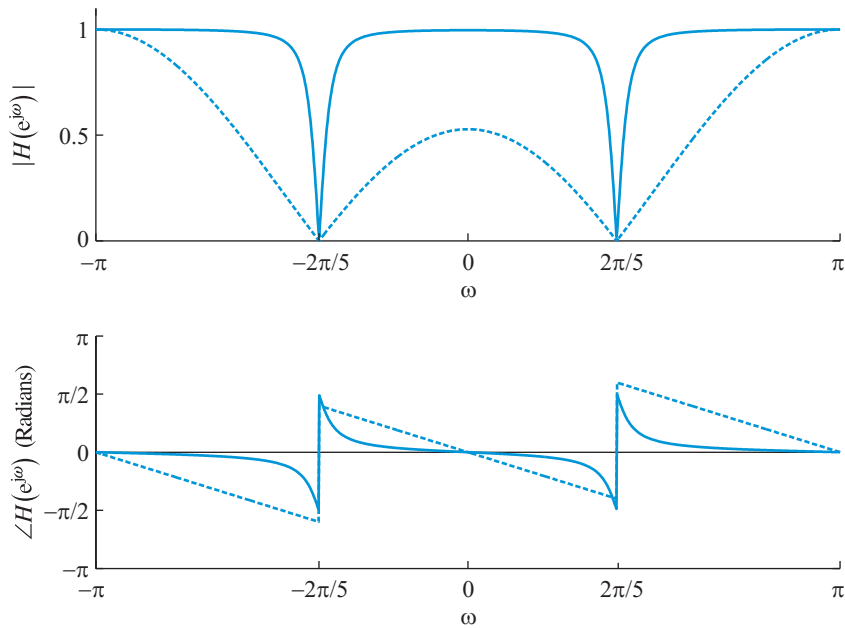
Figure 5.24 Magnitude and phase response of a second-order FIR notch filter (dashed line) and a second-order IIR notch filter with $r = 0.9$ and $\phi = 2\pi/5$.

If we cascade several second-order sections, like (5.124), we can create FIR filters with better stopbands (see Problem 39).

If the zeros of (5.124) are placed on the unit circle by setting $r = 1$, the input frequency components at $\omega = \pm\phi$ are eliminated. Filters that have perfect nulls at certain frequencies are known as *notch* filters. The second-order FIR notch filter has system function

$$H(z) = b_0[1 - (2\cos\phi)z^{-1} + z^{-2}]. \tag{5.125}$$

The problem with FIR notch filters is that the bandwidth of the notches is large. A simple way to create sharper notches is to place the zeros on the unit circle and two complex conjugate poles at the same angle with the zeros, close to the zeros but inside the unit circle. The system function for the resulting notch filter is

$$G(z) = b_0 \frac{1 - (2\cos\phi)z^{-1} + z^{-2}}{1 - (2r\cos\phi)z^{-1} + r^2z^{-2}}. \tag{5.126}$$

The creation of sharper notches is illustrated in Figure 5.24, which shows the magnitude responses of (5.125) and (5.126) for $r = 0.9$ and $\phi = 0.4\pi$ radians. The gain b_0 is chosen so that the maximum magnitude response is equal to one (see Problem 64).

5.7.3   Comb filters

The resonator has a single passband and the notch filter has a single stopband in the interval $0 \le \omega \le \pi$. However, there are applications that require simple filters with multiple passbands and stopbands, known as *comb* filters. To design a comb filter, we start with a filter $H(z)$ having a single passband or stopband in the interval $-\pi < \omega \le \pi$. The frequency response $H(e^{j\omega})$ is periodic with period 2π radians. If we replace z^{-1} by z^{-L} in $H(z)$, that is, if we replace each unit delay by an L-unit delay, we obtain a new system defined by

$$G(z) \triangleq H\left(z^L\right) = \sum_{n=-\infty}^{\infty} h[n]z^{-nL}, \tag{5.127}$$

where L is an integer. The frequency response is given by

$$G(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]e^{-j\omega Ln} = H(e^{j\omega L}). \tag{5.128}$$

Therefore $G(e^{j\omega})$ is periodic with period $2\pi/L$ radians; each new period is a compressed version of the old period. The impulse response $g[n]$ is obtained by inserting $(L-1)$ zeros between successive samples of $h[n]$. Indeed, careful inspection of (5.127) yields the formula

$$g[n] = \begin{cases} h[n/L], & n = 0, \pm L, \pm 2L, \ldots \\ 0. & \text{otherwise} \end{cases} \tag{5.129}$$

To illustrate these ideas consider the first-difference filter

$$H(z) = 1 - z^{-1}, \tag{5.130}$$

which yields the following comb filter

$$G(z) = 1 - z^{-L}. \tag{5.131}$$

Figure 5.25 shows $|H(e^{j\omega})|$ and $\left|G(e^{j\omega})\right|$ for $L = 8$. We note that $G(e^{j\omega})$ is periodic with period $2\pi/8$ radians, as expected. The name comb filter stems from the comblike shape of the resulting magnitude response.

Example 5.7   Comb reverberator unit
In Example 2.8 we discussed a recursive IIR reverberation filter with the following input–output equations:

$$y[n] = ay[n-D] + x[n]. \quad -1 < a < 1 \tag{5.132}$$
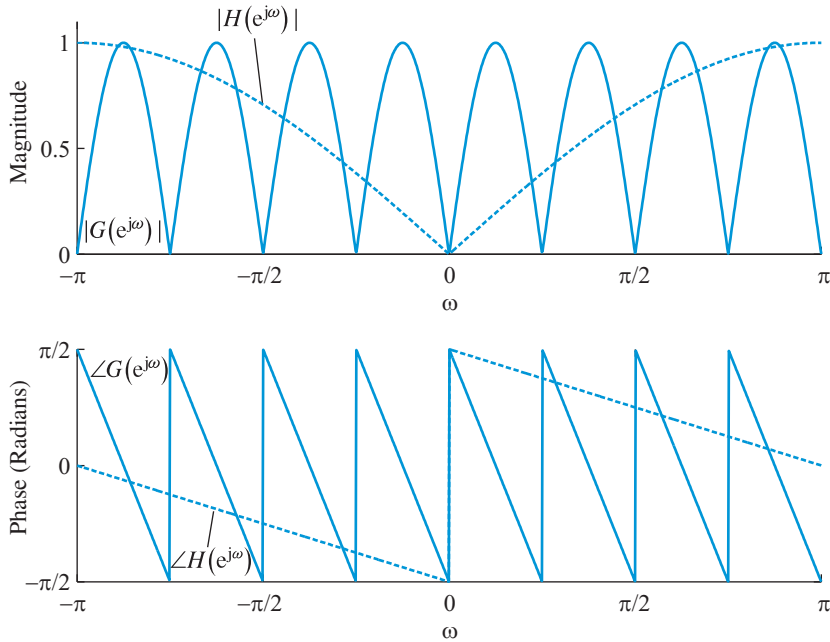
Figure 5.25 Magnitude and phase response of a first-order difference filter $H(e^{j\omega})$ and the corresponding comb filter $G(e^{j\omega})$ for $L = 8$. Note that $G(e^{j\omega})$ is periodic with period $2\pi/8$ radians and that both filters have a linear-phase response.

The impulse response and the system function are given, respectively, by

$$h[n] = \sum_{k=0}^{\infty} a^k \delta[n - kD] \overset{\mathcal{Z}}{\longleftrightarrow} H(z) = \frac{1}{1 - az^{-D}}. \qquad (5.133)$$

We see that the reverberator $H(z)$ is a comb filter derived from the prototype filter $H_p(z) = 1/(1-az^{-1})$. Figure 5.26 shows the pole-zero pattern, the impulse response, the magnitude response, and the group delay for $a = 0.7$ and $D = 5$. Since the poles are located at $p_k = |a|^{1/D} e^{j2\pi k/D}$, the magnitude response exhibits D peaks at $\omega = 2\pi k/D$ and D dips at $\omega = (2k + 1)\pi/D$. For $a > 0$ the magnitude response has a maximum value $H_{max} = 1/(1-a)$ and a minimum value $H_{min} = 1/(1+a)$; therefore, $H_{max}/H_{min} = (1+a)/(1-a)$, which for $a = 0.7$ results in 15 dB attenuation. The coloring of the reverberated speech by these periodic resonances creates a subjective "hollow" effect (Schroeder and Logan 1961). ■

5.7.4   Pole-zero pattern rotation – frequency transformations

The M-point *moving average* filter, which is an FIR system defined by

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n - k], \qquad (5.134)$$
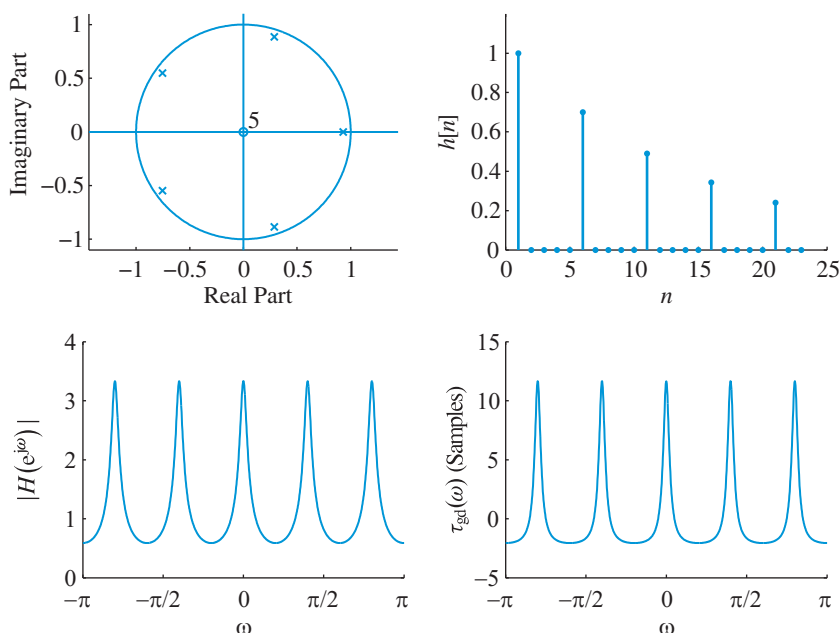
Figure 5.26 Pole-zero pattern, impulse response, magnitude response, and group delay of a comb reverberator with $a = 0.7$ and $D = 5$. Magnitude response of the reverberator filter resembles a comb.

is widely used to "smooth" discrete-time signals. The system function is

$$H(z) = \frac{1}{M} \sum_{k=0}^{M-1} z^{-k} = \frac{1}{M} \frac{1 - z^{-M}}{1 - z^{-1}} = \frac{1}{M} \frac{z^M - 1}{z^{M-1}(z - 1)}. \tag{5.135}$$

The polynomial expression leads to a nonrecursive implementation, whereas the rational form results in a recursive implementation (see Example 2.7).

To understand the equivalence of the two system functions in (5.135), we recall that the equation $z^M - 1 = 0$ has M roots given by

$$z_k = W_M^k, \quad k = 0, 1, \ldots, M - 1 \tag{5.136}$$

where $W_M = \exp(j2\pi/M)$. Therefore the system has M zeros given by (5.136), one pole at $z = 0$ and $M - 1$ poles at $z = 0$. The pole at $z = 1$ is canceled by the zero $z_0 = 1$. This results in a pole-zero pattern shown in Figure 5.27(a) for $M = 10$. The impulse response is real because the zeros are symmetric with respect to the real axis. The $(M - 1)$ zeros distributed over the unit circle keep the magnitude response small, except at $z = 1$, where the missing zero allows for larger values. This yields a "lowpass" filter with the magnitude response shown in 5.27(a). The notches of the filter, one for each zero, are located at $\omega_k = 2\pi k/M$ for $k = 1, 2, \ldots, M - 1$.
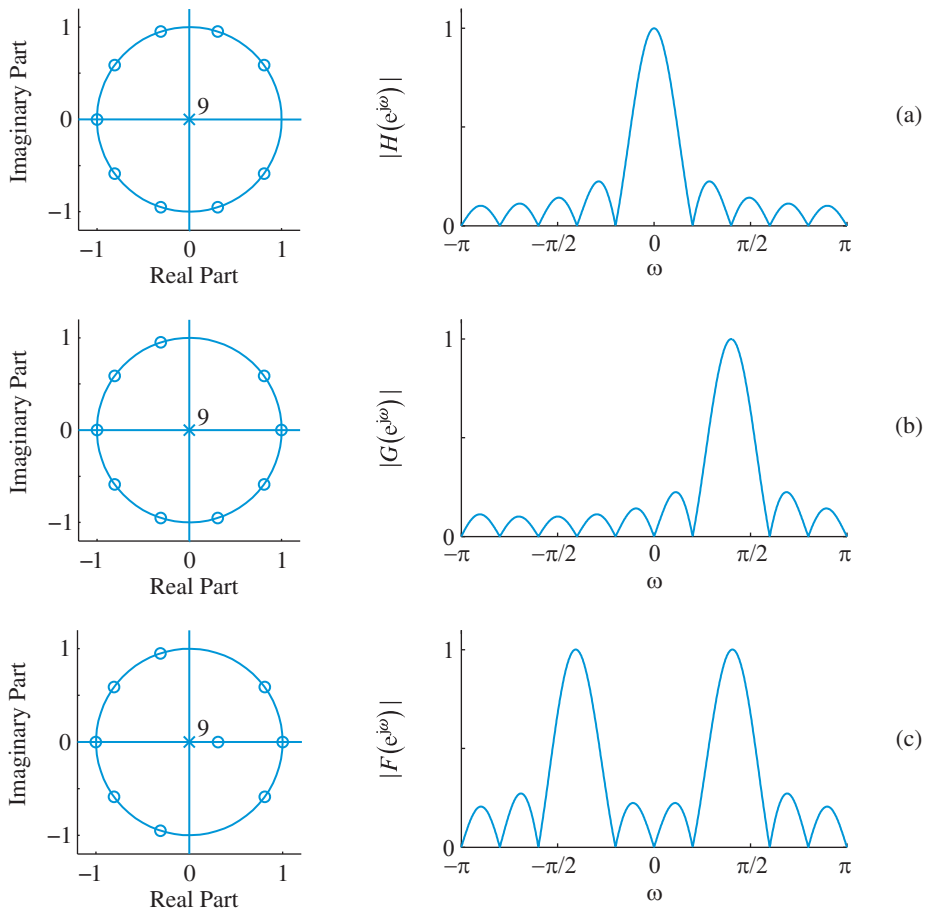
Figure 5.27 Pole-zero pattern and magnitude response for (a) a lowpass moving average filter $h[n] = u[k] - u[k - M]$, (b) a complex bandpass filter $g[k] = h[k] \exp(2\pi mk/M)$, and (c) a real bandpass filter $f[k] = h[k] \cos(2\pi mk/M)$. The responses have been normalized to have maximum gain one, $M = 10$, and $m = 2$.

Complex bandpass filters We can move the passband of the lowpass filter (5.135) to a frequency $\omega_m = 2\pi m/M$, using the frequency shifting property of the DTFT

$$e^{j\omega_m k} h[k] \xrightarrow{\text{DTFT}} H(e^{j[\omega - \omega_m]}). \qquad (5.137)$$

The system function of the resulting bandpass filter $g[k] = W_M^{mk} h[k]$ is given by

$$G(z) = \frac{1}{M} \sum_{k=0}^{M-1} \left(W_M^{-m} z \right)^{-k} = \frac{1}{M} \frac{1 - \left(W_M^{-m} z \right)^{-M}}{1 - \left(W_M^{-m} z \right)^{-1}}, \qquad (5.138)$$

$$= \frac{1}{M} \frac{1 - z^{-M}}{1 - W_M^m z^{-1}} = \frac{1}{M} \prod_{\substack{k=0 \\ k \neq m}}^{M-1} \left(1 - W_M^k z^{-1} \right). \qquad (5.139)$$

From (5.135) and (5.138) we have $G(z) = H(W_M^{-m}z)$, which corresponds to a counter-clockwise rotation of the pole-zero pattern by $2\pi m/M$ radians (see Section 3.4). According to (5.139) this is equivalent to the cancelation of zero at $z = W_M^m$ by the pole at the same location. The pole–zero pattern and the magnitude response of the complex bandpass filter are shown in Figure 5.27(b). Since the pole-zero pattern lacks symmetry about the real axis, the impulse response $g[k] = e^{j2\pi m/M}$, $k = 0, 1, \ldots, M-1$ is complex.

Real bandpass filters To obtain a bandpass filter with real impulse response we use the modulation property of the DTFT,

$$h[k] \cos \omega_{\mathrm{m}} k \xrightarrow{\;\text{DTFT}\;} \frac{1}{2}H(e^{j[\omega-\omega_{\mathrm{m}}]}) + \frac{1}{2}H(e^{j[\omega+\omega_{\mathrm{m}}]}). \tag{5.140}$$

The system function of a filter with $f[k] = \cos(2\pi m/M)k$ is given by

$$F(z) = \sum_{k=0}^{M-1} \cos\left(\frac{2\pi m}{M}k\right) z^{-k} = \frac{1}{2}\sum_{k=0}^{M-1}\left(W_M^{-mk} + W_M^{mk}\right) z^{-k}, \tag{5.141}$$

$$= \frac{1}{2}\frac{1 - z^{-M}}{1 - W_M^{-m}z^{-1}} + \frac{1}{2}\frac{1 - z^{-M}}{1 - W_M^{m}z^{-1}}, \tag{5.142}$$

$$= \frac{\left(1 - z^{-M}\right)\left(1 - z^{-1}\cos\frac{2\pi}{M}m\right)}{\left(1 - W_M^{m}z^{-1}\right)\left(1 - W_M^{-m}z^{-1}\right)}. \tag{5.143}$$

We note that (5.142) expresses the FIR system (5.141) as the parallel connection of two complex bandpass systems obtained by dropping two complex conjugate zeros. According to (5.143), this is equivalent to introducing a zero at the real part of the dropped complex conjugate zeros. The pole-zero pattern and magnitude response of the resulting bandpass filter are shown in Figure 5.27(c). Since the denominator of (5.143) can be written as $1 - (2\cos 2\pi m/M)z^{-1} + z^{-2}$, it is possible to obtain a recursive implementation with real coefficients. Removing the term $1 - z^{-1}\cos(2\pi m/M)$ from (5.143), yields a slightly different bandpass filter with real coefficients. However, the modulation property (5.140) does not hold in this case.

Frequency transformations The frequency modulation theorem provides a simple technique to generate bandpass filters from a lowpass prototype. An interesting simplification occurs if we set $\omega_{\mathrm{m}} = \pi$ in (5.137). Then, we can easily show that

$$g[n] = (-1)^n h[n] \xrightarrow{\;\text{DTFT}\;} G(e^{j\omega}) = H(e^{j[\omega-\pi]}), \tag{5.144}$$

that is, we can convert a lowpass filter into a highpass filter by changing the signs of the odd-numbered samples of its impulse response, and vice versa. The same trick can be used for the coefficients of the difference equation (see Tutorial Problem 16)

$$y[n] = -\sum_{k=1}^{N}(-1)^k a_k y[n-k] + \sum_{k=0}^{M}(-1)^k b_k x[n-k]. \tag{5.145}$$

This approach is intended to provide insight into the concept of frequency transformations that shift the frequency response of prototype lowpass filters. A more sophisticated methodology is provided in Chapter 11.

5.8   Relationship between magnitude and phase responses

The frequency response $H(e^{j\omega})$ is equal to the system function $H(z)$ evaluated on the unit circle. We wish to find out whether there is a z-transform $R(z)$ such that

$$R(z)|_{z=e^{j\omega}} = |H(e^{j\omega})|^2 = H(e^{j\omega})H^*(e^{j\omega}). \tag{5.146}$$

This requires finding a z-transform $V(z)$ such that $V(z)|_{z=e^{j\omega}} = H^*(e^{j\omega})$. Taking the complex conjugate of (5.2), we have

$$H^*(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h^*[k]e^{j\omega k} = \left. \sum_{k=-\infty}^{\infty} h^*[k]z^k \right|_{z=e^{j\omega}} = V(z)|_{z=e^{j\omega}}. \tag{5.147}$$

Comparing (5.2) and (5.147), we obtain

$$V(z) = \sum_{k=-\infty}^{\infty} h^*[k]z^k = \left[\sum_{k=-\infty}^{\infty} h[k](z^*)^k \right]^* = \left[\sum_{k=-\infty}^{\infty} h[k]\left(1/z^*\right)^{-k} \right]^*$$
$$= H^*(1/z^*). \tag{5.148}$$

Therefore, *the function* $V(z) = H^*(1/z^*)$ *is obtained by conjugating the coefficients of* $H(z)$ *and replacing everywhere* z^{-1} *by* z. If $h[n]$ has real values, we have the simplified form $V(z) = H(1/z)$. Hence, the desired z-transform function is

$$R(z) = H(z)H^*(1/z^*), \qquad \text{complex } h[n] \tag{5.149}$$
$$= H(z)H(1/z). \qquad \text{real } h[n] \tag{5.150}$$

If we consider systems with rational system functions of the form (5.77), we have

$$R(z) = |b_0|^2 \frac{\displaystyle\prod_{k=1}^{M} \left(1 - z_k z^{-1}\right)\left(1 - z_k^* z\right)}{\displaystyle\prod_{k=1}^{N} \left(1 - p_k z^{-1}\right)\left(1 - p_k^* z\right)}. \tag{5.151}$$

We note that for each pole p_k of $H(z)$, there are two poles of $R(z)$ at p_k and $1/p_k^*$. Similarly, for each zero z_k of $H(z)$, there are two zeros of $R(z)$ at z_k and $1/z_k^*$. Therefore, the poles and zeros of $R(z)$ occur in *conjugate reciprocal pairs*. If $w = re^{j\phi}$ is a pole (or zero),

then $(1/w^*) = \frac{1}{r}e^{j\phi}$ is a pole (or zero). Clearly, if w is inside the unit circle, then its conjugate reciprocal $1/w^*$ is outside the unit circle. If w is on the unit circle, then $1/w^*$ has to be in the same location on the unit circle. These ideas are illustrated in the following example.

Example 5.8

Consider a second-order FIR system with system function

$$H(z) = \left(1 - az^{-1}\right)\left(1 - bz^{-1}\right). \quad -1 < a < 1, \ -1 < b < 1 \tag{5.152}$$

This system has two real zeros inside the unit circle. The z-transform

$$R(z) = H(z)H(1/z) = \left(1 - az^{-1}\right)\left(1 - bz^{-1}\right)(1 - az)(1 - bz) \tag{5.153}$$

has two zeros from $H(z)$ ($z_1 = a$, $z_2 = b$) and two zeros from $H(1/z)$ ($z_3 = 1/a$, $z_4 = 1/b$). Given $R(z)$, by pairing different zeros, we can form four different second-order FIR systems:

$$H_1(z) = \left(1 - az^{-1}\right)\left(1 - bz^{-1}\right), \tag{5.154a}$$

$$H_2(z) = \left(1 - az^{-1}\right)(1 - bz), \tag{5.154b}$$

$$H_3(z) = (1 - az)\left(1 - bz^{-1}\right), \tag{5.154c}$$

$$H_4(z) = (1 - az)(1 - bz). \tag{5.154d}$$

As expected from (5.153), these systems have the same magnitude response but different phase responses. ∎

In general, given the rational $R(z)$ in (5.151), there are 2^{N+M} different systems $H_k(z)$ of the form (5.77) satisfying (5.146). All these systems have the same magnitude response but different phase responses. Therefore, given $|H(e^{j\omega})|$ we *cannot* uniquely identify the phase response $\angle H(e^{j\omega})$ without additional information. However, it is equally important to understand that the magnitude and phase responses *cannot* be specified independently because of the need for system realizability. This has very important implications in the filter design problem (see Chapter 10). A unique $H(z)$ can be specified by choosing the poles and zeros of $R(z)$ inside the unit circle (see minimum-phase systems in Section 5.10). The problem becomes more difficult if M and N are undefined or they tend to infinity. For arbitrary systems, knowledge about the magnitude does not provide information about the phase, and vice versa. The problem of obtaining the system function $H(z)$ from $R(e^{j\omega}) = |H(e^{j\omega})|^2$ or from the autocorrelation sequence $r[\ell] = \mathcal{Z}^{-1}\{R(z)\}$ is known as *spectral factorization*.

5.9   Allpass systems

The frequency response of an *allpass* system has constant magnitude ($G > 0$) at all frequencies, that is,

$$|H(e^{j\omega})| = G. \tag{5.155}$$

If we assume that $G = 1$, (5.35) and (5.43) imply that allpass systems preserve the power or energy of their input signals. In this sense, allpass systems are said to be *lossless* systems. The simplest allpass systems simply scale and delay the input signal. Hence,

$$H_{ap}(z) = Gz^{-k}. \tag{5.156}$$

A more interesting, non-trivial family of allpass systems (known as *dispersive* allpass systems) can be obtained by noting that the factors $(1 - p_k e^{-j\omega})$ and its complex conjugate $(1 - p_k^* e^{j\omega})$ have the same magnitude. Therefore, the system

$$H_k(z) = z^{-1} \frac{1 - p_k^* z}{1 - p_k z^{-1}} = \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}} \tag{5.157}$$

is an allpass. The term z^{-1}, which has magnitude one on the unit circle, has been introduced to make the system causal. Higher order allpass systems can be obtained by cascading multiple first-order sections, as

$$H_{ap}(z) = e^{j\beta} \prod_{k=1}^{N} \frac{z^{-1} - p_k^*}{1 - p_k z^{-1}}, \tag{5.158}$$

where β is a constant (usually $\beta = 0$). For systems with real coefficients, complex roots should appear in complex conjugate pairs. The parallel connection of allpass systems is, in general, not allpass. For example, the systems $H_1(z) = 1$ and $H_2(z) = z^{-1}$ are allpass but $H(z) = H_1(z) + H_2(z) = 1 + z^{-1}$ is not allpass because $H(e^{j\omega}) = 2\cos(\omega/2)$.

From (5.158) we note that each pole p_k of an allpass system should be accompanied by a *complex reciprocal* zero $1/p_k^*$. Since causal and stable allpass systems must have all poles inside the unit circle, all zeros are outside the unit circle. The key properties of allpass systems can be illustrated geometrically using the pole-zero plot in Figure 5.28. We note that the unit circle is the locus of all points Z such that

$$\frac{(\overline{P_k Z})}{(\overline{Z_k Z})} = \frac{\left| e^{j\omega} - 1/p_k^* \right|}{\left| e^{j\omega} - p_k \right|} = \frac{1}{|p_k|}, \quad |p_k| < 1 \tag{5.159}$$

that is, the ratio of the distances of Z from the pole $p_k = r_k e^{j\phi_k}$ and the zero $1/p_k^* = (1/r_k)e^{j\phi_k}$ is constant. The phase of this pole-zero pair, which is $\alpha_k(\omega) = \Theta_k(\omega) - \Phi_k(\omega)$, decreases monotonically from π to zero as ω increases from ϕ_k to $\phi_k + \pi$. Since $H_k(e^{j\omega}) = -p_k^*(e^{j\omega} - 1/p_k^*)/(e^{j\omega} - p_k)$, we note that $\angle H_k(e^{j\omega}) = \angle(-p_k^*) + \alpha_k(\omega)$. Since p_k is
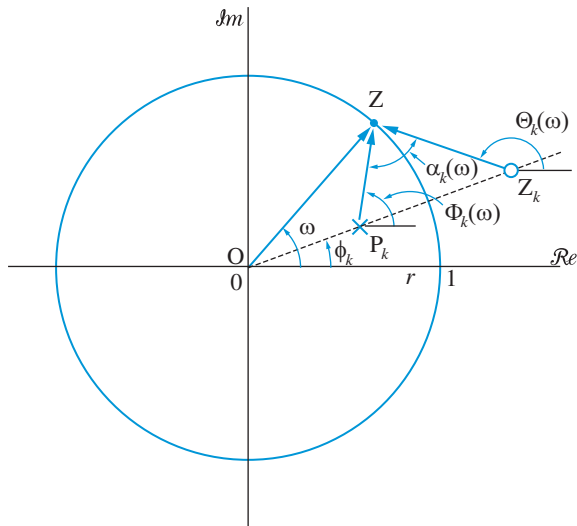
Figure 5.28 Allpass systems properties. The locus of a point the ratio of whose distances from two fixed points is constant, is a circle, called the circle of Apollonius, or the Apollonian circle.

constant, the functions $\angle H_k(e^{j\omega})$ and $\alpha_k(\omega)$ have the same shape; only their range of values is different.

The magnitude, phase, and group-delay responses of a first-order allpass system (5.157), with $p_k = r_k e^{j\phi_k}$, are given by (see Problem 17)

$$\left| H_k(e^{j\omega}) \right| = 1, \tag{5.160}$$

$$\angle H_k(e^{j\omega}) = -\omega - 2\tan^{-1}\frac{r_k \sin(\omega - \phi_k)}{1 - r_k \cos(\omega - \phi_k)}, \tag{5.161}$$

$$\tau_k(\omega)] = \frac{1 - r_k^2}{1 + r_k^2 - 2r_k \cos(\omega - \phi_k)}. \tag{5.162}$$

Figure 5.29 shows the pole-zero plot, the log-magnitude response, the phase response, and the group delay for $p_k = 0.8e^{j\pi/4}$. The continuous phase response $\Psi(\omega)$ has been evaluated using (5.60). The MATLAB function `angle`, which computes the principal value, produces the discontinuous dashed curve.

Since $r_k < 0$ for causal and stable allpass systems, the group delay (5.162) is always positive. From the geometrical interpretation of Figure 5.28 and the phase plot in Figure 5.29, we conclude that the phase decreases monotonically from $\angle H(e^{-j\pi})$ to $\angle H(e^{-j\pi}) - 2\pi$ for each pole. Therefore, the phase response of an Nth order allpass system decreases monotonically in the range

$$\Psi(-\pi) \le \Psi(\omega) \le \Psi(-\pi) - 2\pi N \text{ for } -\pi \le \omega \le \pi. \tag{5.163}$$
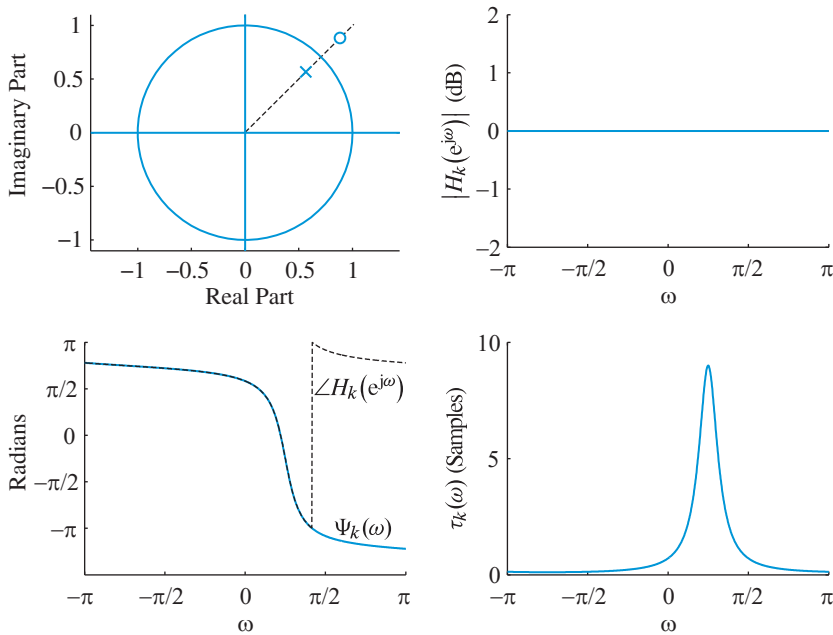
Figure 5.29 Pole-zero plot, log-magnitude response, phase response, and group delay for a first-order allpass filter with a pole at $p_k = 0.8e^{j\pi/4}$. The dashed line shows the principal value of the phase response.
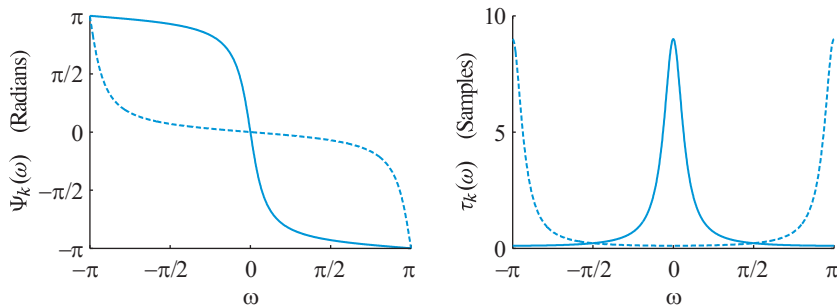
Figure 5.30 Phase and group-delay responses for a first-order allpass filter with a pole at $p = 0.8$ (solid line) and a pole at $p = -0.8$ (dashed line).

For real and complex conjugate poles, see Figures 5.30 and 5.31, the phase response has odd symmetry about $\omega = 0$ and the group delay has even symmetry about $\omega = 0$. Therefore, for allpass systems with real coefficients the unwrapped phase response changes monotonically from $N\pi$ to $-N\pi$ as ω varies from $-\pi$ to π rads.

From (5.158), with $\beta = 0$ and $N = 2$, we can show that the system function of a second-order allpass system can be expressed as

$$H_{ap}(z) = \frac{a_2^* + a_1^* z^{-1} + z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} = z^{-2} \frac{1 + a_1^* z + a_2^* z^2}{1 + a_1 z^{-1} + a_2 z^{-2}}, \tag{5.164}$$
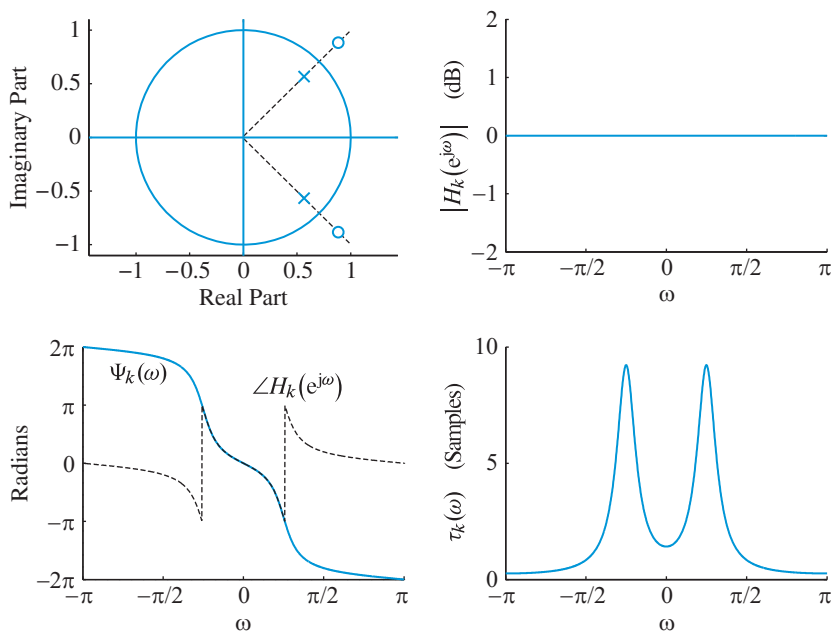
Figure 5.31 Pole-zero plot, log-magnitude response, phase response, and group delay for a second-order allpass filter with poles at $p_k = 0.8e^{\pm j\pi/4}$. The dashed line shows the principal value of the phase response.

where $a_1 = -(p_1 + p_2)$ and $a_2 = p_1 p_2$. Therefore, in general, we have

$$H_{\text{ap}}(z) = z^{-N} \frac{1 + a_1^* z + \cdots + a_N^* z^N}{1 + a_1 z^{-1} + \cdots + a_N z^{-N}} = z^{-N} \frac{A^*(1/z^*)}{A(z)}. \tag{5.165}$$

We note that for real a_k, we have $A^*(1/z^*) = A(1/z)$. The polynomials $A(z)$ and $z^{-N} A^*(1/z^*)$ are said to form a *conjugate reverse* pair. If a_k are real, the terms of the difference equation implementing (5.165) can be grouped in such a way that only N multiplications are required.

Allpass systems have many applications, including the design of phase compensators used to mitigate phase or group delay distortion without altering the magnitude response, frequency transformations, and multirate filtering.

Example 5.9   Allpass reverberator unit

To prevent coloration of the input sound by the comb reverberator in Example 5.7, Schroeder and Logan (1961) introduced an allpass reverberator unit. The desired system is obtained by creating a comb filter from the allpass prototype (5.157). The result is a reverberator with system function

$$H(z) = \frac{z^{-D} - a}{1 - az^{-D}}. \quad -1 < a < 1 \tag{5.166}$$
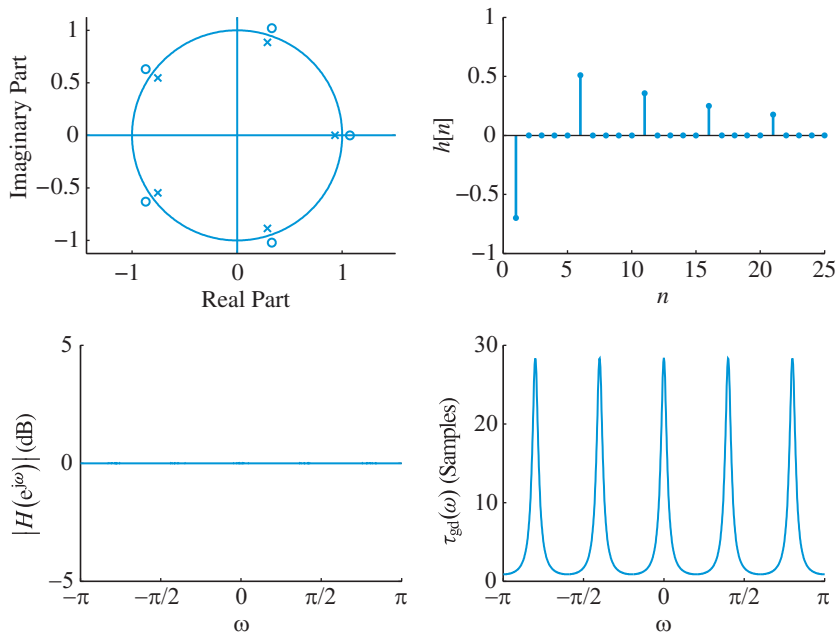
Figure 5.32 Pole-zero pattern, impulse response, magnitude response, and group delay of an allpass reverberator with $a = 0.7$ and $D = 5$.

This system can be implemented with $2D$ delays by the input–output equation

$$y[n] = ay[n - D] - ax[n] + x[n - D]. \tag{5.167}$$

If we define the sequence $w[n - D] \triangleq ay[n - D] + x[n - D]$, we can express (5.167) by the difference equations

$$y[n] = -ax[n] + w[n - D], \tag{5.168}$$

$$w[n] = ay[n] + x[n], \tag{5.169}$$

which provide an implementation with D delays. Figure 5.32 shows the pole-zero pattern, the impulse response, the magnitude response, and the group delay for $a = 0.7$ and $D = 5$. More details about the allpass reverberator unit are provided in Tutorial Problem 22. ■

The comb and allpass units can be combined to form more realistic sounding reverberators. One of the earliest reverberators uses four comb units in parallel followed by two allpass units in series (Schroeder and Logan (1961)); another approach replaces the multiplier a in the comb unit by a system function $G(z)$ (Moore (1990)). More information about reverberation is provided in Moorer (1979) and in Problem 68.

5.10 | Invertibility and minimum-phase systems

An LTI system $H(z)$ with input $x[n]$ and output $y[n]$ is said to be *invertible* if we can uniquely determine $x[n]$ from $y[n]$. The system $H_{\text{inv}}(z)$ that produces $x[n]$ when excited by $y[n]$ is called the *inverse* system. By definition, the cascade of a system and its inverse is equal to the identity system, that is,

$$h[n] * h_{\text{inv}}[n] = \delta[n], \tag{5.170}$$

$$H(z)H_{\text{inv}}(z) = 1, \tag{5.171}$$

where $h[n]$ and $h_{\text{inv}}[n]$ are the corresponding impulse responses. Given $h[n]$, we can obtain $h_{\text{inv}}[n]$ by solving the convolution equation (5.170). A simpler approach is to use the algebraic equation (5.171). For the rational system function (5.77), we have

$$H_{\text{inv}}(z) = \frac{1}{H(z)} = \frac{A(z)}{B(z)}. \tag{5.172}$$

Therefore, the zeros of $H(z)$ become the poles of its inverse system, and vice versa. A causal and stable system $H(z)$ should have its poles inside the unit circle; its zeros can be anywhere. In inverse filtering applications, the system $H_{\text{inv}}(z) = 1/H(z)$ should be causal and stable as well. A causal and stable LTI system with a causal and stable inverse is known as a *minimum-phase* system. Thus, the impulse response sequences $h[n]$ and $h_{\text{inv}}[n]$ should be causal and absolutely summable. Sometimes, to allow for poles or zeros on the unit circle, we only require for the impulse responses to have finite energy. The rational system (5.77) is minimum-phase if both its poles and zeros are inside the unit circle.

Minimum phase and allpass decomposition We shall now show that any system with a rational system function can be decomposed into a minimum-phase system and an allpass system. To this end, suppose that $H(z)$ has one zero $z = 1/a^*$, where $|a| < 1$ outside the unit circle, and all other poles and zeros are inside the unit circle. Then, we can factor $H(z)$ as

$$H(z) = H_1(z)\left(z^{-1} - a^*\right), \tag{5.173}$$

where, by definition, $H_1(z)$ is minimum phase. Equation (5.173) can be written as

$$H(z) = H_1(z)\left(1 - az^{-1}\right)\frac{\left(z^{-1} - a^*\right)}{\left(1 - az^{-1}\right)}, \tag{5.174}$$

where $H_1(z)(1 - az^{-1})$ is minimum phase and $(z^{-1} - a^*)/(1 - az^{-1})$ is allpass. If we repeat this process for every zero outside the unit circle, we obtain

$$H(z) = H_{\text{min}}(z)H_{\text{ap}}(z). \tag{5.175}$$

Thus, any rational system function can be decomposed into the product of a minimum-phase system function and an allpass system function. This procedure is illustrated in the following example.

Example 5.10   Minimum-phase/allpass decomposition

Consider a causal and stable system specified by the system function

$$H(z) = \frac{1 + 5z^{-1}}{1 + \frac{1}{2}z^{-1}}, \tag{5.176}$$

where $H(z)$ has a pole inside the unit circle at $z = -1/2$ and a zero outside the unit circle at $z = -5$. To reflect the zero inside the unit circle without affecting the magnitude response, we should choose an allpass system according to (5.174). Therefore, starting with (5.176) and using (5.174), we successively have

$$H(z) = 5\frac{z^{-1} + \frac{1}{5}}{1 + \frac{1}{2}z^{-1}} = 5\frac{z^{-1} + \frac{1}{5}}{1 + \frac{1}{2}z^{-1}} \frac{1 + \frac{1}{5}z^{-1}}{1 + \frac{1}{5}z^{-1}} = 5\underbrace{\frac{1 + \frac{1}{5}z^{-1}}{1 + \frac{1}{2}z^{-1}}}_{H_{\min}(z)} \underbrace{\frac{z^{-1} + \frac{1}{5}}{1 + \frac{1}{5}z^{-1}}}_{H_{\mathrm{ap}}(z)} . \tag{5.177}$$

Comparing $H(z)$ with $H_{\min}(z)$, we see that a quick way to obtain the minimum-phase system is by replacing each factor $\left(1 + az^{-1}\right)$, where $|a| > 1$, by a factor of the form $a\left(1 + \frac{1}{a}z^{-1}\right)$. ∎

The minimum delay property We shall now derive a property that explains the origin of the name minimum-phase systems. The systems $H(z)$ and $H_{\min}(z)$ in (5.175) have obviously the same magnitude response. Therefore, reflecting a zero into its conjugate reciprocal location does *not* change the magnitude response of a system. To see the effect on the phase response, consider a zero $z = re^{j\theta}$ inside the unit circle ($r < 1$). From (5.88) and (5.89), the negative phase shift $-\Psi(\omega)$ (*phase-lag*) and the group delay are given by

$$-\Psi(\omega) = -\tan^{-1}\frac{\sin(\omega - \theta)}{1/r - \cos(\omega - \theta)}, \tag{5.178}$$

$$\tau(\omega) = \frac{r - \cos(\omega - \theta)}{(r + 1/r) - 2\cos(\omega - \theta)}. \tag{5.179}$$

Suppose now that we reflect the zero $z = re^{j\theta}$ to its conjugate reciprocal location $1/z^* = (1/r)e^{j\theta}$, which is outside the unit circle. If we replace r by $1/r$ in (5.178), the phase-lag increases because the denominator decreases. A geometrical proof follows from Figure 5.33 because we can easily see that $\beta_{\mathrm{out}}(\omega) \geq \beta_{\mathrm{in}}(\omega)$ (see equation (5.112)). Similarly, the numerator of (5.179) increases because r is replaced by $1/r$, which is larger; the denominator remains the same. Therefore, reflecting a zero outside the unit circle at its conjugate reciprocal location increases the group delay. The plots show the phase-lag (5.178)
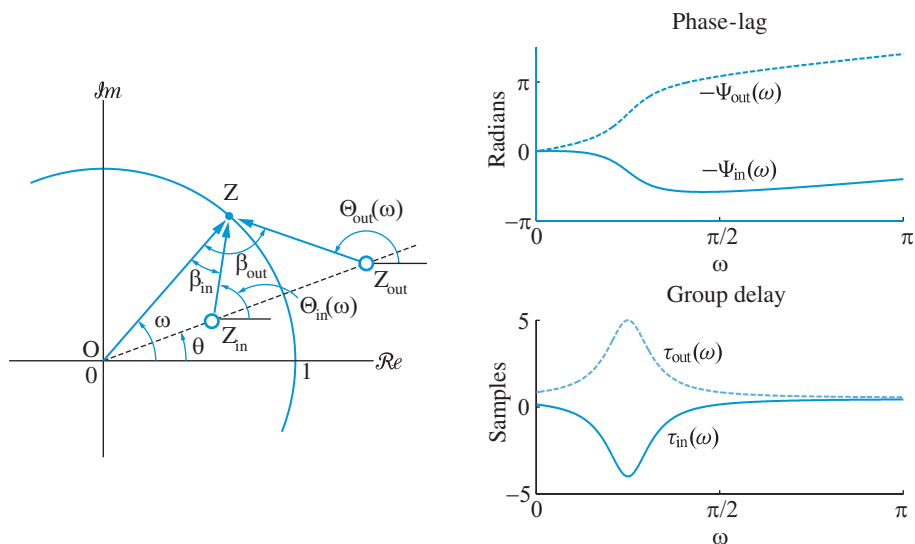
Figure 5.33 (a) Geometrical proof that a zero outside the unit circle introduces a larger phase-lag than a zero inside the unit circle. (b) Phase response and group delay for a zero at $z_{in} = 0.8e^{j\pi/4}$ and its conjugate reciprocal zero $z_{out} = (1/0.8)e^{j\pi/4}$.

and the group delay (5.179) for $z = 0.8e^{j\pi/4}$ and its reciprocal conjugate $1/z^*$. We conclude that *a minimum-phase system has the minimum phase-lag (algebraically) and the minimum group delay among all systems with the same magnitude response*. Although the terms minimum phase-lag or minimum group delay system would be more appropriate, the term minimum phase has been established in the literature.

Maximum- and mixed-phase systems A causal and stable system with a rational system function is called *maximum phase* if all its zeros are outside the unit circle. A system with arbitrary $H(z)$ is maximum phase, if it is causal, stable, and $H(z) \neq 0$ for $|z| < 1$. A system that is neither minimum phase nor maximum phase is called a *mixed-phase* system.

Example 5.11   Minimum-, maximum-, and mixed-phase systems

Consider a minimum-phase system with system function

$$H_{\min}(z) = \left(1 - az^{-1}\right)\left(1 - bz^{-1}\right)$$

$$= 1 - (a + b)z^{-1} + abz^{-2}, \qquad (5.180)$$

where $-1 < a < 1$ and $-1 < b < 1$. This system has two zeros inside the unit circle at $z = a$ and $z = b$. If we only reflect one zero outside the unit circle, we obtain the following mixed-phase systems:

$$H_{\text{mix1}}(z) = a\left(1 - a^{-1}z^{-1}\right)\left(1 - bz^{-1}\right), \tag{5.181}$$

$$H_{\text{mix2}}(z) = b\left(1 - az^{-1}\right)\left(1 - b^{-1}z^{-1}\right). \tag{5.182}$$

Reflecting both zeros outside the unit circle yields the maximum-phase system

$$H_{\text{max}}(z) = ab\left(1 - a^{-1}z^{-1}\right)\left(1 - b^{-1}z^{-1}\right)$$
$$= ab - (a+b)z^{-1} + z^{-2} \tag{5.183}$$

Comparing (5.180) and (5.183) shows that $H_{\text{max}}(z)$ is the reverse polynomial of $H_{\text{min}}(z)$. This can be formally expressed as

$$H_{\text{max}}(z) = z^{-2}H_{\text{min}}(1/z), \tag{5.184}$$

which illustrates how the zeros of $H_{\text{min}}(z)$, which are inside the unit circle, are reflected outside the unit circle to become the zeros of $H_{\text{max}}(z)$.

Figure 5.34 shows the magnitude response, the phase response (principal value), the phase-lag response $-\Psi(\omega)$, and the group delay of the minimum-, mixed-, and maximum-phase systems discussed. As expected (a) all systems have the same magnitude response, (b) the minimum (maximum) phase system has the minimum (maximum) group delay and the algebraically minimum (maximum) phase-lag, and (c) the group delay and phase-lag of mixed-phase systems are between those of minimum- and maximum-phase systems. ∎

Equation (5.184) can be easily generalized for any polynomial with complex or real coefficients. Indeed, given a minimum-phase system

$$A_{\text{min}}(z) = 1 + a_1 z^{-1} + \cdots + a_N z^{-N}, \tag{5.185}$$

the following *conjugate reverse* polynomial yields a maximum-phase system

$$A_{\text{max}}(z) = a_N^* + a_{N-1}^* z^{-1} + \cdots + z^{-N} = z^{-N} A_{\text{min}}^*(1/z^*). \tag{5.186}$$

From (5.165) we can easily conclude that the system $A_{\text{max}}(z)/A_{\text{min}}(z)$ is allpass.

Inversion of nonminimum-phase systems In many applications the magnitude and phase of a signal are distorted when it passes through a system $H(z)$. This distortion can be cured, at least in principle, by cascading an *equalizer* $H_{\text{eq}}(z)$ with $H(z)$. The output of the combined system will be distortionless if $H(z)H_{\text{eq}}(z) = Gz^{-n_d}$, where G and n_d are arbitrary constants. The equalizer $H_{\text{eq}}(z) = Gz^{-n_d}/H(z)$ is causal and stable if $H(z)$ is minimum phase. If $H(z)$ is nonminimum phase, we use

$$H_{\text{eq}}(z) = \frac{Gz^{-n_d}}{H_{\text{min}}(z)}, \tag{5.187}$$

where $H_{\text{min}}(z) = H(z)/H_{\text{ap}}(z)$. Then, the overall system is given by

$$H(z)H_{\text{eq}}(z) = GH_{\text{ap}}(z)z^{-n_d}, \tag{5.188}$$
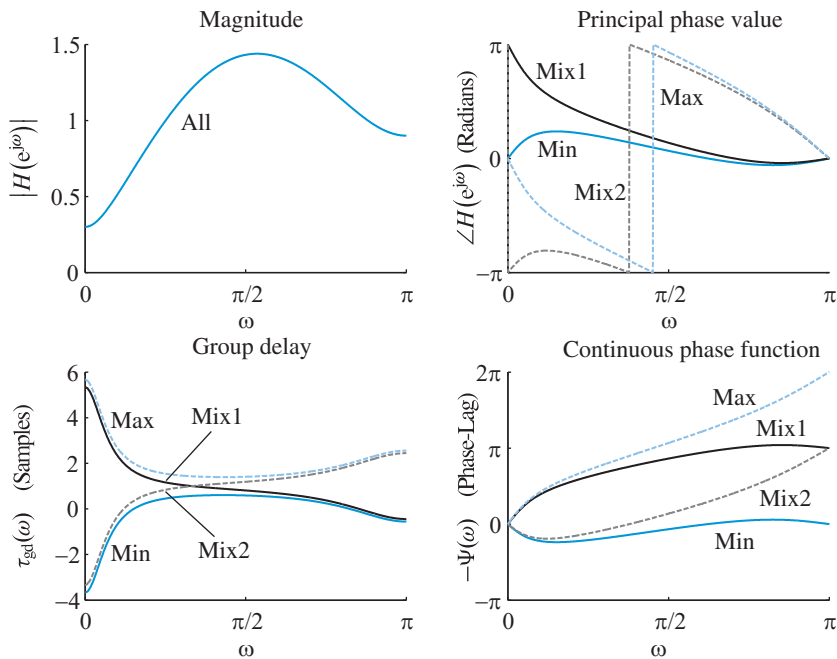
Figure 5.34 Magnitude response, phase response (principal value), group delay, and phase-lag for the minimum-phase, mixed-phase, and maximum-phase systems discussed in Example 5.11.

which is an allpass system. Therefore, $H_{eq}(z)$ "equalizes" the magnitude response to G and modifies the phase response to $\angle H_{ap}(\omega)$ (see Problem 18).

5.11   Transform analysis of continuous-time LTI systems

In this section we discuss the analysis of continuous-time LTI systems using transform techniques. The objective is not a systematic coverage of the subject; we rather plan to draw certain parallels with the concepts we have developed for discrete-time systems. This approach will enhance the understanding of transform domain analysis of LTI systems and will provide sufficient knowledge to use analog filter design techniques to design digital filters in Chapter 11.

5.11.1   System function and frequency response

In Section 5.1 we showed that the complex exponential sequences are eigenfunctions of discrete-time LTI systems. If we express the complex variable z in polar form as $z = r e^{j\omega}$, we have

$$x[n] = r^n e^{j\omega n} \overset{\mathcal{H}}{\longmapsto} y[n] = H\left(r e^{j\omega}\right) r^n e^{j\omega n}, \quad \text{all } n \tag{5.189}$$

where $H(z) = H(re^{j\omega})$ is the system function

$$H(re^{j\omega}) = \sum_{n=-\infty}^{\infty} h[n]r^{-n}e^{-j\omega n}. \tag{5.190}$$

The exponential sequence r^{-n} controls the growth of $e^{-j\omega n}$ based on whether $r < 1$ or $r > 1$. In the complex z-plane, the contour corresponding to $|z| = r = 1$ is the unit circle. Evaluating the system function on the unit circle yields the frequency response $H(e^{j\omega})$. The frequency variable ω is the angle between the vector to a point z on the unit circle and the positive real axis. As a result, $H(e^{j\omega})$ is periodic with period 2π radians. Equivalently, because n is an integer, the eigenfunctions $e^{j\omega n}$ are periodic in ω with period 2π radians. Essentially, *it is the discrete nature of n that enforces the polar form $z = re^{j\omega}$, the unit circle as the "frequency-axis," and the annular shape of ROC in discrete-time systems.*

In contrast, due to the continuity of t, the complex exponentials $e^{j\Omega t}$ are not periodic in Ω. Thus, we need a different representation for the frequency variable Ω in the complex plane. A natural way to control the growth of $e^{j\Omega t}$ is to multiply with the real exponential $e^{\sigma t}$, where σ is a real variable. The result is a complex exponential signal

$$x(t) = e^{(\sigma + j\Omega)t} = e^{st}, \quad \text{all } t \tag{5.191}$$

where the complex variable $s = \sigma + j\Omega$ is expressed in rectangular form.

In an exactly parallel manner with discrete-time systems, we can show that the complex exponential functions (5.191) are eigenfunctions of continuous-time LTI systems. Indeed, since all continuous-time LTI systems are described by the convolution integral

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau, \tag{5.192}$$

we obtain

$$y(t) = \int_{-\infty}^{\infty} h(\tau)e^{s(t-\tau)}d\tau = e^{st}\int_{-\infty}^{\infty} h(\tau)e^{-s\tau}d\tau.$$

Thus, we have

$$x(t) = e^{st} \xrightarrow{\mathcal{H}} y(t) = H(s)e^{st}, \quad \text{all } t \tag{5.193}$$

where the eigenvalue $H(s)$ is given by the *system function*

$$H(s) \triangleq \int_{-\infty}^{\infty} h(\tau)e^{-s\tau}d\tau, \tag{5.194}$$

which is the Laplace transform of the impulse response $h(t)$. When $s = j\Omega$, that is, when (5.194) is evaluated on the imaginary axis, the system function $H(s)$ corresponds to the *frequency response function* $H(j\Omega)$ and (5.193) shows that the response to a complex exponential is a complex exponential with the same frequency.

The Laplace transform

In general, the *Laplace transform* of an arbitrary signal $x(t)$ is defined as

$$X(s) \triangleq \int_{-\infty}^{\infty} x(t) e^{-st} dt. \tag{5.195}$$

We note that $X(s)$ is a function of the independent complex variable s that appears in the exponent of e^{-st}. Furthermore, the frequency variable Ω is the imaginary part of $s = \sigma + j\Omega$, expressed in rectangular form. To appreciate the significance of this observation, we express the Laplace transform in terms of $s = \sigma + j\Omega$ as

$$X(\sigma + j\Omega) = \int_{-\infty}^{\infty} x(t) e^{-\sigma t} e^{-j\Omega t} dt. \tag{5.196}$$

We note that the Laplace transform of $x(t)$ can be viewed as the CTFT of the exponentially weighted signal $e^{-\sigma t} x(t)$. The exponential $e^{-\sigma t}$ decays in time if $\sigma > 0$ and grows in time if $\sigma < 0$. The ROC of the Laplace transform of $x(t)$ consists of all values of s for which $e^{-\sigma t} x(t)$ is absolutely integrable. Therefore, the Laplace transform may exist even if the CTFT does not exist.

To determine the ROC for causal signals, which is sufficient for the issues discussed in this book, we first note that

$$|X(s)| \leq \int_{0}^{\infty} |x(t)| e^{-\sigma t} dt. \tag{5.197}$$

If $|X(s)|$ is finite for $s = \sigma_0 + j\Omega_0$, then it is finite for $s = \sigma_0 + j\Omega$, all Ω. Furthermore, since $e^{-\sigma t} \leq e^{-\sigma_0 t}$ for $\sigma \geq \sigma_0$, we conclude that the ROC will include all values of s for which $\mathcal{Re}\{s\} \geq \sigma_0$ (a *right-half plane*). As in the case of a z-transform, to find $x(t)$ we need to know not only $X(s)$ but also the ROC; otherwise, $x(t)$ is not unique. However, if $x(t)$ is a causal signal, this problem does not arises because the ROC is a right-half plane $\mathcal{Re}\{s\} \geq \sigma_0$. If the ROC includes the $j\Omega$ axis in its interior, then $X(j\Omega)$ exists and equals the Fourier transform of $x(t)$.

Example 5.12

Let us determine the Laplace transform of the causal exponential $x(t) = e^{-at} u(t)$, $a > 0$ in Figure 5.35(a). First of all, we note that

$$\int_{0}^{\infty} e^{-(\sigma + a)t} dt < \infty, \tag{5.198}$$

for $\sigma + a > 0$ or $\mathcal{Re}\{s\} > -a$. Thus, the ROC is the shaded area in Figure 5.35(b). Substitution of $x(t)$ into the definition (5.195) yields

$$X(s) = \int_{0}^{\infty} e^{-(a+s)t} dt = -\frac{e^{-(s+a)t}}{(s+a)}\Big|_{0}^{\infty} = 0 - \frac{-1}{s+a} = \frac{1}{s+a}. \tag{5.199}$$
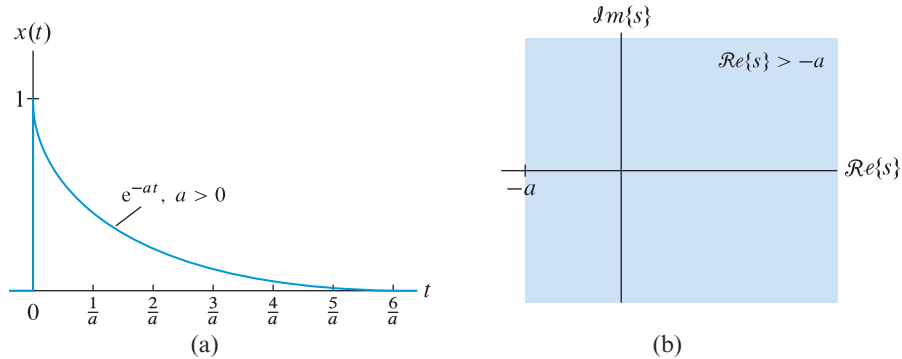
Figure 5.35 The causal exponential signal and its ROC in Example 5.12: (a) signal, (b) ROC.

Therefore, we have the following Laplace transform pair

$$x(t) = e^{-at}u(t) \overset{\mathcal{L}}{\longleftrightarrow} X(s) = \frac{1}{s+a}, \quad \text{ROC: } \mathcal{Re}\{s\} \geq -a \tag{5.200}$$

where we use the symbol $\overset{\mathcal{L}}{\longleftrightarrow}$ to denote a Laplace transform pair. Since $a > 0$, the ROC includes the imaginary axis $s = j\Omega$. Therefore, $X(j\Omega)$ provides the CTFT of $x(t)$. For $a = 0$, (5.200) provides the Laplace transform of the unit step function. The transform (5.200) holds for complex values of a; however, the ROC is now specified by the inequality $\mathcal{Re}\{s\} \geq \mathcal{Re}\{a\}$. ∎

Based on the formal definition (5.195), we can determine the properties of Laplace transforms and evaluate the transforms of common signals. One of the most important properties of the Laplace transform is that it is linear:

$$x(t) = a_1 x_1(t) + a_2 x_2(t) \overset{\mathcal{L}}{\longleftrightarrow} X(s) = a_1 X_1(s) + a_2 X_2(s). \tag{5.201}$$

Example 5.13
Consider the sinusoidal signal $x(t) = \cos(\Omega_0 t)u(t)$. Because

$$\int_0^\infty |e^{-st} \cos \Omega_0 t| dt \leq \int_0^\infty e^{-\sigma t} dt < \infty, \tag{5.202}$$

for all $\mathcal{Re}\{s\} > 0$, the ROC is the right-half plane. The Laplace transform of $x(t)$ can be computed as

$$X(s) = \int_0^\infty \cos(\Omega_0 t) e^{-st} dt = \frac{1}{2} \int_0^\infty \left[e^{-(s+j\Omega_0)t} + e^{-(s-j\Omega_0)t} \right] dt$$

$$= \frac{1}{2} \frac{1}{s+j\Omega_0} + \frac{1}{2} \frac{1}{s-j\Omega_0} = \frac{s}{s^2 + \Omega_0^2}.$$

This yields the following Laplace transform pair

$$x(t) = \cos(\Omega_0 t)u(t) \overset{\mathcal{L}}{\longleftrightarrow} X(s) = \frac{s}{s^2 + \Omega_0^2}. \quad \text{ROC: } \mathcal{R}e\{s\} > 0 \tag{5.203}$$

■

If a signal $x(t)$ is delayed in time, $y(t) = x(t - \tau)$, then with a simple change of variables we can easily show that

$$Y(s) = \int_{-\infty}^{\infty} x(t - \tau)e^{-st}dt = e^{-s\tau} \int_{-\infty}^{\infty} x(\nu)e^{-s\nu}d\nu = e^{-s\tau}X(s). \tag{5.204}$$

Taking the Laplace transform of the convolution integral (2.105) with respect to t and using (5.204) we obtain

$$Y(s) = \int_{-\infty}^{\infty} x(\tau)e^{-s\tau}H(s)d\tau = H(s)X(s), \tag{5.205}$$

which is the convolution theorem of the Laplace transform.

As an application of the convolution theorem (5.205) to the integration property we note that integration can be expressed as

$$y(t) = \int_{-\infty}^{t} x(\tau)d\tau = u(t) * x(t) \overset{\mathcal{L}}{\longleftrightarrow} Y(s) = \frac{1}{s}X(s). \tag{5.206}$$

Since integration and differentiation are inverse operations, we have the pair

$$y(t) = \frac{dx(t)}{dt} \overset{\mathcal{L}}{\longleftrightarrow} Y(s) = sX(s). \tag{5.207}$$

These Laplace transform pairs and properties will be sufficient for our needs.

Example 5.14   A simple RC lowpass filter

The output voltage $y(t)$ of an RC circuit is related to the input voltage $x(t)$ through the following first-order constant-coefficient differential equation

$$RC\frac{y(t)}{dt} + y(t) = x(t). \tag{5.208}$$

Using the linearity and differentiation properties, we obtain

$$RCsY(s) + Y(s) = X(s), \tag{5.209}$$

which yields the system function

$$H(s) = \frac{Y(s)}{X(s)} = \frac{1}{1 + RCs}. \tag{5.210}$$

The impulse response can be easily found using (5.200) as

$$h(t) = \frac{1}{RC} e^{-t/RC} u(t). \tag{5.211}$$

■

Systems with rational system functions

Most continuous-time LTI systems consist entirely of lumped-parameter elements, such as resistors, capacitors, and inductors. Such systems are described by linear differential equations of the form

$$\sum_{k=0}^{N} a_k \frac{d^k y(t)}{dt^k} = \sum_{k=0}^{M} b_k \frac{d^k x(t)}{dt^k}, \tag{5.212}$$

where a_k and b_k are constant coefficients involving the element values, as in (5.208). Taking the Laplace transform of both sides yields

$$(a_0 + a_1 s + \cdots + a_N s^N) Y(s) = (b_0 + b_1 s + \cdots + b_M s^M) X(s), \tag{5.213}$$

because the Laplace transform of the kth derivative corresponds to multiplication by the s^k power. From (5.205) and (5.213) the system function is

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 + b_1 s + \cdots + b_M s^M}{a_0 + a_1 s + \cdots + a_N s^N} \triangleq \frac{B(s)}{A(s)}, \tag{5.214}$$

which is a rational function in s. Now the equation $B(s) = 0$ has M roots (zeros z_k) and $A(s) = 0$ has similarly N roots (poles p_k). Thus, we can express (5.214) in terms of the zero factors $(s - z_k)$ and the pole factors $(s - p_k)$ as follows:

$$H(s) = G \frac{(s - z_1)(s - z_2) \ldots (s - z_M)}{(s - p_1)(s - p_2) \ldots (s - p_N)}, \tag{5.215}$$

where the gain $G = b_M / a_N$. The system function becomes zero at the zeros z_k and infinite at the poles p_k. At all other values of s within the ROC, $H(s)$ takes a finite nonzero value. Figure 5.36 shows a three-dimensional representation of $|H(s)|$ as a function of s for a quadrant of the s-plane. For any rational $H(s)$ the total number of zeros is equal to the total number of poles, if we take into account poles and zeros at zero and infinity. Furthermore, for any physical system $M \leq N$.

Assuming that the poles p_k are real or complex but distinct, we can decompose the rational system function $H(s)$ in (5.215) as the partial fraction

$$H(s) = \frac{A_1}{s - p_1} + \frac{A_2}{s - p_2} + \cdots + \frac{A_N}{s - p_N}. \tag{5.216}$$
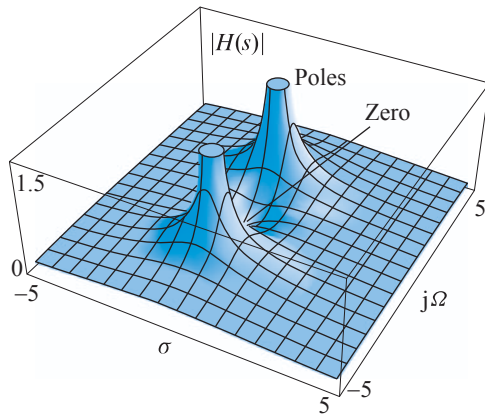
Figure 5.36 The magnitude of $H(s) = \frac{s+2}{s^2+4s+5}$ as a function of $s = \sigma + j\Omega$.

To determine A_1, we multiply both sides of (5.216) by the factor $(s - p_1)$ to get

$$(s - p_1)H(s) = A_1 + \frac{s - p_1}{s - p_2}A_2 + \cdots + \frac{s - p_1}{s - p_N}A_N. \tag{5.217}$$

If we let $s = p_1$ on both sides of (5.217), then all the A_k terms except the first will have zero coefficients. For this term, we have $A_1 = (s - p_1)H(s)|_{s=p_1}$. The other coefficients can be expressed in similar form,

$$A_k = (s - p_k)H(s)|_{s=p_k}. \tag{5.218}$$

To determine A_k using this expression, we ignore the term $(s - p_k)$ in the factored form (5.215) and evaluate the rest of the expression with $s = p_k$. This leads to the following graphical interpretation:

$$A_k = G\frac{\text{Product of distances from each zero to } p_k}{\text{Product of distances from each pole to } p_k}. \tag{5.219}$$

Obviously, we do *not* include the distance of pole p_k from itself. Since all physical systems are causal, using the Laplace transform pair (5.216), we obtain

$$h(t) = \sum_{k=1}^{N} A_k e^{p_k t} u(t). \tag{5.220}$$

Pairs of complex conjugate terms can be expressed in real form as follows:

$$A_i e^{p_i t} + A_i^* e^{p_i^* t} = 2|A_i|e^{\sigma_i t} \cos(\Omega_i t + \angle A_i), \tag{5.221}$$

where $p_i = \sigma_i + j\Omega_i$. This procedure is illustrated in the following example.

Example 5.15

Consider the system function

$$H(s) = \frac{B(s)}{A(s)} = \frac{s^2 + s - 2}{s^3 + 3s^2 + 7s + 5}. \tag{5.222}$$

The roots of $A(s) = 0$, which can be computed using function `p=roots(a)` with `a=[1,3,7,1]`, are $p_1 = -1 + j2$, $p_2 = -1 - j2$, and $p_3 = -1$. Thus, we have

$$H(s) = \frac{A_1}{s + 1 - j2} + \frac{A_2}{s + 1 + j2} + \frac{A_3}{s + 1}. \tag{5.223}$$

Using (5.218) we obtain

$$A_1 = (s - p_1)H(s)|_{s=p_1} = \frac{s^2 + s - 2}{(s + 1 + j2)(s + 1)}\bigg|_{s=-1+j2} = \frac{1}{4}(3 + j),$$

$$A_2 = (s - p_2)H(s)|_{s=p_2} = \frac{s^2 + s - 2}{(s + 1 - j2)(s + 1)}\bigg|_{s=-1-j2} = \frac{1}{4}(3 - j),$$

$$A_3 = (s - p_3)H(s)|_{s=p_3} = \frac{s^2 + s - 2}{(s + 1 - j2)(s + 1 + j2)}\bigg|_{s=-1} = -\frac{1}{2}.$$

The coefficients of the partial fraction expansion (5.223) can be easily obtained, using the MATLAB function `residue`, as follows:

```
>> b=[1,1,-2]; a=[1,3,7,5]; [A,p]=residue(b,a)

A =
    0.7500 + 0.2500i
    0.7500 - 0.2500i
   -0.5000

p =
   -1.0000 + 2.0000i
   -1.0000 - 2.0000i
   -1.0000
```

The impulse response of the system is given by the inverse Laplace transform of (5.223) which is

$$h(t) = A_1 e^{p_1 t} u(t) + A_1^* e^{p_1^* t} u(t) + A_3 e^{-p_3 t} u(t). \tag{5.224}$$

Combining the complex conjugate terms yields

$$h(t) = -0.5e^{-t}u(t) + 1.5811e^{-t}\cos(2t + 0.3218)u(t). \tag{5.225}$$

We note that the real pole $p_3 = -1$ contributes a real exponential term and the complex conjugate poles $p_{1,2} = -1 \pm j2$ contribute a sinusoidal term with an exponential envelope. The integral of $|h(t)|$ is finite because $e^{-t}u(t)$ tends to zero as t increases; hence, the causal system (5.225) is stable. ∎

In general, for a causal LTI system with a rational system function to be stable, its impulse response (5.220) should satisfy the stability condition (2.52). Hence we have

$$\int_0^\infty |h(t)|\mathrm{d}t \le \sum_{k=1}^N |A_k| \int_0^\infty e^{\mathcal{R}e\{p_k\}t}\mathrm{d}t < \infty. \tag{5.226}$$

The last integral is finite if $\sigma_k = \mathcal{R}e\{p_k\} < 0$ because in this case the exponential functions $e^{\sigma_k t}u(t)$ decay asymptotically to zero. Therefore, a continuous-time LTI system with a rational system function having poles at $s = p_k$ is stable if all poles are located to the left-half of the $j\Omega$-axis in the s-plane (left-half plane), that is,

$$H(s) = G\frac{\prod_{k=1}^M (s - z_k)}{\prod_{k=1}^N (s - p_k)} \text{ is stable } \Leftrightarrow \sigma_k = \mathcal{R}e\{p_k\} < 0, \text{ all } k. \tag{5.227}$$

Therefore, the poles determine the shape of the impulse response and the stability of a continuous-time LTI system. The zeros have no effect on the stability of the system and their effect on the shape of impulse response is minor.

5.11.4   Frequency response from pole-zero location

The geometrical approach used to evaluate the frequency response of discrete-time systems with rational system functions can be easily applied to continuous-time systems. However, there is a simple but fundamental difference: instead of the unit circle $z = e^{j\omega}$, the frequency response of continuous-time systems is evaluated on the imaginary axis $s = j\Omega$. Replacing s with $j\Omega$ in (5.214) and (5.215), yields

$$H(j\Omega) = H(s)|_{s=j\Omega} = \frac{\sum_{k=0}^M b_k(j\Omega)^k}{\sum_{k=0}^N a_k(j\Omega)^k} = G\frac{\prod_{k=1}^M (j\Omega - z_k)}{\prod_{k=1}^N (j\Omega - p_k)}. \tag{5.228}$$

As illustrated in Figure 5.37, the zero factor $(j\Omega - z_k) = Q_k e^{j\Theta_k}$ represents a vector from the zero z_k to the point $s = j\Omega$ (zero vector), whereas the pole factor $(j\Omega - p_k) = R_k e^{j\Phi_k}$ represents a vector from the pole p_k to the point $s = j\Omega$ (pole vector). Then, we can express (5.228) as

$$H(j\Omega) = |G|\frac{\prod_{k=1}^M Q_k(j\Omega)}{\prod_{k=1}^N R_k(j\Omega)} \exp\left[\angle G + \sum_{k=1}^M \Theta_k(j\Omega) - \sum_{k=1}^N \Phi_k(j\Omega)\right], \tag{5.229}$$

where we indicate all quantities that are functions of frequency Ω. Therefore,

$$|H(j\Omega)| = |G|\frac{\text{Product of zero vectors to } s = j\Omega}{\text{Product of pole vectors to } s = j\Omega}, \tag{5.230}$$

$$\angle H(j\Omega) = \angle G + \text{Sum of zero angles to } s = j\Omega$$
$$-\text{Sum of pole angles to } s = j\Omega, \tag{5.231}$$
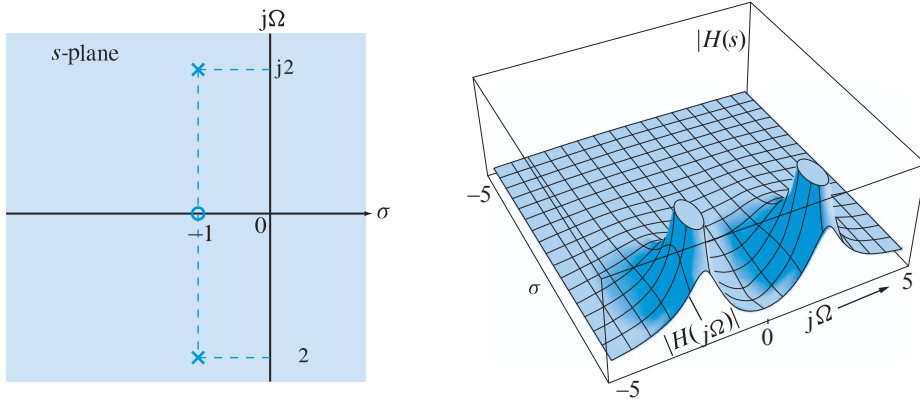
Figure 5.37 Geometrical evaluation of the continuous-time frequency response $H(j\Omega)$ from the pole-zero pattern. The "frequency-domain" is the $j\Omega$ axis.
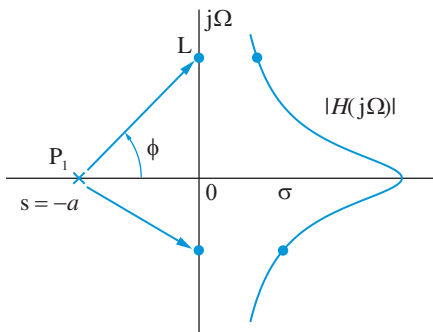
Figure 5.38 Geometrical explanation of the frequency response characteristics of a continuous-time system with one real pole.

where the angles are measured with respect to the positive real axis. To compute the frequency response $H(j\Omega)$, we choose a point $s = j\Omega$ on the imaginary axis, connect all poles and zeros to this point, and determine $|H(j\Omega)|$ and $\angle H(j\Omega)$ using (5.230) and (5.231). We then repeat this procedure for all frequencies of interest.

MATLAB **computations** The pole-zero pattern of (5.215) can be obtained using the function `[p,z]=splane(b,a)`. To compute the frequency response $H(j\Omega)$ we use the function `H=freqs(b,a,Omega)`, which evaluates the polynomials at each frequency point `s = j*Omega` and then divides the numerator response by the denominator response using `H=polyval(b,s)./polyval(a,s)`. The group delay is obtained by evaluating the derivative of the continuous phase response function. The frequency `Omega` in radians/second is obtained by `Omega=2*pi*F`, where `F` is the frequency in Hz. The discussion regarding the numerical computation of angles in the discrete-time case holds for the continuous-time case.

Real poles If $H(s)$ has one real pole, from Figure 5.38 and (5.215) we obtain

$$H(s) = \frac{G}{s+a}, \quad |H(j\Omega)| = \frac{G}{(\overline{P_1 L})}, \quad \text{and } \angle H(j\Omega) = -\phi, \tag{5.232}$$

for $a > 0$ and $G > 0$. Clearly, the distance $(\overline{P_1 L})$ from the pole $s = -a$ to the point $j\Omega$ increases as Ω moves away from pole. Hence, $|H(j\Omega)|$ decreases monotonically as $|\Omega|$ increases from zero to infinity. Since $(\overline{P_1 L}) = \sqrt{\Omega^2 + a^2}$, the magnitude response attains $1/\sqrt{2}$ of its peak value at $\Omega = a$. The phase response $\angle H(j\Omega)$ decreases from $\pi/2$ to $-\pi/2$ as Ω increases from $-\infty$ to ∞ (see Figure 5.38). Similar results hold if $H(s)$ has two real poles.

Complex conjugate poles Consider a second-order system described by the linear constant-coefficient differential equation

$$\frac{d^2 y(t)}{dt^2} + 2\zeta \Omega_n \frac{dy(t)}{dt} + \Omega_n^2 y(t) = \Omega_n^2 x(t). \tag{5.233}$$

The system function of this system is given by the Laplace transform

$$H(s) = \frac{Y(s)}{X(s)} = \frac{\Omega_n^2}{s^2 + 2\zeta \Omega_n s + \Omega_n^2}. \tag{5.234}$$

The parameter ζ is known as the *damping ratio* and the parameter $\Omega_n \geq 0$ as the *undamped natural frequency*, for reasons to be seen below. This system has two complex conjugate poles if $(2\zeta \Omega_n)^2 - 4\Omega_n^2 < 0$ or equivalently $-1 < \zeta < 1$. The poles of the system are given by

$$p_1 = -\zeta \Omega_n + j\Omega_n \sqrt{1 - \zeta^2} \triangleq -\alpha + j\beta, \tag{5.235a}$$

$$p_2 = -\zeta \Omega_n - j\Omega_n \sqrt{1 - \zeta^2} \triangleq -\alpha - j\beta. \tag{5.235b}$$

For the system to be stable $\zeta > 0$; therefore, the range of the damping ratio is $0 < \zeta < 1$. The impulse response, obtained as in Example 5.15, is given by

$$h(t) = \frac{\Omega_n}{\sqrt{1 - \zeta^2}} e^{-\zeta \Omega_n t} \left[\sin\left(\Omega_n \sqrt{1 - \zeta^2} \right) t \right] u(t). \tag{5.236}$$

For $0 < \zeta < 1$ the impulse response has a damped oscillatory behavior with frequency $\beta = \Omega_n \sqrt{1 - \zeta^2} < \Omega_n$. In the undamped case, $\zeta = 0$, the poles move on the $j\Omega$ axis. In this case, the frequency of oscillation in $h(t)$ is equal to the undamped natural frequency Ω_n and the system becomes an analog sinusoidal generator (oscillator).

We next investigate the frequency domain properties of (5.234) using geometric arguments. We first note that the poles are on a circle with radius Ω_n because

$$|p_1|^2 = |p_2|^2 = \alpha^2 + \beta^2 = \Omega_n^2. \tag{5.237}$$
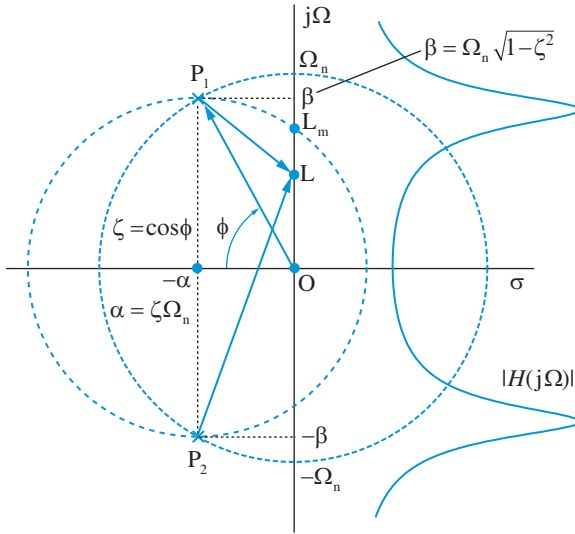
Figure 5.39 Geometrical explanation of the frequency response characteristics of a continuous-time system with two complex conjugate poles (resonator). Note that the magnitude response of a resonator can be obtained by shifting the response of the one-pole system in Figure 5.38 to the resonant frequencies $\pm\Omega_m$.

From (5.230) and Figure 5.39, the magnitude response can be expressed as

$$|H(j\Omega)| = \frac{\Omega_n^2}{(\overline{P_1 L})(\overline{P_2 L})}. \tag{5.238}$$

The form of $|H(j\Omega)|$ depends on the behavior of the product $(\overline{P_1 L})(\overline{P_2 L})$ as L moves along the $j\Omega$ axis. To analyze this behavior, we draw a circle with center at $s = -\alpha$ and radius β. If $\beta > \alpha$, this circle intersects the $j\Omega$ axis at the points $\pm j\Omega_m$. Then, as L moves from the origin to infinity, the product $(\overline{P_1 L})(\overline{P_2 L})$ decreases at first, reaching a minimum at a point $L_m = j\Omega_m$, and then increases. Hence, $|H(j\Omega)|$ increases as Ω goes from 0 to Ω_m, and then decreases monotonically. The result is a bandpass filter with maximum response at $\Omega = \Omega_m$. From the geometrical construction in Figure 5.39 we can easily show that $(\overline{P_1 O})(\overline{P_2 O}) = \alpha^2 + \beta^2$ and $(\overline{P_1 L_m})(\overline{P_2 L_m}) = 2\alpha\beta$. Substituting into (5.237) and (5.238) yields

$$\Omega_m = \sqrt{\beta^2 - \alpha^2} = \Omega_n\sqrt{1 - 2\zeta^2}, \tag{5.239}$$

$$\frac{|H(j\Omega_m)|}{|H(j0)|} = \frac{\alpha^2 + \beta^2}{2\alpha\beta} = \frac{1}{2}\left(\frac{\beta}{\alpha} + \frac{\alpha}{\beta}\right). \tag{5.240}$$

If $\alpha \ll \beta$, or equivalently $\zeta \approx 0$, the poles move very close to the $j\Omega$ axis. In this case, $|H(j\Omega)|$ takes large values for Ω in the vicinity of Ω_n, $\Omega_m \approx \Omega_n \approx \beta$, and $\overline{P_2 L} \approx 2j\beta$. Therefore, using the relation $(\overline{P_1 L})(\overline{P_2 L}) \approx [\alpha + j(\Omega - \beta)]2j\beta$, we obtain the approximation

$$|H(j\Omega)| \approx \frac{\Omega_n^2/(2\beta)}{\sqrt{\alpha^2 + (\Omega - \beta)^2}}, \quad \alpha \ll \beta \tag{5.241}$$

which shows that the frequency response in the vicinity of pole P_1 is not affected significantly by the pole P_2 and vice versa. The magnitude response (5.241) assumes $1/\sqrt{2}$ of its peak value at the frequencies $\Omega_1 = \beta - \alpha$ and $\Omega_2 = \beta + \alpha$. The relative bandwidth of the resonance is defined by

$$B \triangleq \frac{\Omega_2 - \Omega_1}{\Omega_n} = \frac{2\alpha}{\Omega_n} = 2\zeta. \tag{5.242}$$

In conclusion, as ζ decreases from 1 towards 0, the relative bandwidth B decreases, and the frequency response becomes sharper and more frequency selective.

If $\beta < \alpha$, the circle does not intersect the $j\Omega$ axis, whereas for $\beta = \alpha$ the circle is tangent to the axis at the origin. In the last two cases $|H(j\Omega)|$ has a maximum at $\Omega = 0$, that is, the filter has lowpass characteristics.

5.11.5   Minimum-phase and allpass systems

The definitions of allpass and minimum-phase systems are the same for discrete-time and continuous-time systems; however, the poles and zeros are constrained by symmetries with respect to the $j\Omega$ axis instead of the unit circle.

By definition, the magnitude response is given by

$$|H(j\Omega)|^2 = H(j\Omega)H^*(j\Omega). \tag{5.243}$$

If $h(t)$ is real, then

$$H^*(j\Omega) = H(-j\Omega). \tag{5.244}$$

Next we note that $H(s)|_{s=j\Omega} = H(j\Omega)$. Therefore, we obtain

$$|H(j\Omega)|^2 = H(s)H(-s)|_{s=j\Omega}. \tag{5.245}$$

The poles of $V(s) = H(s)H(-s)$ occur in pairs, so if there is a pole at $s = p_k$, then there is also a pole at $s = -p_k$. The same argument applies to the zeros. Consequently, to determine $H(s)$ from the poles and zeros of $V(s)$, we choose one pole or zero from each pair. This is the spectral factorization problem for continuous-time systems with rational system functions. Clearly, the system $H(s)$ cannot be uniquely determined from $V(s)$.

Minimum-phase systems A minimum-phase system $H_{\min}(s)$ has, by definition, all its poles and zeros on the left-half plane. Therefore, the system $H_{\min}(s)$ and its inverse $1/H_{\min}(s)$ are both causal and stable. Consider the minimum-phase system in Figure 5.40. If we reflect the zero about the $j\Omega$ axis, there is no change in the magnitude response; only the phase response changes. From the geometrical construction we can easily see that $\angle H_{\min}(j\Omega) \le \angle H(j\Omega)$, which is also illustrated in Figure 5.40. Therefore, a minimum-phase system has the smallest phase shift among all systems with the same magnitude response. Clearly, a rational minimum-phase system can be uniquely determined from its magnitude response $|H(j\Omega)|$ through spectral factorization by choosing the poles and zeros on the left-half plane.
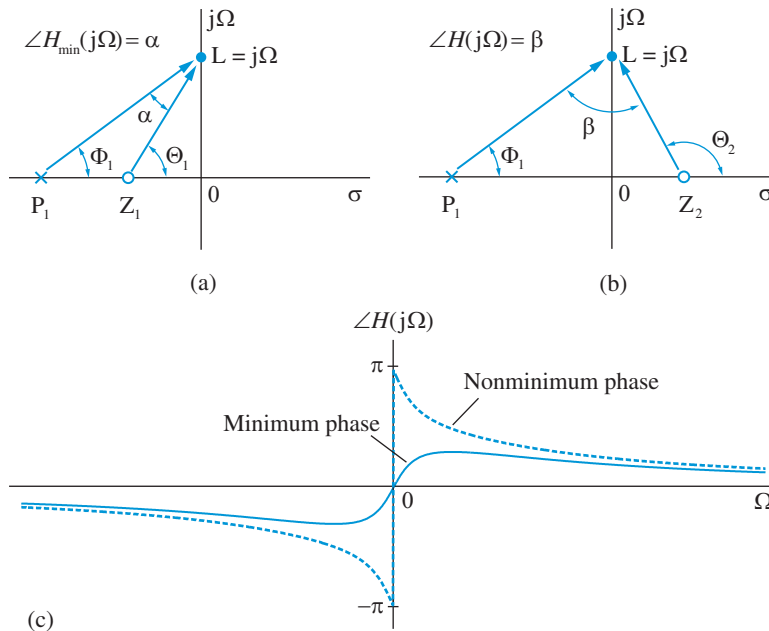
Figure 5.40 Geometrical constructions to determine the phase response of a minimum-phase system (a), and a nonminimum-phase system (b), with the same magnitude response. The resulting phase responses are shown in (c).

Allpass systems Although the poles of a stable system should be on the left-half plane, the zeros can be everywhere. Consider a system with a real pole at $s = -a$ $(a > 0)$ and a symmetric zero at $s = a$, as shown in Figure 5.41(a). From this figure, we see that for any point along the jΩ-axis, the pole and zero vectors have equal lengths. Thus, $|H(j\Omega)| = (\overline{P_kL})/(\overline{Z_kL}) = 1$, that is, the system is allpass. The phase of the frequency response is $\Theta_k - \Phi_k$, or, since $\Theta_k = \pi - \Phi_k$, we obtain

$$\angle H(j\Omega) = \pi - 2\tan^{-1}\frac{\Omega}{a}. \tag{5.246}$$

Furthermore, we can show that $\angle H(j\Omega) = \alpha$. Therefore, $\angle H(j\Omega)$ decreases monotonically from 2π to zero as Ω increases from $-\infty$ to ∞. The group delay is

$$\tau(\Omega) = \frac{2/a}{1 + (\Omega/a)^2}. \tag{5.247}$$

Since a is positive, the group-delay function $\tau(\Omega)$ is always positive.

Consider next a system with two complex conjugate poles $p_k = -a \pm jb$ $(a > 0)$ on the left-half plane and two complex conjugate zeros $z_k = a \pm jb$ symmetrically located on the right-half plane. The complex conjugate symmetry is necessary for the system to have real coefficients. The geometrical construction in Figure 5.41(b) shows that $(\overline{P_kL})(\overline{\bar{P}_kL}) = (\overline{Z_kL})(\overline{\bar{Z}_kL})$. Thus, the system is allpass because it has unit magnitude response for all $s = j\Omega$. With geometrical arguments, we can also show that the phase response is given
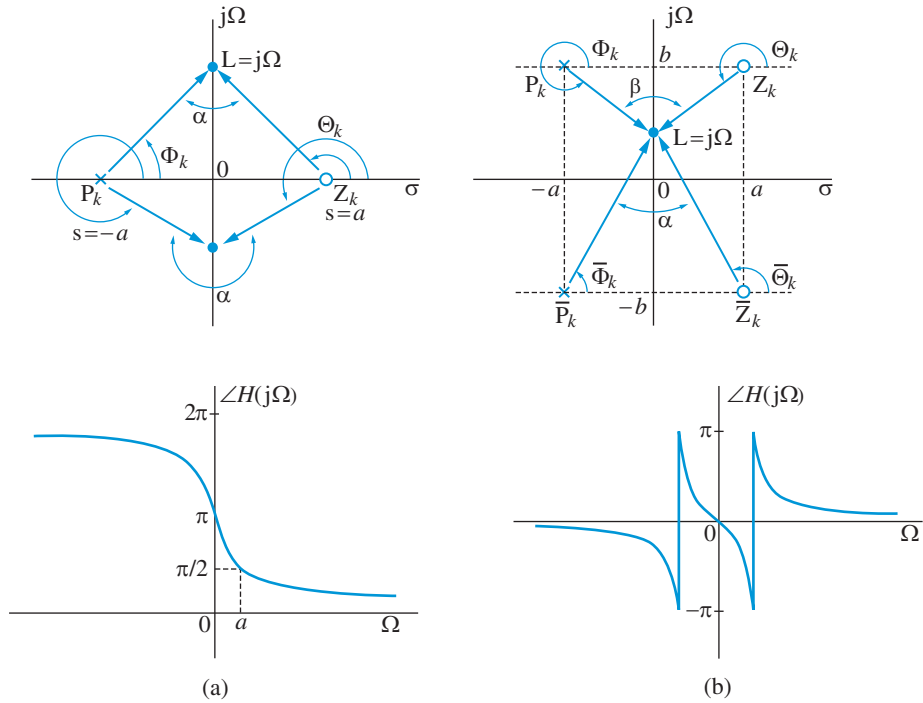
Figure 5.41 Geometrical constructions for computation of the magnitude and phase response of continuous-time allpass systems with (a) a real pole and zero, and (b) complex conjugate pairs of poles and zeros.

by $\angle H(j\Omega) = \alpha - \beta$. As Ω increases from zero to infinity, the phase response decreases monotonically from zero to -2π. Also, as Ω decreases from zero to negative infinity, the phase response increases monotonically from zero to 2π.

The magnitude response of a system with rational system function is

$$|H(j\Omega)|^2 = H(s)H(-s)|_{s=j\Omega} = \left. \frac{B(s)}{A(s)} \frac{B(-s)}{A(-s)} \right|_{s=j\Omega}. \tag{5.248}$$

Therefore, the system is allpass if $A(s) = B(-s)$, that is, the zeros are mirror images of the poles about the $j\Omega$ axis and vice versa.

Since higher-order allpass systems can be obtained by in series connection of first-order and second-order systems, we conclude that an Nth order allpass system has the following properties:

1. The zeros and poles are symmetric with respect to the $j\Omega$ axis, that is, if the poles are p_k, then the zeros are $-p_k^*$. Therefore, the system function is

$$H(s) = \frac{(s + p_1^*) \dots (s + p_N^*)}{(s - p_1) \dots (s - p_N)}. \tag{5.249}$$

2. The phase response is monotonically decreasing from $2\pi N$ to zero as Ω increases from $-\infty$ to ∞. This is illustrated in Figure 5.41(a) for $N = 1$; however, we typically plot the principal value as shown in Figure 5.41(b).
3. The group-delay response is positive for every value of Ω because it is equal to the sum of N components of the form (5.247).

Minimum-phase and allpass decomposition A nonminimum-phase system function can be decomposed into a product of a minimum phase and an allpass system function, that is,

$$H(s) = H_{\min}(s)H_{\text{ap}}(s), \tag{5.250}$$

using the following process:

1. For each zero in the right-half plane, include a pole and a zero at its mirror position in the left-half plane.
2. Assign the left-half plane zeros and the original poles to $H_{\min}(s)$.
3. Assign the right-half plane zeros and the left-half plane poles introduced in step 1 to $H_{\text{ap}}(s)$.

This procedure, which is illustrated in Problem 21, is used to design equalizers for nonminimum-phase systems.

5.11.6 Ideal filters

The ideal continuous-time delay system should satisfy the following condition

$$y(t) = Gx(t - t_{\text{d}}), \tag{5.251}$$

for any $t_{\text{d}} \geq 0$. The frequency response $H(j\Omega) = Ge^{-j\Omega t_{\text{d}}}$ of the ideal delay has constant magnitude and linear phase. Since the $H(s) = Ge^{-st_{\text{d}}}$ is not rational, the ideal delay is not easily realizable in practice.

Ideal filters have distortionless response at a certain band of frequencies and zero response at the remaining frequencies. For example, the frequency response of the ideal lowpass filter is given by

$$H_{\text{lp}}(j\Omega) = \begin{cases} e^{-j\Omega t_d}, & |\Omega| \leq \Omega_{\text{c}} \\ 0. & \text{otherwise} \end{cases} \tag{5.252}$$

The impulse response is

$$h_{\text{lp}}(t) = \frac{\Omega_{\text{c}}}{\pi} \frac{\sin \Omega_{\text{c}}(t - t_d)}{\Omega_{\text{c}}(t - t_d)}. \tag{5.253}$$

Since $h_{\text{lp}}(t) \neq 0$ for $t < 0$ and $\int |h_{\text{lp}}(t)|dt = \infty$, the ideal lowpass filter is noncausal and unstable. Therefore, it is not practically realizable. In practice, we can realize a variety of frequency responses that approach the ideal one to various degrees of approximation (see Chapters 10 and 11).

Learning summary

- The response of a stable LTI system to an everlasting complex exponential sequence is a complex exponential sequence with the same frequency; only the amplitude and phase are changed by the system. More specifically,

$$x[n] = A e^{j(\omega n + \phi)} \overset{\mathcal{H}}{\longmapsto} y[n] = A|H(e^{j\omega})| e^{j[\omega n + \phi + \angle H(e^{j\omega})]},$$

where $H(e^{j\omega})$ is the frequency response function of the system. The complex exponential sequences are said to be eigenfunctions of LTI systems.

- The response of an LTI system to a periodic input sequence $x[n]$ is a periodic sequence $y[n]$ with the same fundamental period N, that is,

$$c_k^{(y)} = H\left(\frac{2\pi}{N}k\right) c_k^{(x)}, \quad -\infty < k < \infty$$

where $c_k^{(x)}$ and $c_k^{(y)}$ are the DTFS coefficients of $x[n]$ and $y[n]$, respectively.

- The response of an LTI system to an aperiodic input signal $x(t)$ with Fourier transform $X(e^{j\omega})$ is a signal $y(t)$ with Fourier transform given by

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}),$$

which corresponds to point-by-point weighting of the input frequency components by the frequency response function.

- A system with distortionless response is defined by $y[n] = Gx[n-n_d]$, where $G > 0$ and n_d are constants. The frequency response function of distortionless systems has constant magnitude, $|H(e^{j\omega})| = G$, and linear phase $\angle H(e^{j\omega}) = -\omega n_d$; deviations from these conditions result in magnitude and phase distortions, respectively.

- The shape of the magnitude and phase frequency responses is determined by the locations of the poles and zeros with respect to the unit circle. Poles (zeros) close to the unit circle amplify (attenuate) input frequency components corresponding to the angle of these poles (zeros).

- A system with constant magnitude response $|H(e^{j\omega})| = G$ is called allpass. A causal and stable LTI system which has a causal and stable inverse is known as a minimum-phase system. Systems with rational system functions are minimum phase if all poles and zeros are inside the unit circle.

- The magnitude and phase responses of an arbitrary LTI system are independent. However, for minimum-phase systems the magnitude (phase) response uniquely specifies the phase (magnitude) response to within a scale factor. Every nonminimum-phase system can be expressed as the cascade connection of a minimum-phase system and an allpass system.

TERMS AND CONCEPTS

Allpass system Systems that have constant magnitude (>0) at all frequencies which are obtained by placing a complex reciprocal zero for each pole inside the unit circle.

Comb filter These are filters with multiple passbands and stopbands and are obtained by placing several poles near the unit circle.

Continuous-phase function A phase function that varies continuously without any jumps of 2π due to periodicity, denoted by $\Psi(\omega)$. Also known as an unwrapped-phase function.

Delay distortion A distortion in the shape of the response if the phase response is not a linear function of ω, defined by $\tau_{pd}(\omega) = -\angle H(\omega)/\omega$. Also known as the phase distortion.

Discrete-time oscillator A marginally stable system that has poles on the unit circle. Useful for generating sinusoidal carrier signals.

Discrete-time resonator A system that has a large magnitude response (that is, it resonates) in the vicinity of a pole location. It is essentially a bandpass filter.

Distortionless system A system whose input $x[n]$ and output $y[n]$ have the same shape, that is, $y[n] = Gx[n - n_d]$ or $H(\omega) = Ge^{-j\omega n_d}$.

Eigenfunctions of LTI systems The complex exponential, $e^{j\omega n}$, signals are the eigenfunctions since they are not distorted as they travel from the input to the output of an LTI system.

Energy or power gain The logarithm of $|H(\omega)|^2$, measured in decibels, is called the energy or power gain. Termed as attenuation if the value is negative.

Frequency response function It is the response of a stable LTI system to the complex exponential signal. It is denoted by $H(\omega)$ and is an eigenvalue of an LTI system.

Group delay Defined as the negative of the slope of the phase response, $\tau_{gd}(\omega) = -d\Psi(\omega)/d\omega$. Useful in checking the linearity of the phase response.

Ideal frequency-selective filters Has a distortionless response over one or more

frequency bands and zero response elsewhere. Major categories are: lowpass (LPF), highpass (HPF), bandpass (BPF), and bandstop (BSF) filter.

Invertible system If we can determine the input $x[n]$ uniquely for each output $y[n]$ the system is invertible.

Linear FM (LFM) A sinusoidal signal with a frequency that grows linearly with time.

Magnitude distortion A system introduces magnitude distortion if $|H(\omega)| \neq$ a constant.

Magnitude response The magnitude, $|H(\omega)|$, of the frequency response function is called the magnitude response. It is also known as the gain of the system.

Maximum-phase system An anticausal and stable system with an anticausal and stable inverse is called a maximum-phase system. It has all poles and zeros outside the unit circle and imparts the maximum phase or group delay to the input signal.

Minimum-phase system A causal and stable system with a causal and stable inverse is called a minimum-phase system. It has all poles and zeros inside the unit circle and imparts the minimum phase or group delay to the input signal.

Mixed-phase system It is a system that is neither minimum phase nor maximum phase and has all poles inside the unit circle but zeros can be inside or outside the unit circle.

Notch filter These are filters with perfect null at certain frequencies and are obtained by placing zeros at those frequencies.

Phase distortion A distortion in the shape of the response if the phase response is not a linear function of ω, defined by $\tau_{pd}(\omega) = -\angle H(\omega)/\omega$. Also known as the delay distortion.

Phase response The angle, $\angle H(\omega)$, of the the frequency response function is called the phase response.

Practical or nonideal filters Approximation of ideal filters that are stable and realizable.

Principal-phase function A piecewise function with jumps of 2π due to periodicity,

denoted by $\angle H(\omega)$, and is a result of the
MATLAB `angle` function. Also known as a
wrapped-phase function.

Steady-state response A response of a stable
LTI system that continues or persists as
$n \to \infty$. It is either a constant or sinusoidal
in nature.

System gain The magnitude, $|H(\omega)|$, of the the
frequency response function is called the
system gain. Also known as the magnitude
response of the system.

Transient response A response of an LTI
system that decays to zero as $n \to \infty$.

Unwrapped-phase function A phase function
that varies continuously without any jumps
of 2π due to periodicity, denoted by $\Psi(\omega)$.
Also known as a continuous-phase
function.

Wrapped-phase function A piecewise
function with jumps of 2π due to periodicity,
denoted by $\angle H(\omega)$, and is a result of the
MATLAB `angle` function. Also known as a
principal-phase function.

Zero-state response A response of an LTI
system due to an applied input when no
initial conditions are present.

MATLAB functions and scripts

Name	Description	Page
abs	Computes the magnitude of frequency response	226
angle	Computes the principal value of phase response	208, 226
contphase	Computes the continuous phase from group delay	228
fft	Computes equidistant values of the DTFT	226
freqs	Computes continuous-time frequency response	267
freqz	Computes the frequency response function	226
freqz0*	Computes the frequency response function	227
grpdelay	Computes the group-delay response	228
grpdelay0*	Computes the group-delay response	228
fvtool	Filter analysis and visualization tool	229
phasez	Computes phase response in radians	229
phasedelay	Computes phase delay in "samples"	229
polystab	Converts polynomial to minimum phase	289
splane	Plots poles and zeros of a rational $H(s)$	267

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

- A detailed treatment of continuous-time and discrete-time Fourier series and transforms at the same level as in this book is given in Oppenheim *et al.* (1997) and Lathi (2005).
- The standard references for Fourier transforms from an electrical engineering perspective are Bracewell (2000) and Papoulis (1962).
- A mathematical treatment of Fourier series and transforms is given in Walker (1988) and Kammler (2000).

Review questions

1. What are the eigenfunctions of LTI systems and how are their responses computed?
2. Describe the response of real LTI systems to sinusoidal signals in terms of important system characteristics.
3. What is the difference between continuous and principal phase functions? Between unwrapped and wrapped phase functions?
4. For a stable FIR system, the transient response vanishes after a finite number of samples. Do you agree or disagree? Explain.
5. Describe a simple and practical method to check whether a system is linear and time-invariant.
6. Response of an LTI system to a periodic excitation is also periodic with the same period. Explain why?
7. What allows us, for all practical purposes, to compute response of a stable system to a periodic input either in the time-domain or in the frequency-domain?
8. Define a distortionless LTI system in the time-domain and in the frequency-domain? What are the corresponding parameters known as?
9. What is a phase-delay response and what should it be for a distortionless system?
10. What is a group-delay response and why is it called a group delay? What should it be for a distortionless system?
11. Describe the four ideal frequency-selective filters using frequency response functions.
12. Explain in clear terms why an ideal filter is unstable and practically unrealizable.
13. Define the impulse responses of ideal highpass, bandpass, and bandstop ideal filters in terms of the impulse response of the ideal lowpass filter.
14. How does a practical frequency selective filter differ from the corresponding ideal filter? Describe the needed parameters.
15. Provide a geometrical interpretation of the magnitude response in terms of a pole-zero description of the LTI system.
16. Provide a geometrical interpretation of the phase response in terms of a pole-zero description of the LTI system.
17. What is the significance of a pole and the significance of a zero in determination of the gain response?
18. How would you convert an impulse response representation of an LTI system into a difference equation representation? What is the necessary condition for this conversion?
19. Define a digital resonator in terms of pole-zero placement.
20. A discrete-time sinusoidal oscillator is not a stable system. Can we still implement and use it? How and why?
21. What are comb filters and how are they designed? Why are they called comb filters?

22. Given a rational magnitude-squared response, can we uniquely determine the impulse response? If not why not? If not uniquely, how many different impulse responses are possible?

23. It is argued in the chapter that the magnitude and phase responses cannot be specified independently. Explain this argument.

24. What is the requirement on the poles and zeros for the magnitude-squared response?

25. A parallel connection of two allpass systems is an allpass system. Do you agree or disagree? Explain.

26. Define a dispersive allpass system in terms of its pole-zero placement and in terms of its rational system function.

27. What is a minimum-phase system? A maximum-phase system? A mixed-phase system? An invertible system?

28. Can any system with a rational system function be decomposed into a product of an allpass system and a minimum-phase system? If yes, then explain how?

Problems

Tutorial problems

1. Consider the first-order LTI system discussed in Example 5.1.
   (a) Write an analytical expression for the output sequence $y[n]$ if the input is (i) $x[n] = 3\cos(\pi n/2)$, and (ii) $x[n] = 3\sin(\pi n/4)$, for $a = 0.5$.
   (b) Write a MATLAB script that computes and plots the magnitude and phase response for $0 \le \omega \le \pi$ at increments of $\pi/8$, the input $x[n]$, and the output $y[n]$. Assume that $a = 0.8$. Hint: use the `freqz` function to compute $H(e^{j\omega})$ and a do loop that includes a `pause` statement for the different values of input frequency.
   (c) Repeat (b) for $a = -0.8$ and compare the frequency responses and the outputs of the two filters.

2. An LTI system is described by the difference equation

   $$y[n] = bx[n] - 0.81y[n-2].$$

   (a) Determine the frequency response $H(e^{j\omega})$ of the system in terms of b.
   (b) Determine b so that $|H(e^{j\omega})|_{\max} = 1$. Plot the resulting magnitude response.
   (c) Graph the wrapped and the unwrapped phase responses in one plot.
   (d) Determine analytically the response $y[n]$ to the input $x[n] = 2\cos(0.5\pi n + 60°)$.
   (e) Using MATLAB compute the steady-state response to $x[n]$ above and verify your result.

3. Consider the first-order system described by $y[n] = 0.8y[n-1] + 0.2x[n]$. It is excited by the linear FM signal (see Example 5.2) given by

   $$x[n] = \cos\left\{\pi(B/F_s/N)n^2\right\}, \ 0 \le n \le N$$

where $B = 10$ Hz, $F_s = 100$ Hz, and $\tau = N/F_s = 10$ sec.

(a) Determine and plot $|H(e^{j2\pi F})|$ over $0 \leq F \leq B$ Hz.

(b) Plot $x[n] = x(nT)$ over $0 \leq t \leq \tau$ sec.

(c) Process the signal $x[n]$ through the system using MATLAB to obtain $y[n] = y(nT)$ and plot it over $0 \leq t \leq \tau$ sec. Verify that the amplitude of $x[n]$ is attenuated according to the frequency response $H(e^{j2\pi F})$.

4. For the following input–output pairs determine whether or not there is an LTI system producing $y[n]$ when the input is $x[n]$. If such a system exists, determine if it is unique and its frequency response function; otherwise, explain why such a system is not possible:

(a) $x[n] = (1/2)^n u[n] \overset{\mathcal{H}}{\longmapsto} y[n] = (1/3)^n u[n]$,

(b) $x[n] = e^{j\pi n/3} \overset{\mathcal{H}}{\longmapsto} y[n] = 2e^{j\pi n/3}$ all n,

(c) $x[n] = \frac{\sin \pi n/4}{\pi n} \overset{\mathcal{H}}{\longmapsto} y[n] = \frac{\sin \pi n/2}{\pi n}$,

(d) $x[n] = u[n] \overset{\mathcal{H}}{\longmapsto} y[n] = \delta[n]$.

5. A discrete-time system is implemented by MATLAB function `y=ltiwhich(x)`.

(a) Determine whether the system is linear and time-invariant. Hint: You can check the response to the input $x[n] = (-1)^n u[n]$. Why?

(b) If the system is LTI determine its impulse response $h[n]$ and its frequency response $H(e^{j\omega})$ using MATLAB.

(c) Estimate $|H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ using (5.14) for $\omega = k\pi/10$, $0 \leq k \leq 10$ and compare with the results obtained in (b).

6. Compute and plot the phase response of $H(z) = [(1 + z^{-1})/2]^6$.

(a) Determine analytically $\angle H(e^{j\omega})$ and use the formula obtained to compute and plot the phase response.

(b) Compute and plot the phase response using the function `freqz`.

7. Determine the system function, magnitude response, and phase response of the following systems and use the pole-zero pattern to explain the shape of their magnitude response:

(a) $y[n] = \frac{1}{2}(x[n] - x[n-1])$,

(b) $y[n] = \frac{1}{2}(x[n] - x[n-2])$,

(c) $y[n] = \frac{1}{4}(x[n] + x[n-1]) - \frac{1}{4}(x[n-2] + x[n-3])$,

(d) $y[n] = \frac{1}{4}(x[n] + x[n-1]) - \frac{1}{4}(x[n-3] + x[n-4])$.

8. Derive formula (5.72) for the impulse response of an ideal bandpass filter by:

(a) using the impulse response (5.70) of the ideal lowpass filter and the modulation property of DTFT,

(b) expressing the frequency response of the ideal bandpass filter as the difference between the frequency responses of two ideal lowpass filters.

9. Compute the group delay of the system (5.99) using the following functions and compare the results obtained.

(a) The MATLAB function `grpdelay`.

(b) The function `grpdelay0` given in Figure 5.13.

(c) A function `[grp,omega]=mygrpdelay(b,a)` designed to implement equation (5.89).

10. Show that the group delay of an LTI system with frequency response function $H(e^{j\omega}) = H_R(\omega) + jH_I(\omega)$ can be expressed as

$$\tau(\omega) = \frac{H_R(\omega)G_R(\omega) + H_I(\omega)G_I(\omega)}{|H(e^{j\omega})|^2},$$

where $G(e^{j\omega}) = G_R(\omega) + jG_I(\omega)$ is the DTFT of $nh[n]$.

11. Write a MATLAB script that generates Figures 5.7 and 5.8 in Example 5.6 using the following approaches and compare the results.
    (a) Treat the system $H(z)$ as a cascade connection of K second-order systems $H_k(z) = b_0^{1/K}/A_k(z)$, where $A_k(z) = 1 - 2r\cos\omega_0 z^{-1} + r^2 z^{-2}$.
    (b) Treat the system $H(z) = H_1(z)\ldots H_K(z)$ as a single $2K$th-order system. Hint: Determine the coefficients of $A(z) = A_1(z)\ldots A_K(z)$ using convolution.

12. Using equations (5.62) and (5.63), prove (5.64).

13. Compute and plot the phase response using the functions `freqz`, `angle`, `phasez`, `unwrap`, and `phasedelay` for the following systems:
    (a) The pure delay $y[n] = x[n - 15]$.
    (b) The system defined by (5.99).

14. Consider the digital resonator with two zeros given in (5.121) and repeated below:

$$H(z) = b_0 \frac{1 - z^{-2}}{1 - 2r\cos(\phi)z^{-1} + r^2 z^{-2}}.$$

    (a) Determine the constant b_0 for the normalization condition $|H(e^{j\phi})| = 1$. Compute the value of b_0 for $r = 0.9$ and $\phi = \pi/3$.
    (b) Determine the impulse response $h[n]$ of the above resonator and plot it for $r = 0.9$ and $\phi = \pi/3$.
    (c) Determine the approximate 3-dB bandwidth of the resonator.
    (d) For $r = 0.9$ and $\phi = \pi/3$, plot the magnitude response in dB, phase response and group-delay. Determine the exact 3-dB bandwidth and compare it with your answer in (c) above.
    (e) Compare your frequency response plots in (d) with those in Figure 5.23 and comment on the effectiveness of two zeros.

15. Consider the notch filter given in (5.124).
    (a) Determine the magnitude of its frequency response $H(e^{j\omega})$ for $\omega \approx \phi$ and determine b_0 to normalize this response.
    (b) Compute and plot $|H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ for $r = 0.9$ and $\phi = 2\pi/5$.
    (c) Determine an analytic expression for the 3-dB bandwidth of the above notch filter and verify it using your plot in (b) above.

16. Use the modulation property (5.144) to prove the lowpass to highpass transformation (5.145). Illustrate the application of this technique by computing and plotting the magnitude response of the following lowpass and the highpass filters derived from them.
    (a) The moving average filter $y[n] = \frac{1}{M}\sum_{k=0}^{M-1} x[n - k]$.
    (b) The lowpass filter defined by (5.99).

17. Show that the magnitude, phase, and group delay of the allpass filter defined by (5.157) are given by formulas (5.160)–(5.162) and use the proper MATLAB functions to generate Figure 5.29.

18. Consider a causal system given by the system function

$$H(z) = \frac{2 + 3.125z^{-2}}{1 - 0.9z^{-1} + 0.81z^{-2}}.$$

(a) Compute and plot magnitude and phase responses of the system.

(b) Determine the minimum-phase system $H_{min}(z)$ corresponding to $H(z)$ and plot its magnitude and phase responses.

(c) Determine the equalizer system $H_{eq}(z)$ corresponding to $H(z)$ and choose gain G so that the overall system $|H(z)H_{eq}(z)| = 1$. Plot its magnitude and phase responses.

19. Consider the first-order, real, causal, and stable allpass system function given by

$$H(z) = \frac{z^{-1} - a}{1 - az^{-1}}. \quad -1 < a < 1$$

(a) Show that

$$|H(z)| \begin{cases} <1 & \text{for } |z| > 1, \\ =1 & \text{for } |z| = 1, \\ > 1 & \text{for } |z| < 1. \end{cases}$$

(b) Let $\tau(\omega)$ denote the group-delay function of the above allpass system. Show that $\int_0^\pi \tau(\omega)d\omega = \pi$.

20. Consider a comb filter generated by an infinite number of echoes spaced D samples apart with exponentially decaying amplitudes using a transfer function of the form

$$H(z) = \frac{z^{-D}}{1 - az^{-D}}. \quad -1 < a < 1$$

(a) Determine the impulse response $h[n]$ as a function of D and a and verify it using MATLAB for $D = 4$ and $a = 0.8$.

(b) Determine the magnitude response of the IIR filter and show that it exhibits D peaks and D dips over $0 \le \omega < 2\pi$. Determine the values and locations of these peaks and dips. Verify your results by plotting magnitude response for $D = 4$ and $a = 0.8$.

(c) Plot impulse and magnitude responses for $D = 5$, $a = 0.9$ and $D = 8$, $a = -0.8$.

21. Consider the following continuous-time system

$$H(s) = \frac{s^4 - 6s^3 + 10s^2 + 2s - 15}{s^5 + 15s^4 + 100s^3 + 370s^2 + 744s + 720}.$$

(a) Show that the system $H(s)$ is a nonminimum phase system.

(b) Decompose $H(s)$ into the product of minimum phase component $H_{\min}(s)$ and an all pass component $H_{ap}(s)$.

(c) Plot the magnitude and phase responses of $H(s)$ and $H_{\min}(s)$ in one figure and explain your plots.

(d) Plot the magnitude and phase responses of $H_{ap}(s)$.

22. Consider the allpass reverberator system given in (5.166)

$$H(z) = \frac{z^{-D} - a}{1 - az^{-D}}. \quad -1 < a < 1$$

(a) Determine and plot $h[n]$ for $D = 5$ and $a = 0.7$.

(b) Determine and plot magnitude, wrapped-phase, unwrapped-phase, and group-delay responses for $D = 5$ and $a = 0.7$.

(c) Determine and plot magnitude, wrapped-phase, unwrapped-phase, and group-delay responses for $D = 5$ and $a = -0.7$.

(d) Comment on your results in (b) and (c) above.

Basic problems

23. An LTI system is described by the difference equation

$$y[n] = bx[n] + 0.8y[n - 1] - 0.81y[n - 2].$$

(a) Determine the frequency response $H(e^{j\omega})$ of the system in terms of b.

(b) Determine b so that $|H(e^{j\omega})|_{\max} = 1$. Plot the resulting magnitude response.

(c) Graph the wrapped and the unwrapped-phase responses in one plot.

(d) Determine analytically the response $y[n]$ to the input $x[n] = 2\cos(\pi n/3 + 45°)$.

(e) Using MATLAB compute the steady-state response to $x[n]$ above.

24. Consider a second-order LTI system

$$y[n] = bx[n] - ay[n - 2]. \quad 0 < a < 1$$

(a) Determine analytical expressions for the magnitude and phase responses in terms of a and b. Choose b so that the maximum magnitude response is equal to 1. Plot magnitude and phase responses for $a = 0.8$.

(b) Write an analytical expression for the output sequence $y[n]$ if the input is (i) $x[n] = 3\cos(\pi n/2)$, and (ii) $x[n] = 3\sin(\pi n/4)$, for $a = 0.8$.

(c) Write a MATLAB script that computes and plots the input $x[n]$, the output $y[n]$, and frequency response for $0 \leq \omega \leq \pi$ at increments of $\pi/8$. Assume that $a = 0.8$.

25. A causal and stable LTI system is described by the impulse response

$$h[n] = 2(0.8)^n \cos(0.25\pi n + \pi/6)u[n] + 5(-0.9)^n u[n].$$

(a) Determine and plot the frequency response $H(e^{j\omega})$ of the system.

(b) Graph the wrapped and the unwrapped phase responses in one plot.

(c) Determine analytically the response $y[n]$ to the input $x[n] = 1 + 3\cos(\pi n/4 + 30°) + 5e^{-j\pi n}$.

(d) Using MATLAB compute the steady-state response to $x[n]$ above.

26. Signal $x[n]$ is periodic with fundamental period $N = 10$. It is given by $x[n] = (0.8)^n$ over its primary interval $0 \le n < 10$. It is applied as an input to a causal LTI system with system function

$$H(z) = \frac{1 - z^{-2}}{1 - 1.5588z^{-1} + 0.81z^{-2}}.$$

(a) Determine the DTFS $c_k^{(x)}$, $0 \le k < 10$ of $x[n]$.
(b) Determine the DTFS $c_k^{(y)}$, $0 \le k < 10$ of $y[n]$.
(c) Compute the steady-state response $y_{ss}[n]$.
(d) Using MATLAB compute and plot signals $x[n]$, $y[n]$, and $y_{ss}[n]$ over $0 \le n \le 50$ and verify your results in part (c) above.

27. For the following input–output pairs determine whether or not there is an LTI system producing $y[n]$ when the input is $x[n]$. If such a system exists, determine if it is unique and its frequency response function; otherwise, explain why such a system is not possible.

(a) $x[n] = (0.25^n + 0.2^n)u[n] \overset{\mathcal{H}}{\longmapsto} y[n] = 0.1^n u[n]$.
(b) $x[n] = \cos(0.2\pi n) \overset{\mathcal{H}}{\longmapsto} y[n] = \cos(0.1\pi n - \pi/3)$ all n.
(c) $x[n] = \sin(\pi n/3) \overset{\mathcal{H}}{\longmapsto} y[n] = 5\cos(\pi n/3 + \pi/4)$.
(d) $x[n] = nu[n] \overset{\mathcal{H}}{\longmapsto} y[n] = u[n]$.

28. Compute and plot the phase response of $H(z) = (1 - 2z^{-1} + 4z^{-2} - 2z^{-3} + z^{-4})/2$.

(a) Determine analytically $\angle H(e^{j\omega})$ and use the formula obtained to compute and plot the phase response.
(b) Compute and plot the phase response using the function `freqz`.

29. Compute and plot the phase response of the system given by $h[n] = \{1, -2, 3, -4, 0, 4, -3, 2, -1\}$.
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad\uparrow$

(a) Determine analytically $\angle H(e^{j\omega})$ and use the formula to plot the phase response.
(b) Compute and plot the phase response using the function `freqz`.

30. Consider a periodic signal

$$x[n] = \sin(0.1\pi n) + \tfrac{1}{3}\sin(0.3\pi n) + \tfrac{1}{5}\sin(0.5\pi n).$$

For each of the following systems, determine if the system imparts (i) no distortion, (ii) magnitude distortion, and/or (iii) phase (or delay) distortion. In each case, graph the input and the steady state response for $0 \le n \le 60$.

(a) $h[n] = \{1, -2, 3, -4, 0, 4, -3, 2, -1\}$.
$\qquad\qquad\quad\uparrow$
(b) $y[n] = 10x[n - 10]$.
(c) $H(z) = \frac{1}{9}(1 + 2z^{-1} + 3z^{-2} + 2z^{-3} + z^{-4})$.
(d) $h[n] = \{1, -1.1756, 1\}$.
$\qquad\quad\uparrow$
(e) $H(z) = \dfrac{1 + 1.778z^{-2} + 3.1605z^{-4}}{1 + 0.5625z^{-2} + 0.3164z^{-4}}$.

31. A multiband ideal bandpass filter is given by

$$H(e^{j\omega}) = \begin{cases} e^{-j\omega n_d}, & \frac{\pi}{8} < |\omega| < \frac{2\pi}{8} \\ 0.5e^{-j\omega n_d}, & \frac{5\pi}{8} < |\omega| < \frac{7\pi}{8} \\ 0. & \text{otherwise} \end{cases}$$

(a) Determine the impulse response of the filter.

(b) Graph the impulse response for $n_d = 0$ for $-100 \le n \le 100$.

(c) From the above truncated impulse response, compute and plot the magnitude response of the filter using MATLAB and compare it with the ideal filter response.

32. Use a figure similar to Figure 5.19 to show geometrically that a pair of complex conjugate zeros has even magnitude response and odd phase response.

33. Compute the magnitude and phase responses of the system

$$H(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{1 + 0.9z^{-1} + 0.81z^{-2} + 0.927z^{-3}}$$

using the following functions and compare the results obtained.

(a) The MATLAB function `freqz`.

(b) The function `freqz0` given in Figure 5.12.

(c) A function `[mag,pha,omega]=myfreqz(b,a)` that implements (5.87) and (5.88).

34. Compute the group delay of the system

$$H(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{1 + 0.9z^{-1} + 0.81z^{-2} + 0.927z^{-3}}$$

using the following functions and compare the results obtained.

(a) The MATLAB function `grpdelay`.

(b) The function `grpdelay0` given in Figure 5.13.

(c) A function `[grp,omega]=mygrpdelay(b,a)` that implements equation (5.89).

35. Find the group delay of the following systems, where α is a real number:

(a) $y[n] = x[n] - \alpha x[n-1]$,

(b) $y[n] = \alpha y[n-1] + x[n]$,

(c) $y[n] = 2\alpha \cos\phi y[n-1] - \alpha^2 y[n-2] + x[n]$.

36. Determine the system function, magnitude response, and phase response of the following systems and use the pole-zero pattern to explain the shape of their magnitude response:

(a) $y[n] = \frac{1}{2}(x[n] + x[n-1])$,

(b) $y[n] = \frac{1}{2}(x[n] + x[n-2])$,

(c) $y[n] = \frac{1}{4}(x[n] - x[n-1]) + \frac{1}{4}(x[n-2] - x[n-3])$,

(d) $y[n] = \frac{1}{4}(x[n] - x[n-1]) + \frac{1}{4}(x[n-3] - x[n-4])$.

37. Compute and plot the phase response using the functions `freqz`, `angle`, `phasez`, `unwrap`, and `phasedelay` for the following systems:

(a) The pure delay $y[n] = x[n-15]$,

(b) The system defined by

$$H(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{1 + 0.9z^{-1} + 0.81z^{-2} + 0.927z^{-3}}.$$

38. We want to design a second-order IIR filter using pole-zero placement that satisfies the following requirements: (1) the magnitude response is 0 at $\omega_1 = 0$ and $\omega_3 = \pi$; (2) The maximum magnitude is 1 at $\omega_{2,4} = \pm\pi/4$; and (3) the magnitude response is approximately $1/\sqrt{2}$ at frequencies $\omega_{2,4} \pm 0.05$.

    (a) Determine locations of two poles and two zeros of the required filter and then compute its system function $H(z)$.

    (b) Graph the magnitude response of the filter and verify the given requirements.

    (c) Graph phase and group-delay responses in one plot.

39. Consider the FIR notch filter given in (5.124) and repeated below

    $$H(z) = b_0[1 - (2r\cos\phi)z^{-1} + r^2 z^{-2}].$$

    Let $r = 0.95$ and $\phi = 2\pi/5$.

    (a) Determine b_0 so that $|H(e^{j\omega})|_{max} = 1$. Using this value of b_0 plot $|H(e^{j\omega})|$ in dB.

    (b) Repeat part (a) using $r = 1$. Comment on your results.

    (c) Now consider a cascade of three FIR notch filters, $H(z) = b_0 \prod_{k=-1}^{1} H_k(z)$ of the form

    $$H_k(z) = [1 - (2\cos\phi_k)z^{-1} + z^{-2}],$$

    where $\phi_k = (1 + 0.05k)2\pi/5$, $k = -1, 0, 1$. Choose b_0 so that $|H(e^{j\omega})|_{max} = 1$ and plot $|H(e^{j\omega})|$ in dB. Comment on your plot in terms of stopband width.

    (d) Repeat (c) for a cascade of five FIR notch filters using $k = 0, \pm1, \pm2$. Comment on your plot in terms of stopband width.

40. Consider a second-order IIR notch filter specification that satisfies the following requirements: (1) the magnitude response has notches at $\omega_{1,2} = \pm 2\pi/3$; (2) The maximum magnitude response is 1; (3) the magnitude response is approximately $1/\sqrt{2}$ at frequencies $\omega_{1,2} \pm 0.01$.

    (a) Using the pole-zero placement approach determine locations of two poles and two zeros of the required filter and then compute its system function $H(z)$.

    (b) Graph the magnitude response of the filter and verify the given requirements.

    (c) Graph phase and group-delay responses in one plot.

41. Let $H_{ap}(z)$ be a causal and stable allpass system. Let $x[n]$ be a causal input and $y[n]$ be its response. Show that for any time $n_0 > 0$,

    $$\sum_{n=0}^{n_0} |y[n]|^2 \le \sum_{n=0}^{n_0} |x[n]|^2.$$

42. Consider a causal and stable system given by the system function

    $$H(z) = \frac{1 + 5.6569z^{-1} + 16z^{-2}}{1 - 0.8z^{-1} + 0.64z^{-2}}.$$

    (a) Express $H(z)$ as a decomposition of a minimum-phase and an allpass system.

    (b) Graph the magnitudes of $H(z)$ and its minimum-phase and allpass components in one plot and comment on your observation.

    (c) Graph the group-delays of $H(z)$ and its minimum-phase and allpass components in one plot and comment on your observation.

43. A continuous-time LTI system is described by the differential equation

$$y''(t) + 2y'(t) + 101y(t) = 10x'(t).$$

The input to the system is $x(t) = 5 - 4\cos(10t - 2\pi/3) + 3\sin(20t) + 2e^{-j100t}$.

(a) Using the fact that if $x(t) = e^{st}$ then the output is $y(t) = H(s)e^{st}$, determine the system function $H(s)$ from the differential equation.

(b) Determine the output $y(t)$.

44. The magnitude-squared response of a continuous-time LTI system is given by

$$|H(j\Omega)|^2 = \frac{\Omega^6 + 64}{2\Omega^4 - \Omega^6 - \Omega^2 - 100}.$$

(a) Determine the minimum-phase and maximum-phase system components and graph the magnitude response.

(b) Graph the phase responses of the minimum- and maximum-phase components in one plot and comment on your observations.

Assessment problems

45. Consider a second-order LTI system

$$y[n] = bx[n] - bx[n-2] - ay[n-2], \quad 0 < a < 1.$$

(a) Determine analytical expressions for the magnitude and phase responses in terms of a and b. Choose b so that the maximum magnitude response is equal to 1. Plot magnitude and phase responses for $a = 0.9$.

(b) Write an analytical expression for the output sequence $y[n]$ if the input is (i) $x[n] = 3\cos(\pi n/2)$, and (ii) $x[n] = 3\sin(\pi n/4)$, for $a = 0.9$.

(c) Write a MATLAB script that computes and plots the input $x[n]$, the output $y[n]$, and frequency response for $0 \le \omega \le \pi$ at increments of $\pi/8$. Assume that $a = 0.9$.

46. An LTI system is described by the difference equation $y[n] = x[n] - x[n-4] - 0.81y[n-2] - 0.6561y[n-4]$.

(a) Determine the frequency response $H(e^{j\omega})$ of the system.

(b) Plot the magnitude and the dB-gain responses.

(c) Graph the wrapped and the unwrapped phase responses in one plot.

(d) Determine analytically the response $y[n]$ to the input $x[n] = 2\cos(0.5\pi n + 60°) + \sin(\pi n/3 - 45°)$.

(e) Using MATLAB compute the steady-state response to $x[n]$ above and verify your result.

47. Let $x[n]$ be periodic with fundamental period $N = 5$. It is given by $x[n] = (-0.5)^n$ over its primary interval $0 \le n < 5$. It is applied as an input to a causal LTI system with system function

$$H(z) = \frac{1 - z^{-1}}{1 + 1.5588z^{-1} + 0.81z^{-2}}.$$

Problems

(a) Determine the DTFS $c_k^{(x)}$, $0 \le k < 5$ of $x[n]$.

(b) Determine the DTFS $c_k^{(y)}$, $0 \le k < 5$ of $y[n]$.

(c) Compute the stead-state response $y_{ss}[n]$.

(d) Using MATLAB compute and plot signals $x[n]$, $y[n]$, and $y_{ss}[n]$ over $0 \le n \le 30$ and verify your results in part (c) above.

48. Consider a periodic signal with period $N = 8$ given by

$$x[n] = \left\{ \underset{\uparrow}{1}, 2, 3, 4, 3, 2, 1, 0 \right\}_{\text{periodic}}.$$

For each of the following systems, determine if the system imparts (i) no distortion, (ii) magnitude distortion, and/or (iii) phase (or delay) distortion. In each case, graph the input and the steady state response for $0 \le n \le 60$.

(a) $h[n] = \left\{ \underset{\uparrow}{2}, -1, 1, 3, 6, 3, 1, -1, 2 \right\}$.

(b) $H(z) = \frac{1}{9}\left(1 - 0.5z^{-1} + 2z^{-2} + 0.5z^{-3} - z^{-4}\right)$.

(c) $H(e^{j\omega}) = 5e^{j\pi/4}$.

(d) $h[n] = \left\{ \underset{\uparrow}{1}, 0.5, 0.25, 0.125, 0.0625 \right\}$.

(e) $H(z) = \dfrac{1 + 1.7928z^{-2} + 1.2277z^{-4}}{1 + 1.4603z^{-2} + 0.8145z^{-4}}$.

49. Compute and plot the phase response of $H(z) = 5 - 4z^{-1} + 3z^{-2} - 3z^{-3} + 4z^{-4} - 5z^{-5}$.

(a) Determine analytically $\angle H(e^{j\omega})$ and use the formula obtained to plot the phase response.

(b) Compute and plot the phase response using the function `freqz`.

50. Compute the magnitude and phase responses of the system

$$H(z) = \frac{1 - 2.73z^{-1} + 3.73z^{-2} - 2.73z^{-3} + z^{-4}}{1 - 2.46z^{-1} + 3.02z^{-2} - 1.99z^{-3} + 0.66z^{-4}}$$

using the following functions and compare the results obtained.

(a) The MATLAB function `freqz`.

(b) The function `freqz0` given in Figure 5.12.

(c) A function `[mag,pha,omega]=myfreqz(b,a)` that implements equation (5.87).

51. Compute the group delay of the system

$$H(z) = \frac{1 - 2.73z^{-1} + 3.73z^{-2} - 2.73z^{-3} + z^{-4}}{1 - 2.46z^{-1} + 3.02z^{-2} - 1.99z^{-3} + 0.66z^{-4}}$$

using the following functions and compare the results obtained.

(a) The MATLAB function `grpdelay`.

(b) The function `grpdelay0` given in Figure 5.13.

(c) A function `[grp,omega]=mygrpdelay(b,a)` that implements equation (5.89).

52. For the following input–output pairs determine whether or not there is an LTI system producing $y[n]$ when the input is $x[n]$. If such a system exists, determine if it is unique and its frequency response function; otherwise, explain why such a system is not possible.

(a) $x[n] = \left(\frac{1}{4}\right)^n$ all $n \xmapsto{\mathcal{H}} y[n] = \left(\frac{1}{2}\right)^n$ all n.

(b) $x[n] = \cos\left(\frac{\pi n}{4}\right) \xmapsto{\mathcal{H}} y[n] = 2\cos\left(\frac{\pi n}{4} - \frac{\pi}{4}\right)$.

(c) $x[n] = e^{j\frac{2\pi n}{5}}u[n] \xmapsto{\mathcal{H}} y[n] = 3e^{j\frac{2\pi n}{5} - \frac{j\pi}{2}}u[n]$.

(d) $x[n] = 2u[n] \xmapsto{\mathcal{H}} y[n] = 3u[n] - 5u[n-5]$.

53. Consider the system function $H(z) = \frac{1}{8}(1 + z^{-2})^8$.

   (a) Compute and plot the phase response of the system.

   (b) Determine analytically $\angle H(e^{j\omega})$ and use the formula obtained to compute and plot the phase response.

   (c) Compute and plot the phase response using the function `freqz`.

54. A multiband ideal bandstop filter is given by

$$H(e^{j\omega}) = \begin{cases} e^{-j\omega n_{\mathrm{d}}}, & 0 < |\omega| < \frac{\pi}{8} \\ \frac{2}{3}e^{-j\omega n_{\mathrm{d}}}, & \frac{3\pi}{8} < |\omega| < \frac{5\pi}{8} \\ \frac{1}{3}e^{-j\omega n_{\mathrm{d}}}, & \frac{7\pi}{8} < |\omega| < \pi \\ 0. & \text{otherwise} \end{cases}$$

   (a) Determine the impulse response of the filter.

   (b) Graph the impulse response for $n_{\mathrm{d}} = 0$ for $-200 \le n \le 200$.

   (c) From the above truncated impulse response, compute and plot the magnitude response of the filter using MATLAB and compare it with the ideal filter response.

55. Determine the system function, magnitude response, and phase response of the following systems and use the pole-zero pattern to explain the shape of their magnitude response:

   (a) $y[n] = x[n] - x[n-2] - 0.81y[n-2]$,

   (b) $y[n] = x[n] - x[n-4] + 0.6561y[n-4]$,

   (c) $y[n] = x[n] - x[n-4] - 0.6561y[n-4]$,

   (d) $y[n] = x[n] - x[n-1] + 0.99y[n-1] - 0.9801y[n-2]$.

56. Derive a formula similar to (5.72) for the impulse response of an ideal bandstop filter by:

   (a) using the impulse response (5.70) of the ideal lowpass filter and the modulation property of DTFT,

   (b) expressing the frequency response of the ideal bandstop filter as the difference between the frequency responses of two ideal lowpass filters.

57. Determine the group delay of the following systems:

   (a) $y[n] = x[n] - 0.9x[n-1]$,

   (b) $y[n] = 0.8y[n-1] + x[n]$,

   (c) $y[n] = 0.7y[n-1] - 0.49y[n-2] + x[n]$.

58. Modify the function `grpdelay0` to compute the group delay directly from the coefficients $\{a_k, b_k\}$ of $H(z) = B(z)/A(z)$ using the following steps:

   (a) Define an FIR filter with z-transform $C(z) = B(z)A^*(1/z^*)$.

   (b) Show that $\angle H(\omega) = \angle C(\omega) + N\omega$.

   (c) Use results for the FIR case and part (b) to write a function `mygrpdelay` that computes the group delay from $\{a_k, b_k\}$.

(d) Compute the group delay of the system (5.99) using `mygrpdelay` and `grpdelay` and compare the results.

59. Compute and plot the phase response using the functions `freqz`, `angle`, `phasez`, `unwrap`, and `phasedelay` for the following systems:
    (a) The system defined by $y[n] = x[n] - 2x[n-1] + 3.2x[n-2] - 2x[n-3] + x[n-4]$.
    (b) The system defined by

    $$H(z) = \frac{1 - 2.4142z^{-1} + 2.4142z^{-2} - z^{-3}}{1 - 1.8z^{-1} + 1.62z^{-2} + 0.729z^{-3}}.$$

60. Determine, compute, and plot the frequency responses of a minimum phase equalizer for the system (5.176) in Example 5.10. Use $G = 1$ and $n_d = 0$.

61. MATLAB provides a function called `polystab` that stabilizes the given polynomial with respect to the unit circle, that is, it reflects those roots which are outside the unit-circle into those that are inside the unit circle but with the same angle. Using this function, convert the following systems into minimum-phase and maximum-phase systems. Verify your answers using a pole-zero plot for each system.
    (a) $y[n] = x[n] - 1.5x[n-1] - x[n-2]$,
    (b) $H(z) = \dfrac{z^2 + 2z + 0.75}{z^2 - 0.5z}$,
    (c) $y[n] = x[n] - 2x[n-1] + 3.2x[n-2] - 2x[n-3] + x[n-4]$,
    (d) $H(z) = \dfrac{1 - 2.4142z^{-1} + 2.4142z^{-2} - z^{-3}}{1 - 1.8z^{-1} + 1.62z^{-2} + 0.729z^{-3}}.$

62. Show that if $h(t)$ is real, then the phase response is given by

    $$\angle H(j\Omega) = \frac{1}{2j} \ln \left[\frac{H(j\Omega)}{H(-j\Omega)} \right].$$

63. Consider a single echo system $y[n] = x[n] + 0.1x[n-5]$.
    (a) Determine and plot the impulse, magnitude and phase responses of the system $H(z)$.
    (b) Determine the difference equation of an inverse system $H_i(z)$ so that $H(z)H_i(z) = 1$.
    (c) Determine and plot the impulse, magnitude and phase responses of the inverse system $H_i(z)$.
    (d) Let $x[n] = \sum_{k=1}^{10} 1/k \sin(0.01k^2\pi n)$, $0 \le n \le 50$. Let $y[n]$ be the output of the system $H(z)$ and input to the inverse system $H_i(z)$ and let $v[n]$ be the output of the inverse system. Plot sequences $x[n]$, $y[n]$ and $v[n]$ and comment on your results.

64. Consider the IIR notch filter given in (5.126) and repeated here

    $$G(z) = b_0 \frac{1 - (2\cos\phi)z^{-1} + z^{-2}}{1 - (2r\cos\phi)z^{-1} + r^2 z^{-2}}.$$

    (a) Determine b_0 so that the maximum magnitude response is equal to one.
    (b) For $r = 0.9$ and $\phi = 2\pi/5$ and using the above value of b_0 plot the log-magnitude (dB) response and the phase response.

(c) For $r = 0.9$ and $\phi = 3\pi/5$ and using the above value of b_0 plot the log-magnitude (dB) response and the phase response.

65. Consider a second-order IIR filter specification that satisfies the following requirements: (1) the magnitude response is 0 at $\omega_1 = \pi/2$ and $\omega_4 = 3\pi/2$; (2) the maximum magnitude response is 1 at $\omega_{2,3} = \pm 2\pi/3$; and (3) the magnitude response is approximately $1/\sqrt{2}$ at frequencies $\omega_{2,3} \pm 0.05$.
   (a) Using the pole-zero placement approach determine locations of two poles and two zeros of the required filter and then compute its system function $H(z)$.
   (b) Graph the magnitude response of the filter and verify the given requirements.
   (c) Graph phase and group-delay responses in one plot.

66. Consider a causal and stable system given by the system function

$$H(z) = \frac{1 - 2.1z^{-1} + 2.7z^{-2}}{1 + 0.3126z^{-1} + 0.81z^{-2}}.$$

   (a) Express $H(z)$ as a decomposition of a minimum-phase and an allpass system.
   (b) Graph the magnitudes of $H(z)$ and its minimum-phase and allpass components in one plot and comment on your observation.
   (c) Graph the group-delays of $H(z)$ and its minimum-phase and allpass components in one plot and comment on your observation.

67. A continuous-time LTI system is described by the impulse response

$$h(t) = 4e^{-0.05t} \cos(10\pi t + \pi/4)u(t).$$

   The input to the system is $x(t) = 4 - 3\cos(4\pi t + \pi/3) + 5\sin(20\pi t)$.
   (a) Determine the system function $H(s)$ and graph its magnitude and phase responses.
   (b) Determine the output $y(t)$.

68. Consider a plain reverberator given by $H(z) = \dfrac{1}{1 - az^{-D}}$ and an allpass reverberator given by $H_{ap}(z) = \dfrac{z^{-D} - a}{1 - az^{-D}}$. Let inputs to these reverberators be: (i) $x[n] = \delta[n]$, (ii) $x[n] = \{\underset{\uparrow}{1}, 1, 1, 1, 1\}$, and (iii) $x[n] = \{\underset{\uparrow}{0}, 1, 2, 3, 4, 5, 4, 3, 2, 1\}$. For each of these inputs, determine and graph input and output sequences over $0 \leq n \leq 100$ in one plot for both reverberators. Let $D = 5$.

Review problems

69. When causal filters are used in real-time signal processing, the output signal always lags the input signal. If the signal is recorded for later "off-line" filtering, then one simple approach to eliminate this lag problem is: (i) filter the signal and record its response; (ii) filter the recorded response *backwards* using the same filter; and (iii) store the resulting response backwards. Let the filter be a causal single pole filter

$$H(z) = \frac{1}{1 - az^{-1}}, \quad 0 < a < 1$$

(a) Determine the *effective* frequency response of the above described "off-line" system of filtering signal forward and then backward.

(b) Determine the effective impulse response.

(c) If the effective filter system function should be

$$H(z) = \frac{z^2/0.64}{z^4 - 2.05z^3 + 3.202z^2 - 2.05z + 1},$$

determine the causal filter needed in the "off-line" system.

70. The following filter:

$$H(z) = \frac{(z^2 + 2z - 3)(z^2 - 3z + 5)}{(z^2 + 3.7z + 1.8)(z^2 - 0.4z + 0.35)}$$

is unstable.

(a) Determine a stable system function $G(z)$ such $\left|G(e^{j\omega})\right| = \left|H(e^{j\omega})\right|$.

(b) Determine other possible system functions that have the same magnitude response as $H(z)$.

6 Sampling of continuous-time signals

This chapter is primarily concerned with the conversion of continuous-time signals into discrete-time signals using uniform or periodic sampling. The presented theory of sampling provides the conditions for signal sampling and reconstruction from a sequence of sample values. It turns out that a properly sampled bandlimited signal can be perfectly reconstructed from its samples. In practice, the numerical value of each sample is expressed by a finite number of bits, a process known as quantization. The error introduced by quantizing the sample values, known as quantization noise, is unavoidable. The major implication of sampling theory is that it makes possible the processing of continuous-time signals using discrete-time signal processing techniques.

Study objectives

After studying this chapter you should be able to:

- Determine the spectrum of a discrete-time signal from that of the original continuous-time signal, and understand the conditions that allow perfect reconstruction of a continuous-time signal from its samples.

- Understand how to process continuous-time signals by sampling, followed by discrete-time signal processing, and reconstruction of the resulting continuous-time signal.

- Understand how practical limitations affect the sampling and reconstruction of continuous-time signals.

- Apply the theory of sampling to continuous-time bandpass signals and two-dimensional image signals.

6.1 Ideal periodic sampling of continuous-time signals

In the most common form of sampling, known as *periodic* or *uniform sampling*, a sequence of samples $x[n]$ is obtained from a continuous-time signal $x_c(t)$ by taking values at equally spaced points in time. Periodic sampling is defined by the relation

$$x[n] \triangleq x_c(t)|_{t=nT} = x_c(nT), \quad -\infty < n < \infty \tag{6.1}$$

where T, the fixed time interval between samples, is known as the *sampling period*. The reciprocal of the sampling period, $F_s = 1/T$, is called *sampling frequency* (when expressed in cycles per second or Hz) or *sampling rate* (when expressed in samples per second). The system that implements the operation in (6.1) is known as an *ideal analog-to-digital converter (ADC)* or *ideal sampler* and is depicted in Figure 6.1. By definition, the term ideal sampling means instantaneous sampling, where each sample is measured with infinite accuracy. The main difference between an ideal ADC and a practical ADC is the finite number of bits (typically 12 or 16 bits) used to represent the value of each sample (see Section 1.2.3).

We now ask: are the samples $x[n]$ sufficient to describe uniquely the original continuous-time signal and, if so, how can $x_c(t)$ be reconstructed from $x[n]$? From the example in Figure 6.2 we conclude that an infinite number of signals can generate the same set of samples. In other words, the samples do not tell us anything about the values of the signal between sampling times. The only way to determine these values is by putting some constraints on the behavior of the continuous-time signal. The answer to these questions lies in the frequency domain, that is, in the relation between the spectra of $x_c(t)$ and $x[n]$.

The input to the ideal ADC is a continuous-time signal which is represented mathematically by a function of time, $x_c(t)$, where t is a continuous variable. The signal $x_c(t)$ and its

$$x_c(t) \longrightarrow \boxed{\begin{array}{c} \text{Ideal} \\ \text{ADC} \\ F_s = 1/T \end{array}} \longrightarrow x[n] = x_c(nT)$$

Figure 6.1 Representation of the ideal analog-to-digital converter (ADC) or ideal sampler.
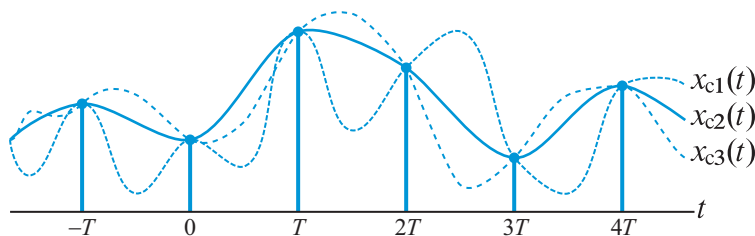
Figure 6.2 Three different continuous-time signals with the same set of sample values, that is, $x[n] = x_{c1}(nT) = x_{c2}(nT) = x_{c3}(nT)$.

spectrum $X_c(j\Omega)$ are uniquely related by the following CTFT pair (see Section 4.2.2):

$$X_c(j\Omega) = \int_{-\infty}^{\infty} x_c(t)e^{-j\Omega t}dt, \tag{6.2}$$

$$x_c(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_c(j\Omega) e^{j\Omega t}d\Omega. \tag{6.3}$$

The discrete-time signal at the output of the ideal ADC is represented mathematically by an indexed sequence $x[n]$ of numbers. The sequence $x[n]$ and its periodic spectrum $X(e^{j\omega})$ are uniquely related by the following DTFT pair (see Section 4.3.2):

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \tag{6.4}$$

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega, \tag{6.5}$$

Since $x[n]$ and $x_c(t)$ are related by (6.1), there should exist an equivalent relationship between $X(e^{j\omega})$ and $X_c(j\Omega)$. Finding this relationship, which is the frequency domain representation of periodic sampling, requires first establishment of a relationship between the frequency variables ω and Ω. To this end, substituting t by $t = nT$ in (6.3) yields

$$x_c(nT) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_c(j\Omega) e^{j\Omega Tn}d\Omega. \tag{6.6}$$

Comparing (6.6) to (6.5) shows that the necessary relationship is (see Section 4.1.2)

$$\omega = \Omega T = 2\pi FT = 2\pi \frac{F}{F_s} = 2\pi f. \tag{6.7}$$

If we replace the variable ω in (6.5) by the equivalent variable ΩT, we have

$$x[n] = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \left[TX(e^{j\Omega T}) \right] e^{j(\Omega T)n}d\Omega. \tag{6.8}$$

Since $x[n] = x_c(nT)$ the right hand sides of (6.6) and (6.8) should be equal. To relate $X(e^{j\Omega T})$ and $X_c(j\Omega)$, we replace Ω by θ and express (6.8) as a sum of integrals over intervals of length $2\pi/T$, which is the interval of integration in (6.5). The result is

$$x_c(nT) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \int_{(2k-1)\pi/T}^{(2k+1)\pi/T} X_c(j\theta) e^{j\theta Tn}d\theta. \tag{6.9}$$

Changing the integration variable from θ to $\Omega = \theta + (2\pi/T)k$ then yields

$$x_c(nT) = \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} \int_{-\pi/T}^{\pi/T} X_c\left(j\Omega - j\frac{2\pi}{T}k \right) e^{j\Omega Tn}e^{-j2\pi kn}d\Omega. \tag{6.10}$$

If we interchange the order of summation and integration and note that $e^{-j2\pi kn} = 1$ for all integer values of k and n, then we obtain

$$x_c(nT) = \frac{1}{2\pi} \int_{-\pi/T}^{\pi/T} \left[\sum_{k=-\infty}^{\infty} X_c\left(j\Omega - j\frac{2\pi}{T}k\right) \right] e^{j\Omega Tn} d\Omega. \tag{6.11}$$

Comparing (6.11) to (6.8) yields the desired relationship between $X(e^{j\Omega T})$ and $X_c(j\Omega)$

$$X(e^{j\Omega T}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(j\Omega - j\frac{2\pi}{T}k\right). \tag{6.12}$$

This equation provides the frequency domain representation of (6.1). Alternatively, we can express (6.12) in terms of the normalized frequency variable ω as

$$X(e^{j\omega}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(j\frac{\omega}{T} - j\frac{2\pi}{T}k\right). \tag{6.13}$$

We note that the spectrum of $x[n]$ is obtained by scaling the spectrum of $x_c(t)$ by $1/T$, putting copies of the scaled spectrum $(1/T)X_c(j\Omega)$ at all integer multiples of the sampling frequency $\Omega_s \triangleq 2\pi/T$ in rad/sam, and then superimposing all scaled and shifted replicas. In practical applications, it is convenient to express (6.12) in terms of the frequency variable F and plot $X(e^{j2\pi FT})$ as a function of F. The resulting formula is

$$X(e^{j2\pi FT}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left[j2\pi(F - kF_s)\right]. \tag{6.14}$$

The spectrum of $x[n]$ can be readily sketched if $x_c(t)$ is assumed to be bandlimited, that is, $X_c(j\Omega) = 0$ for $|\Omega| > \Omega_H \triangleq 2\pi F_H$. Figure 6.3 shows a convenient $X_c(j\Omega)$ and the corresponding $X(e^{j\Omega T})$ for two cases, $\Omega_s > 2\Omega_H$ and $\Omega_s < 2\Omega_H$. Figure 6.3(b) reveals that the sampling operation leaves the input spectrum $X_c(j\Omega)$ *intact* when $\Omega_s > 2\Omega_H$; therefore, it should be possible to recover or reconstruct $x_c(t)$ from the sequence $x[n]$. In contrast, as illustrated in Figure 6.3(c), if $\Omega_s < 2\Omega_H$, the scaled copies of $X_c(j\Omega)$ overlap, so that when they are added together, $X_c(j\Omega)$ cannot be recovered from $X(e^{j\Omega T})$. This effect, in which individual terms in (6.14) overlap is known as *aliasing distortion* or simply *aliasing*.

Two conditions obviously are necessary to prevent overlapping spectral bands: the continuous-time signal must be bandlimited, and the sampling frequency must be sufficiently large so that $\Omega_s - \Omega_H > \Omega_H$. Thus we require

$$X_c(j\Omega) = 0, \quad |\Omega| > \Omega_H \tag{6.15}$$

and

$$\Omega_s \geq 2\Omega_H \quad \text{or} \quad T \leq \frac{1}{2F_H}. \tag{6.16}$$
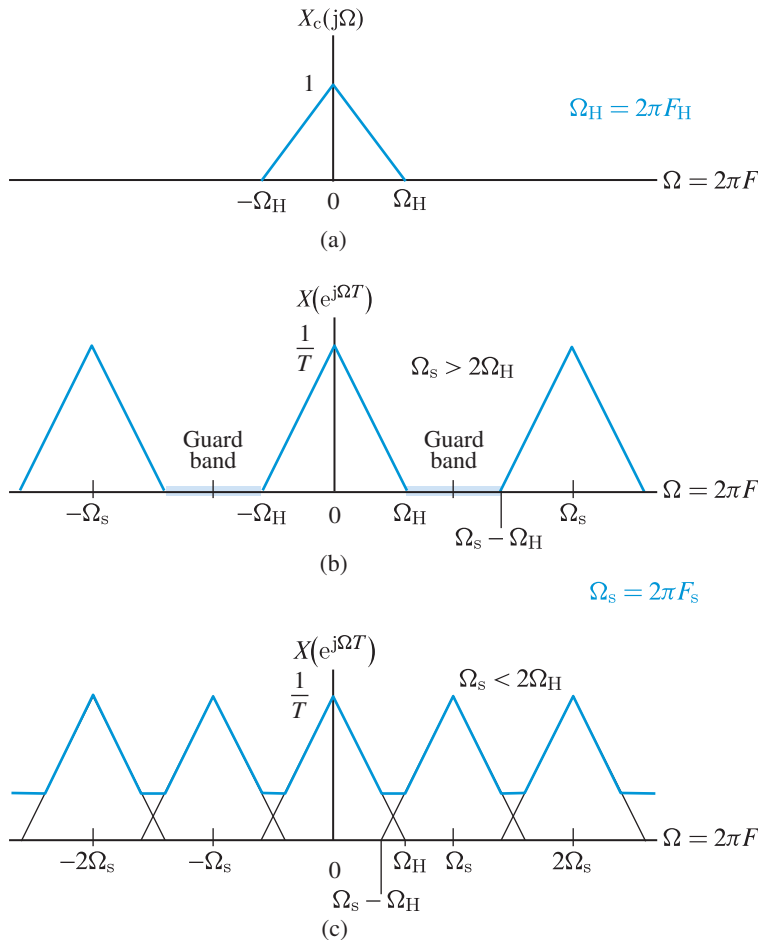
Figure 6.3 Frequency-domain interpretation of uniform sampling. (a) Spectrum of continuous-time bandlimited signal $x_c(t)$, (b) spectrum of discrete-time signal $x[n] = x_c(nT)$ with $\Omega_s > 2\Omega_H$, and (c) spectrum of $x[n]$, showing aliasing distortion, when $\Omega_s < 2\Omega_H$.

Sampling at $\Omega_s > 2\Omega_H$ creates a *guard band* as shown in Figure 6.3(b) which simplifies the reconstruction process in practical applications. If the sampled signal is sinusoidal, its spectrum consists of lines and hence equality in (6.16) does not hold; in this case, we should require that $\Omega_s > 2\Omega_H$ (see Tutorial Problem 4).

Terminology in sampling operation The highest frequency F_H, in Hz, present in a bandlimited signal $x_c(t)$ is called the *Nyquist frequency*. The minimum sampling frequency required to avoid overlapping bands is $2F_H$, which is called the *Nyquist rate*. The actual highest frequency that the sampled signal $x[n]$ contains is $F_s/2$, in Hz, and is termed as the *folding frequency*. These somewhat confusing frequency terms are illustrated in Figure 6.4.

Sampling theorem From Figure 6.3(b) it is immediately apparent that if $x_c(t)$ is bandlimited in Ω_H and $\Omega_s \geq 2\Omega_H$, we have
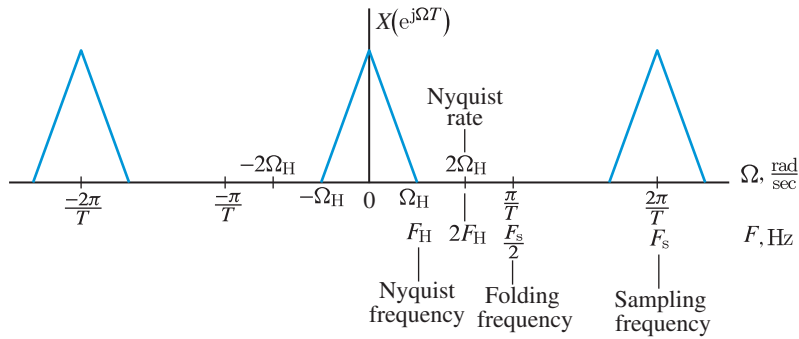
Figure 6.4 Frequency terminology related to sampling operation.

$$X_c(j\Omega) = \begin{cases} TX(e^{j\Omega T}), & |\Omega| \le \Omega_s/2 \\ 0. & |\Omega| > \Omega_s/2 \end{cases} \qquad (6.17)$$

Hence $x_c(t)$ can be recovered from $X_c(j\Omega)$ using the inverse Fourier transform. We note that $X_c(j\Omega)$ can be obtained by multiplying $X(e^{j\Omega T})$ by a function $G_{BL}(j\Omega)$, which is equal to T for $|\Omega| < \Omega_s/2$ and zero elsewhere (rectangular spectral windowing). This discussion is the basis for the famous *sampling theorem*, which can be stated as follows:

Sampling theorem: Let $x_c(t)$ be a continuous-time bandlimited signal with Fourier transform

$$X_c(j\Omega) = 0 \quad \text{for} |\Omega| > \Omega_H. \qquad (6.18)$$

Then $x_c(t)$ can be uniquely determined by its samples $x[n] = x_c(nT)$, where $n = 0, \pm 1, \pm 2, \dots$, if the sampling frequency Ω_s satisfies the condition

$$\Omega_s = \frac{2\pi}{T} \ge 2\Omega_H. \qquad (6.19)$$

In terms of the frequency $F = \Omega/2\pi$ in Hz, the conditions of the sampling theorem are: $X_c(j2\pi F) = 0$ for $F > F_H$ and $F_s = 1/T \ge 2F_H$. The sampling theorem, which in its most widely known form was first published in the communication theory context by Shannon (1949), provides the cornerstone of digital signal processing.

6.2   Reconstruction of a bandlimited signal from its samples

In this section we show how a properly sampled bandlimited signal can be reconstructed exactly from its samples. The objective is to find a formula which yields the values of $x_c(t)$ between samples in terms of the sample values. The purpose of this formula is to "fill-in" the gaps between samples or equivalently to "connect the dots" in a stem plot with
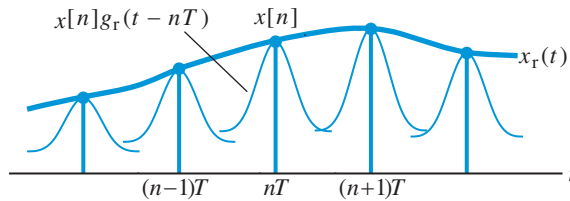
Figure 6.5 The interpolation process for reconstruction of a continuous-time signal from its samples.

a continuous-time curve. A general formula that describes a broad class of reconstruction processes is given by

$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n]g_r(t - nT), \tag{6.20}$$

where $x_r(t)$ is the reconstructed signal and $g_r(t)$ is an interpolating reconstruction function. The process of fitting a continuous function to a set of samples is known in numerical analysis as an *interpolation*. According to (6.20) the reconstructed signal is obtained by attaching a copy of $g_r(t)$ at each sample and summing all these curves. This idea is illustrated in Figure 6.5. We note that if the interpolation function has duration greater than or equal to T, the addition of the overlapping copies fills the gaps between samples.

To fully appreciate the meaning and implications of (6.20), we shall find its frequency-domain expression. Taking the CTFT of (6.20) yields

$$X_r(j\Omega) = \sum_{n=-\infty}^{\infty} x[n]G_r(j\Omega)\, e^{-j\Omega nT}. \tag{6.21}$$

Factoring $G_r(j\Omega)$ out of the summation, we obtain the desired formula

$$X_r(j\Omega) = G_r(j\Omega)\, X(e^{j\Omega T}). \tag{6.22}$$

This equation provides a natural approach to deriving the ideal reconstruction formula, whose existence is guaranteed by the sampling theorem.

To this end, we return to Figure 6.3(b), which shows the spectrum of the sampled signal when there is no aliasing. We note that if we choose $g_r(t)$ so that

$$G_r(j\Omega) \triangleq G_{BL}(j\Omega) = \begin{cases} T, & |\Omega| \leq \Omega_s/2 \\ 0, & |\Omega| > \Omega_s/2 \end{cases} \tag{6.23}$$

then $X_r(j\Omega) = X_c(j\Omega)$ and therefore $x_r(t) = x_c(t)$. Evaluating the inverse Fourier transform of $G_r(j\Omega)$, we obtain the interpolation function

$$g_r(t) \triangleq g_{BL}(t) = \frac{\sin(\pi t/T)}{\pi t/T}. \tag{6.24}$$

Substitution of (6.24) into (6.20) yields the following ideal reconstruction formula

$$x_c(t) = \sum_{n=-\infty}^{\infty} x[n] \frac{\sin\left[\pi(t-nT)/T\right]}{\pi(t-nT)/T}. \qquad (6.25)$$

Interpolation based on (6.25) is usually called *ideal bandlimited interpolation*, because it provides a perfect reconstruction for all t, if $x_c(t)$ is bandlimited in F_H and $F_s \geq 2F_H$. If there is aliasing, the reconstruction formula (6.25) generates a signal $x_r(t) \neq x_c(t)$ for all $t \neq nT$. The system used to implement (6.25), which is known as an *ideal DAC*, is depicted in block diagram form in Figure 6.6. For generality, we denote the output of the ideal DAC by $x_r(t)$.

To understand the meaning and implications of the ideal interpolation we look more closely at the sinc function $g_{BL}(t)$, which is depicted in Figure 6.7 (see discussion following (4.31)). We note that $g_{BL}(t) = 0$ at all sampling instants $t = nT$, except at $t = 0$ where $g_{BL}(t) = 1$. Thus, it is always true that $x_r(nT) = x_c(nT)$ regardless of whether aliasing distortion occurred during sampling. However, if there is no aliasing, equation (6.25) reconstructs the original signal perfectly for all values of time. Because $g_{BL}(t)$ has infinite
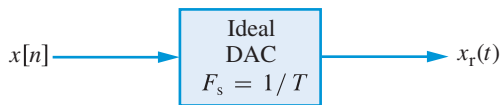
Figure 6.6 Representation of the ideal digital-to-analog converter (DAC) or ideal bandlimited interpolator.
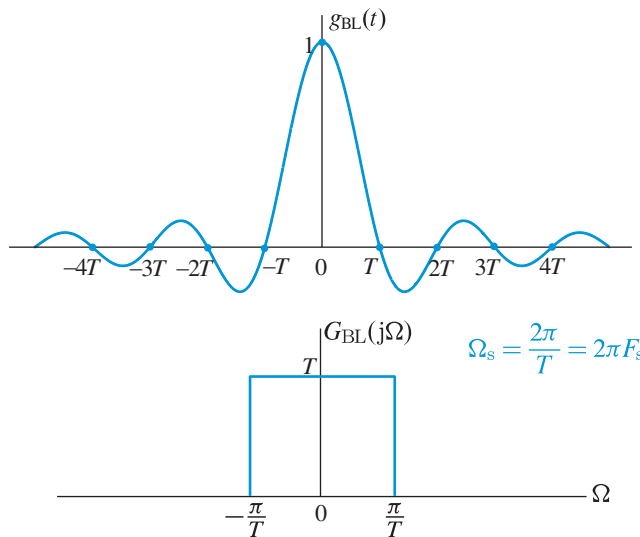
Figure 6.7 Time- and frequency-domain characteristics of the ideal DAC.
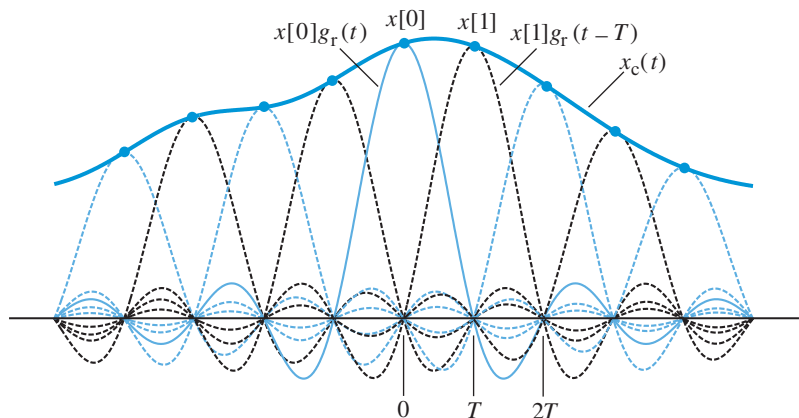
Figure 6.8 Ideal bandlimited interpolation in the time domain.

extent, each sample value contributes to the reconstruction $x_r(t)$ of $x_c(t)$ for all values of t. The time-domain bandlimited interpolation is shown in Figure 6.8.

In essence, equation (6.22) removes from $X(e^{j\Omega T})$ all replicas of $X_c(j\Omega)$ introduced by the sampling process. We note that if $x_c(t)$ is bandlimited in Ω_H and $\Omega_s \geq 2\Omega_H$, then $X_r(j\Omega) = X_c(j\Omega)$; otherwise, there is aliasing distortion which cannot be removed by the spectral windowing (6.22).

The relationships between the Fourier transform pair for continuous-time signals and the Fourier transform pair for discrete-time signals formed by uniformly sampling continuous-time signals are summarized in Figure 6.9. The relationships shown in blue lines are true for any continuous-time signal having a Fourier transform; the relationships shown in dashed lines are true only for bandlimited signals sampled at a rate greater than twice the maximum frequency present in the sampled signal.

6.3 The effect of undersampling: aliasing

According to the sampling theorem, a continuous-time signal $x_c(t)$ with frequencies no higher than F_H can be reconstructed exactly from its samples $x[n] = x_c(nT)$, if the samples are taken at a rate $F_s = 1/T$ that is greater than the Nyquist rate $2F_H$. The spectrum of $x[n]$ is obtained by scaling the spectrum of $x_c(t)$ by F_s and putting copies at all integer multiples of F_s.

If these copies do *not* overlap we can reconstruct perfectly the original signal by taking the inverse Fourier transform of the copy centered at zero frequency. If the spectral copies overlap, the spectrum of $x_c(t)$ is no longer recoverable from the spectrum of $x[n]$. In this case, the reconstructed signal $x_r(t)$ is related to the original signal $x_c(t)$ through aliasing distortion. In this section we explore the effects and consequences of sampling in general and aliasing in particular.
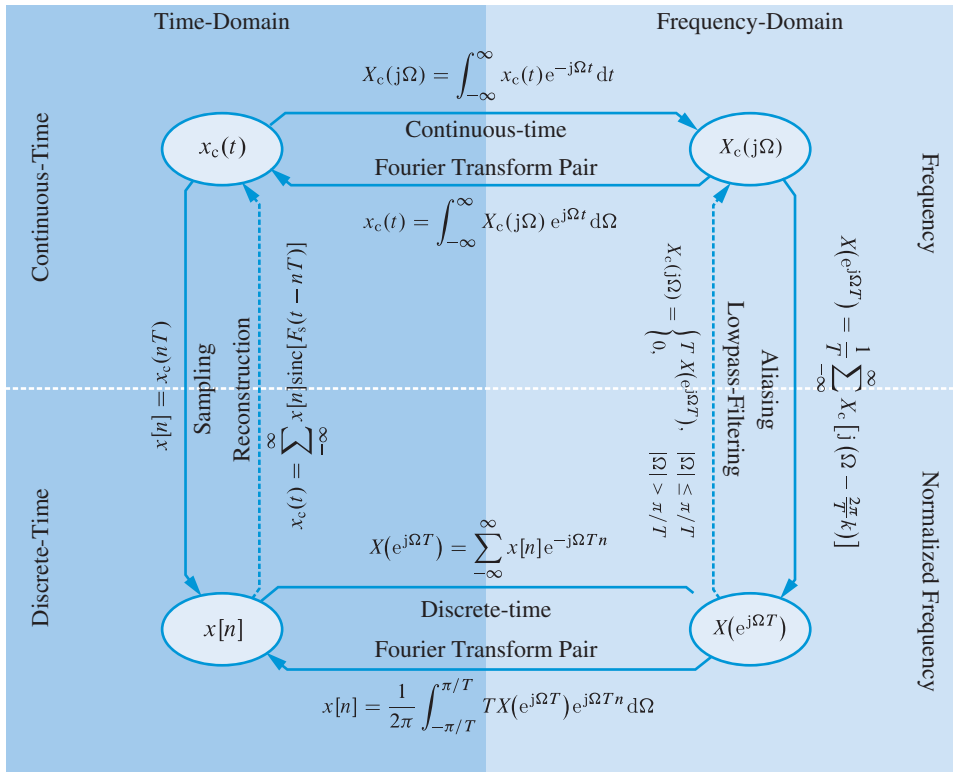
Figure 6.9 Relationships between the spectra of a continuous-time signal $x_c(t)$ and the discrete-time signal $x[n] = x_c(nT)$ obtained by periodic sampling. The dashed paths hold for bandlimited signals sampled at a rate $F_s > 2F_H$.
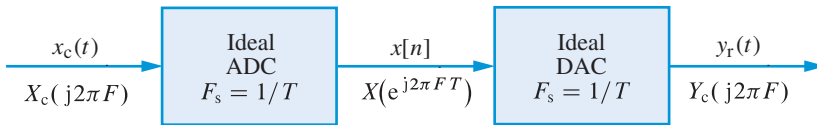
Figure 6.10 A talk-through system. Ideally, the reconstructed signal $y_r(t)$ should be identical to the input signal $x_c(t)$.

The phenomenon of aliasing has a clear meaning in the time-domain. Two continuous-time sinusoids of different frequencies appear at the *same* frequency when sampled. Since we cannot distinguish them based on their samples alone, they assume the same identity ("alias") and they produce the same continuous-time sinusoidal signal.

In this section we shall provide more insight into the causes and effects of aliasing using the system shown in Figure 6.10. This "talk-through" system is often used in practice to verify the correct operation and limitations of A/D and D/A converters before their actual application.

Example 6.1   Aliasing in sinusoidal signals

The continuous-time cosine signal

$$x_c(t) = \cos(2\pi F_0 t) = \frac{1}{2} e^{j2\pi F_0 t} + \frac{1}{2} e^{-j2\pi F_0 t} \tag{6.26}$$

has a discrete spectrum with spectral lines at frequencies $F = \pm F_0$, as shown in Figure 6.11(a). For convenience, we distinguish between the lines at $-F_0$ and F_0. We shall study the effect of changing the frequency F_0 while the sampling frequency F_s is kept fixed. We recall that uniform sampling of a continuous-time signal results in a repetition of its spectrum at all integer multiples of the sampling frequency. To obtain the spectrum of $x[n] = x_c(nT)$, we replicate (after scaling by $1/T$) the line at $F = F_0$ to $F = kF_s + F_0$ and the line at $F = -F_0$ to $F = kF_s - F_0$, for all k. There are two cases of interest.

First, suppose that $0 < F_0 < F_s/2$. Figure 6.11(a) shows the spectrum of $x_c(t)$ and Figure 6.11(b) shows the first three replicas of the spectrum of $x[n] = x_c(nT)$. The ideal DAC scales all spectral lines by T and removes all lines outside the fundamental range $-F_s/2 \le F \le F_s/2$; the reconstructed signal $x_c(t)$ has the spectrum shown in Figure 6.11(c). Since the spectra shown in Figures 6.11(a) and 6.11(b) are identical, we conclude that $x_r(t) = x_c(t)$, that is, there is no aliasing.

Suppose now that $F_s/2 < F_0 < F_s$, that is, the input frequency is outside the fundamental range, as illustrated in 6.11(d). After sampling, the line at $F = -F_0$ moves to $F = F_s - F_0$ and the line at $F = F_0$ moves to $F = -F_s + F_0$. As shown in Figure 6.11(e) these are the only lines within the fundamental range. In this case the reconstructed signal, shown in Figure 6.11(f), is given by

$$x_r(t) = \cos[2\pi (F_s - F_0)t] \ne x_c(t). \tag{6.27}$$

We note that as a result of aliasing the higher frequency signal $\cos(2\pi F_0 t)$ takes on the identity of or "impersonates" the lower frequency signal $\cos[2\pi (F_s - F_0)t]$. If we set $F_0 \triangleq F_s/2 + \Delta F$, where $0 \le \Delta F \le F_s/2$, the original signal can be written as

$$x_c(t) = \cos(2\pi F_0 t) = \cos[2\pi (F_s/2 + \Delta F) t]. \tag{6.28}$$

Since the *apparent frequency* of the reconstructed signal $x_c(t)$ is determined by $F_a = F_s - F_0 = F_s/2 - \Delta F$, we have

$$x_r(t) = \cos 2(\pi F_a t) = \cos[2\pi (F_s/2 - \Delta F) t]. \tag{6.29}$$

We note that sampling a sinusoidal signal below the Nyquist rate $F_s/2$ causes aliasing, which makes a sinusoid of higher frequency $F_0 = F_s/2 + \Delta F$ *appear* as a sinusoid of lower ("apparent") frequency $F_a = F_s/2 - \Delta F$. ∎
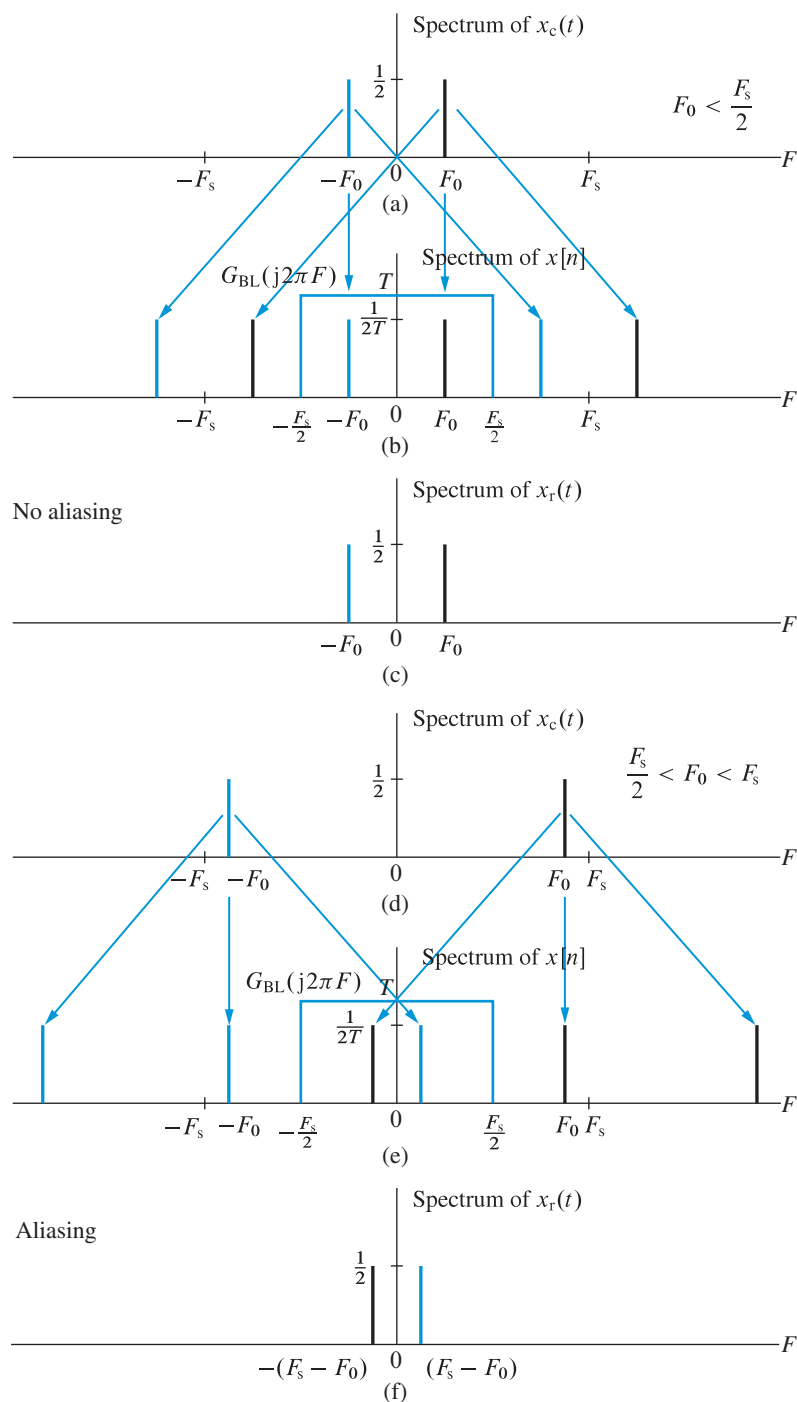
Figure 6.11 Effects of oversampling (see (a)–(c)) and undersampling (see (d)–(f)) in the frequency domain by sampling a continuous-time cosine signal. Undersampling (that is, choosing $F_s < 2F_0$) always causes aliasing.

Example 6.2   Verification of aliasing in the time domain

We shall demonstrate that sampling the signals $x_c(t)$ in (6.28) and $x_r(t)$ in (6.29), at a sampling frequency $F_s = 1/T$, results in identical sample values. Indeed, using the identities $\cos(a \pm b) = \cos(a)\cos(b) \mp \sin(a)\sin(b)$, $\cos(\pi n) = (-1)^n$, and $\sin(\pi n) = 0$ for all integer n, we can show that

$$\cos[2\pi \ (F_s/2 \pm \Delta F) \ nT] = (-1)^n \cos(2\pi \Delta F nT), \tag{6.30}$$

that is, the samples of a sinusoid of frequency $F_s/2 + \Delta F$ are identical to the samples of a sinusoid of frequency $F_s/2 - \Delta F$. This is illustrated in Figure 6.12.

In the case of a continuous-time sine signal, relation (6.30) takes the following form:

$$\sin[2\pi \ (F_s/2 \pm \Delta F) \ nT] = \pm(-1)^n \sin(2\pi \Delta F nT), \tag{6.31}$$

which shows that undersampling makes a sine signal of higher frequency $F_0 = F_s/2 + \Delta F$ appear as a sine of lower frequency $F_a = F_s/2 - \Delta F$ and opposite amplitude. The effect of phase in aliasing is discussed in Tutorial Problem 5.   ∎

Apparent frequency   The discussion in the previous examples considered aliasing for sinusoids in the frequency range $0 \le F_0 \le F_s$. However, we can use the identity

$$\cos[2\pi (kF_s + F_0)nT] = \cos(2\pi kn + 2\pi F_0 nT) = \cos(2\pi F_0 nT) \tag{6.32}$$

to understand aliasing for frequencies outside this range. After sampling and reconstruction at a rate $F_s = 1/T$, a sinusoid of frequency $F > F_s$ appears as a sinusoid of frequency $F_0 = F - kF_s$, where k is chosen such that $0 \le F_0 \le F_s$. If $0 \le F_0 \le F_s/2$, the apparent frequency of the reconstructed signal is $F_a = F_0$; however, if $F_s/2 \le F_0 \le F_s$, then $F_a = F_0 - F_s$. Stated differently, if $F_0 = F_s/2 \mp \Delta F$, where $0 \le \Delta F \le F_s/2$, then $F_a = \pm \Delta F$. The
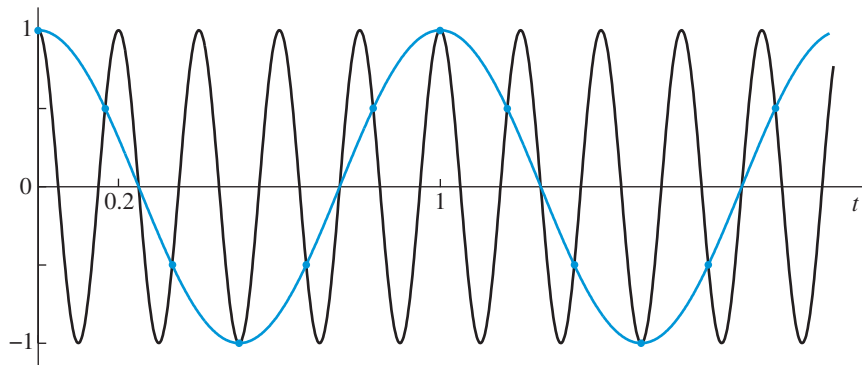
Figure 6.12 Demonstration of aliasing in sinusoidal signals. The signals $x(t) = \cos(2\pi F_1 t)$, $F_1 = 1$ Hz, and $x_2(t) = \cos(2\pi F_2 t)$, $F_2 = 5$ Hz, sampled at a rate $F_s = 6$ Hz generate the same set of samples. The ideal DAC reconstructs the sinusoid whose frequency is in the fundamental range.
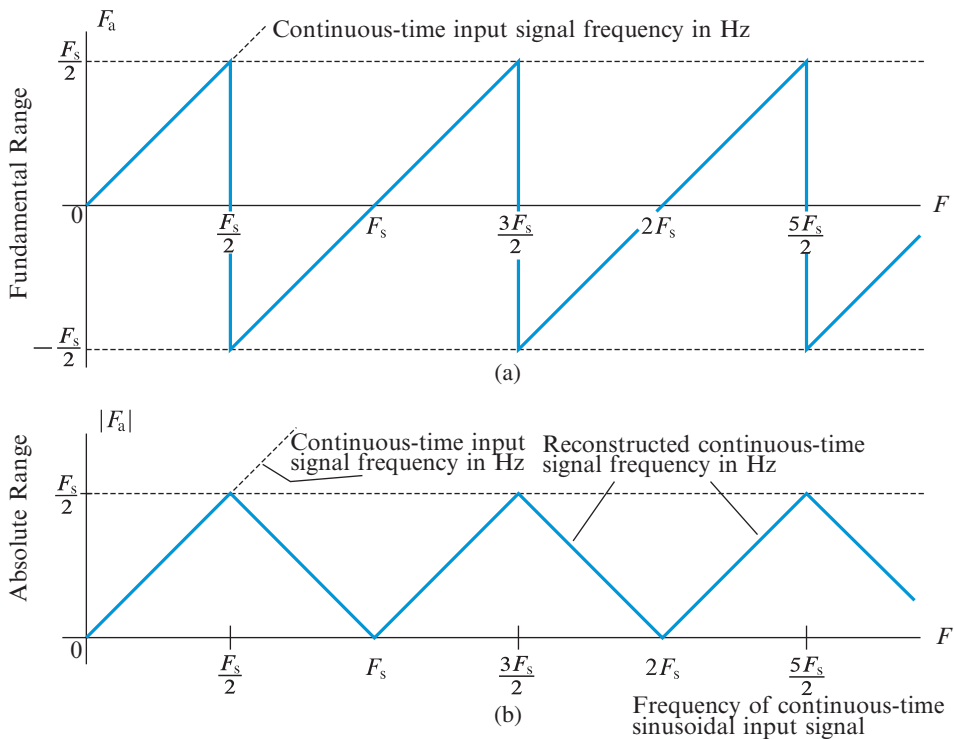
Figure 6.13 Apparent frequencies reconstructed by an ideal DAC of a continuous-time sinusoidal signal sampled by an ideal ADC: (a) fundamental range of F_a, and (b) absolute range $|F_a|$.

relationship between the original frequency F and the apparent frequency F_a is depicted graphically in Figure 6.13(a). This kind of shape should be expected because the ideal DAC always reconstructs a cosine or sine signal with frequency in the range $-F_s/2 < F_0 \leq F_s/2$. However, since $\cos(-2\pi F_a t + \theta) = \cos(2\pi F_a t - \theta)$, the *apparent value* of $-F_a$ is also F_a with a reversal of sign change in phase. This implies that the apparent frequency of any sinusoid lies in the range $0 \leq F_0 \leq F_s/2$ as shown in Figure 6.13(b). In the context of Figure 6.10, the apparent frequency is the lowest frequency of a sinusoid that has exactly the same samples as the input sinusoid.

Folding frequency The relation between the frequency of the sampled signal and apparent frequency of the reconstructed signal can also be explained by a different interpretation of the process shown in Figure 6.3. Figure 6.14(a) shows how the tails of the left $X_c[j2\pi(F + F_s)]$ and right $X_c[j2\pi(F-F_s)]$ shifted replicas distort the original spectrum $X_c(j2\pi F)$ in the fundamental range. If we focus in the range $0 \leq F \leq F_s/2$, we can create the same effect by first folding the right tail of $X_c(F)$ about $F_s/2$ and then fold again the "remaining" tail about zero, as shown in Figure 6.14(b). In general, we fold the right tail of $X_c(F)$ about $F_s/2$ and zero, until the entire tail is inside the fundamental interval. The folding or inversion of tails during sampling is the cause of aliasing. For this reason, the frequency $F_s/2$ is called the *folding frequency*. As a result of this folding, the higher frequency $F_s/2 + \Delta F$ appears

$$\frac{1}{T}X_c[j2\pi(F-F_s)] \qquad \frac{1}{T}X_c(j2\pi F) \qquad \frac{1}{T}X_c[j2\pi(F-F_s)]$$

(a)

$$\frac{1}{T}X_c(j2\pi F)$$

Folding frequency

First folding

Second folding

0 $\dfrac{F_s}{2}$

(b)
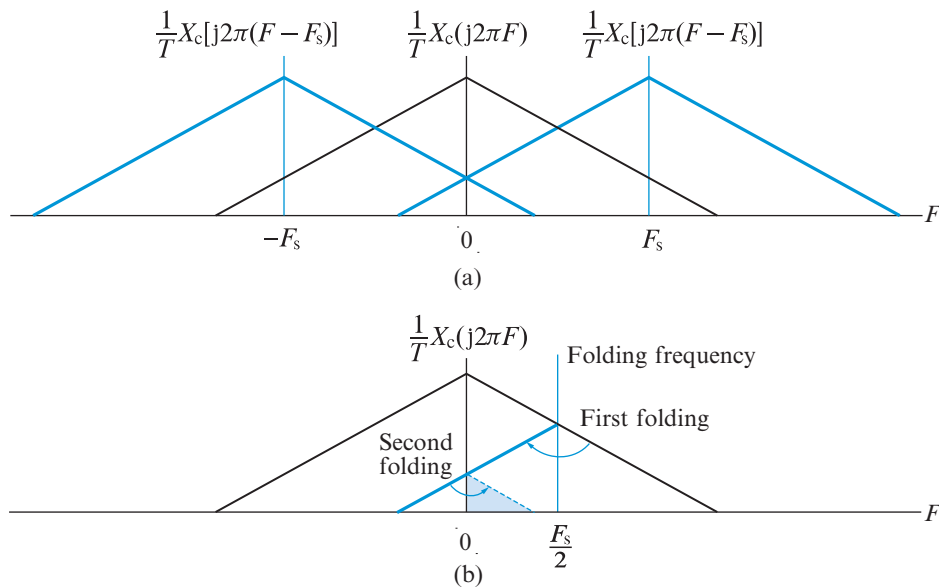
Figure 6.14 Aliasing distortion can be equivalently explained (a) by considering the overlapping tails of shifted replicas of the input spectrum, or (b) by folding the right tail of the input spectrum about $F_s/2$ and zero until the entire tail is in the range $0 \le F \le F_s/2$.

after reconstruction as a lower frequency $F_s/2 - \Delta F$; this multiple folding explains the relationship between the input and apparent frequencies shown in Figure 6.13.

Sampling a linear FM signal The consequences of aliasing, as described by the graph in Figure 6.13, can be lucidly demonstrated by applying a linear FM signal in the talk-through system of Figure 6.10. We use the linear FM signal $x_c(t) = \sin(\pi B t^2/\tau)$, $0 \le t \le \tau$, whose instantaneous frequency increases from $F = 0$ at $t = 0$ to $F = B$ at $t = \tau$ (see Example 5.2). Figure 6.15(a) shows the signal $x_c(t)$ for $B = 10$ Hz and $\tau = 10$ seconds, Figure 6.15(b) shows the samples of $x_c(nT)$ for $F_s = 1/T = B$ connected by line segments, and Figure 6.15(c) shows the output of the ideal DAC. As the input frequency increases from zero to the folding frequency $F_s/2$ the apparent frequency is equal to the input frequency; however, as the input frequency increases linearly from $F_s/2$ to F_s the apparent frequency decreases linearly from $F_s/2$ to zero. We note that it is only when the signal is oversampled with respect to the instantaneous apparent frequency that linear interpolation provides a good approximation of the reconstructed signal.

The "visual" effect of Figure 6.15 can be demonstrated acoustically by generating a linear FM signal and using the audio output of a computer to listen to the resulting sound. As the frequency rises from zero to the Nyquist frequency $F_s/2$, so does the perceived pitch. However, after this point the perceived pitch begins to fall even if the input frequency continues to rise. In fact, the perceived pitch follows the variation of the apparent frequency curve shown in Figure 6.13. A detailed description of this experiment is provided in Tutorial Problem 6.
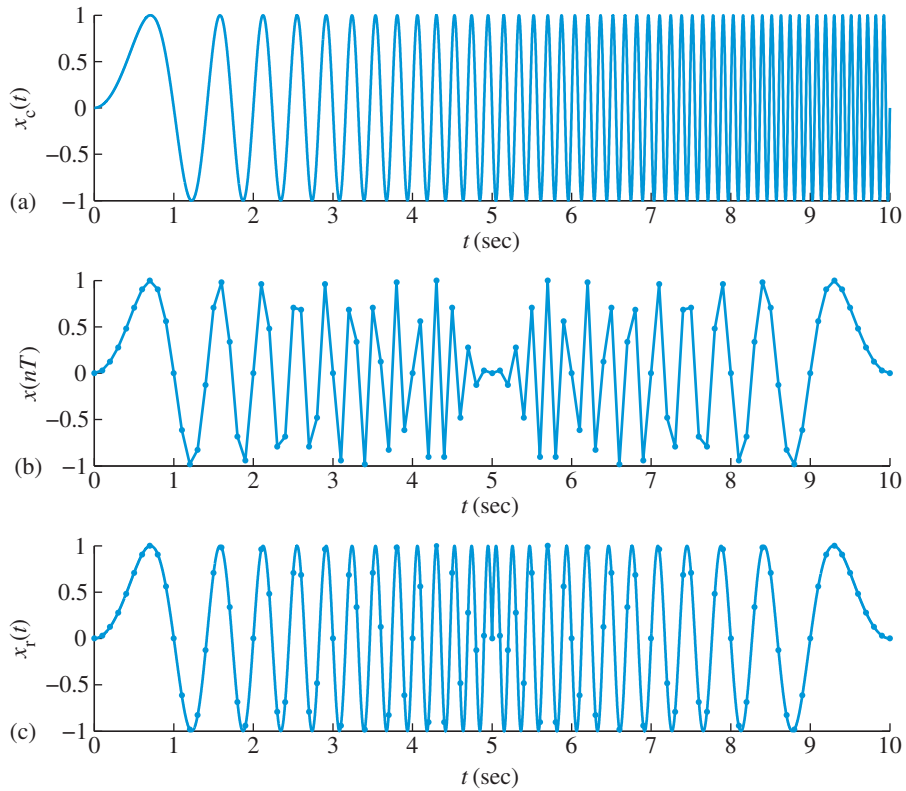
Figure 6.15 Sampling a continuous-time linear FM signal: (a) signal, (b) samples connected by line segments, and (c) output of ideal DAC.

Example 6.3   Aliasing in nonbandlimited signals

To illustrate the effects of aliasing for aperiodic signals, consider the two-sided exponential signal in Figure 6.16(a):

$$x_c(t) = e^{-A|t|} \xleftarrow{\text{CTFT}} X_c(j\Omega) = \frac{2A}{A^2 + \Omega^2}. \quad A > 0 \tag{6.33}$$

As shown in Figure 6.16(b) this signal has infinite duration and infinite bandwidth.

Sampling $x_c(t)$ at a rate $F_s = 1/T$ yields the discrete-time signal

$$x[n] = x_c(nT) = e^{-A|n|T} = (e^{-AT})^{|n|} = a^{|n|}, \quad a \triangleq e^{-AT}, \tag{6.34}$$

shown in Figure 6.16(c). The Fourier transform of $x[n]$ is given by (see Chapter 4)

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} = \frac{1 - a^2}{1 - 2a\cos(\omega) + a^2}, \quad \omega = \Omega/F_s \tag{6.35}$$
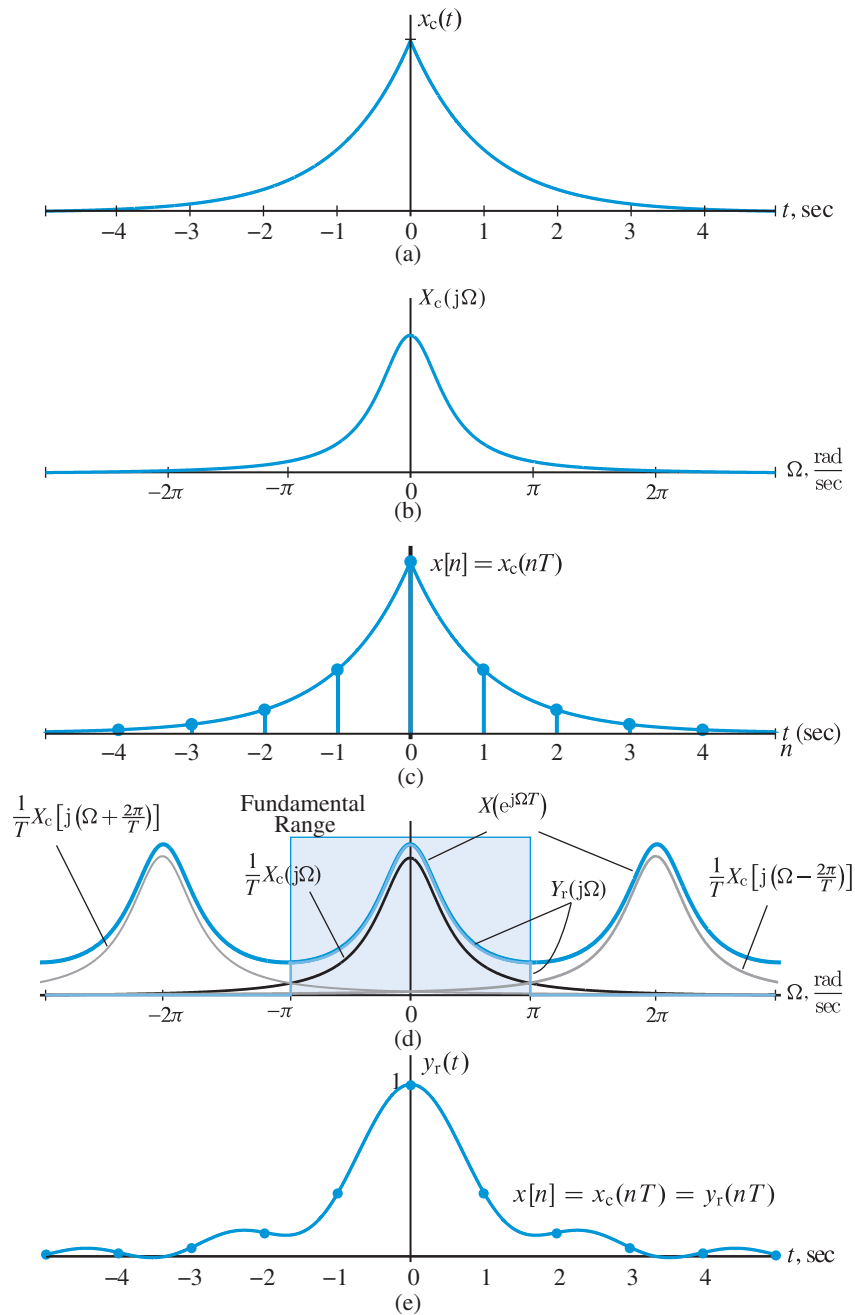
Figure 6.16 Aliasing effects in sampling and reconstruction of a continuous-time nonbandlimited signal: (a) continuous-time signal $x_c(t)$, (b) spectrum of $x_c(t)$, (c) discrete-time signal $x[n]$ sampled at $T = 1$ s, (d) spectrum of $x[n]$, and (e) bandlimited reconstruction $y_r(t)$. In this case, aliasing distortion is unavoidable.

which is periodic in ω with period 2π or periodic in Ω with period $2\pi/T$. Figure 6.16(d) shows the spectrum

$$X(e^{j\Omega T}) = \frac{1 - a^2}{1 - 2a\cos(\Omega T) + a^2}$$

of $x_c(nT)$ for $T = 1$ s. To explain the shape of $X(e^{j\Omega T})$ we recall its computation according to (6.12). The values of $X(e^{j\Omega T})$, within the fundamental range $-\pi/T \leq \Omega \leq \pi/T$, are determined by adding to $(1/T)X_c(j\Omega)$ (desired spectrum) the tails of $(1/T)X_c[j(\Omega - k2\pi/T)]$ for all $k \neq 0$ (aliased spectrum). The same result can be obtained by folding both tails of $(1/T)X_c(j\Omega)$ about the points π/T and $-\pi/T$ until the entire tails fall within the fundamental interval.

The reconstructed signal $y_r(t)$ corresponds to the inverse Fourier transform of $Y_c(j\Omega) = TX(e^{j\Omega T})$ for $|\Omega| < \pi/T$ and $Y_c(j\Omega) = 0$ for $|\Omega| > \pi/T$. The signal $y_r(t)$ shown in Figure 6.16 was obtained using the approximation

$$y_r(m\Delta t) \approx \sum_{n=N_1}^{N_2} x[n] \frac{\sin[\pi(m\Delta t - nT)/T]}{\pi(m\Delta t - nT)/T}, \quad t_1 \leq m\Delta t \leq t_2 \tag{6.36}$$

A MATLAB implementation of (6.36) as a matrix-vector multiplication is given by

```
t=(t1:dt:t2); ts=(t1:T:t2); [G1,G2]=meshgrid(t,ts);
S=sinc(Fs*(G1-G2)); yr=x*S;
```

where the row vector x contains the available signal samples.

We note that as a result of aliasing, $y_r(t) \neq x_c(t)$ for $t \neq nT$. The strong frequency components of $Y_c(j\Omega)$ close to π/T create oscillations in $y_r(t)$ with an approximate period of $1/(F_s/2) = 2T$. The rounding of $y_r(t)$ around zero is due to elimination of the high frequency components of $X_c(j\Omega)$. Because $x_c(t)$ has infinite bandwidth, sampling at a higher rate reduces but does not eliminate overlapping between $X_c(j\Omega)$ and the tails of its shifted replicas. However, as the sampling rate increases this overlapping decreases and $y_r(t)$ provides a more accurate approximation of $x_c(t)$. These issues are discussed, with more detail, in Tutorial Problem 7. ∎

Example 6.4 Sampling of a periodic signal

In this example we consider sampling effects on Fourier series coefficients when a periodic signal is sampled to obtain a periodic sequence. Figure 6.17(a) shows a periodic triangular signal $x_c(t)$ with period $T_0 = 4$, given by

$$x_c(t) = x_c(t + 4) = \begin{cases} t, & -2 \leq t \leq 0 \\ -t, & 0 \leq t \leq 2 \end{cases}$$

Its Fourier series is given by the CTFS coefficients (see (4.25))
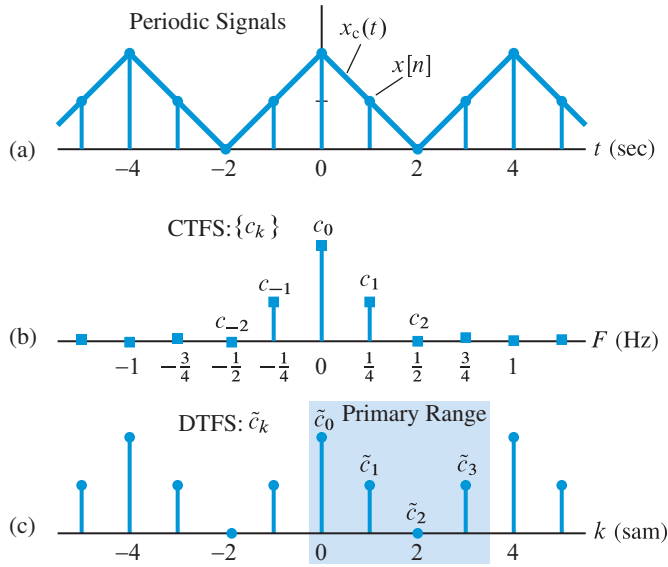
Figure 6.17 Sampling of a periodic signal: (a) periodic signal $x_c(t)$ and its samples $x[n]$, (b) CTFS coefficients $\{c_k\}$, and (c) DTFS coefficients $\tilde{c}_k$

$$c_k = \frac{1}{T_0}\int_{T_0} x_c(t)\,e^{-j\frac{2\pi}{T_0}kt}\,dt = \frac{1}{4}\left[\int_{-2}^{0} t e^{-j\frac{2\pi}{4}kt}\,dt - \int_{0}^{2} t e^{-j\frac{2\pi}{4}kt}\,dt\right]$$

$$= \begin{cases} 1, & k=0, \\ \frac{4}{\pi^2 k^2}, & k=\pm 1,\pm 3,\dots; \\ 0, & k=\pm 2,\pm 4,\dots \end{cases} \qquad F_0 = \frac{1}{T_0} = \frac{1}{4}\ \text{Hz},$$

which are shown in Figure 6.17(b) in which harmonics are $\frac{1}{4}$ Hz apart. Clearly, $x_c(t)$ has an infinite number of harmonics and hence an infinite bandwidth.

We sample $x_c(t)$ at a rate of $F_s = 1/T = N/T_0 = 1$ Hz with $T = 1$ and $N = 4$ to obtain the periodic sequence $x[n] = \{2, 1, 0, 2\}$, $0 \le n \le 3$ with period $N = 4$, also shown in Figure 6.17(a). Then according to the aliasing formula (6.14), all higher harmonics of $x_c(t)$ above $|k| = 2$ will be folded and added to the lower harmonics to obtain $N = 4$ periodic DTFS description $\{\tilde{c}_k\}$ of $x[n]$ where the added $\tilde{\ }$ is used to signify a periodic quantity. In particular,

$$\tilde{c}_0 = \cdots + c_{-4} + c_0 + c_4 + \cdots = \sum_{\ell} c_{0-4\ell} = \cdots + 0 + 1 + 0 + \cdots = 1,$$

$$\tilde{c}_1 = \cdots + c_{-3} + c_1 + c_5 + \cdots = \sum_{\ell} c_{1-4\ell} = \sum_{\ell} \frac{4}{\pi^2(1-4\ell)^2} = \frac{1}{2},$$

$$\tilde{c}_2 = \cdots + c_{-2} + c_2 + c_6 + \cdots = \sum_{\ell} c_{2-4\ell} = 0,$$

$$\tilde{c}_3 = \cdots + c_{-1} + c_3 + c_7 + \cdots = \sum_\ell c_{3-4\ell} = \sum_\ell \frac{4}{\pi^2(3-4\ell)^2} = \frac{1}{2},$$

or

$$\tilde{c}_k = \sum_\ell c_{k-\ell N} = \begin{cases} 1, & k = 0 \\ \frac{1}{2}, & k = \pm 1, \pm 3, \ldots \\ 0. & k = \pm 2, \pm 4, \ldots \end{cases} \qquad N = 4$$

Indeed using (4.68), the DTFS coefficients of $x[n]$ are given by

$$\tilde{c}_k = \frac{1}{N} \sum_{k=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} = \frac{1}{4} \left[2 + e^{-j\frac{\pi}{2}k} + e^{-j3\frac{\pi}{2}k} \right]$$

$$= \frac{1}{2} \left[1 + \cos(\pi k/2) \right] = \begin{cases} 1, & k = 0 \\ \frac{1}{2}, & k = \pm 1, \pm 3, \ldots \\ 0. & k = \pm 2, \pm 4, \ldots \end{cases}$$

which agree with the aliasing formula and are shown in Figure 6.17(c).   ■

In conclusion, a careful sampling of a periodic signal produces a periodic sequence whose DTFS is an aliased version of the corresponding CTFS coefficients given by

$$\tilde{c}_k = \sum_{\ell=-\infty}^{\infty} c_{k-\ell N}, \quad k = 0, \pm 1, \pm 2, \ldots \tag{6.37}$$

Note a similarity with the DTFT–CTFT aliasing formula (6.14). This aspect is discussed in more detail in Tutorial Problem 8.

6.4   Discrete-time processing of continuous-time signals

In many applications it is advantageous to filter a continuous-time signal using a discrete-time filter. The process involves three steps implemented by the systems shown in Figure 6.18.

Ideal ADC The input to the ideal ADC is a continuous-time signal $x_c(t)$ and the output is a sequence of samples $x[n]$ defined by

$$x[n] = x_c(t)|_{t=nT} = x_c(nT). \tag{6.38}$$

We emphasize that once the samples are taken from $x_c(t)$ and stored in memory, the time scale information is lost. The discrete-time signal $x[n]$ is just a sequence of numbers, and

Effective continuous-time system
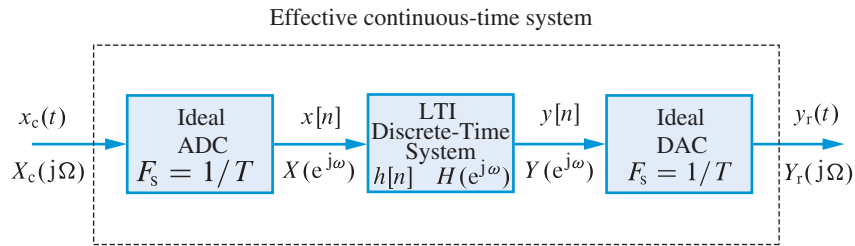
Figure 6.18 Discrete-time filtering of continuous-time signals.

these numbers carry no information about the sampling period, T, which was used to obtain them. Conceptually, in the time domain, the ideal ADC performs two operations:

1. Samples the signal $x_c(t)$ every T seconds and assigns the value $x_c(nT)$ to the nth sample of the sequence.
2. Scales or normalizes the time axis, by $n = t/T$, so that the distance between successive samples is always unity and changes the time axis from absolute time (t) to normalized time (n).

In the frequency-domain the operation of the ideal ADC is described by

$$X(e^{j\omega})|_{\omega=\Omega T} = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c\left(j\Omega - j\frac{2\pi}{T}k\right). \tag{6.39}$$

We emphasize that the equality between the left and right sides of (6.39) holds only if $\omega = \Omega T$ (see (6.12) in Section 6.1). In this sense, the ideal ADC performs the following operations:

1. Scales the input spectrum $X_c(j\Omega)$ by $1/T$ and copies the scaled spectrum $(1/T)X_c(j\Omega)$ at all integer multiples of the sampling frequency $\Omega_s = 2\pi/T$.
2. Scales the frequency axis using the relation $\omega = \Omega T = 2\pi F/F_s$. In other words, it changes the frequency axis from absolute frequency (F or Ω) to normalized frequency (f or ω). This frequency scaling, $f = F/F_s$, is directly related to the time scaling $n = t/T$ introduced in (6.38).

Figure 6.19(a)–(c) illustrates the operation of the ideal ADC in both the time-domain and the frequency-domain.

Discrete-Time LTI System The sequence $x[n]$ is processed by a discrete-time LTI system to produce another sequence $y[n]$. The operation of the LTI system is described by the following relationships

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k], \tag{6.40}$$

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}), \tag{6.41}$$
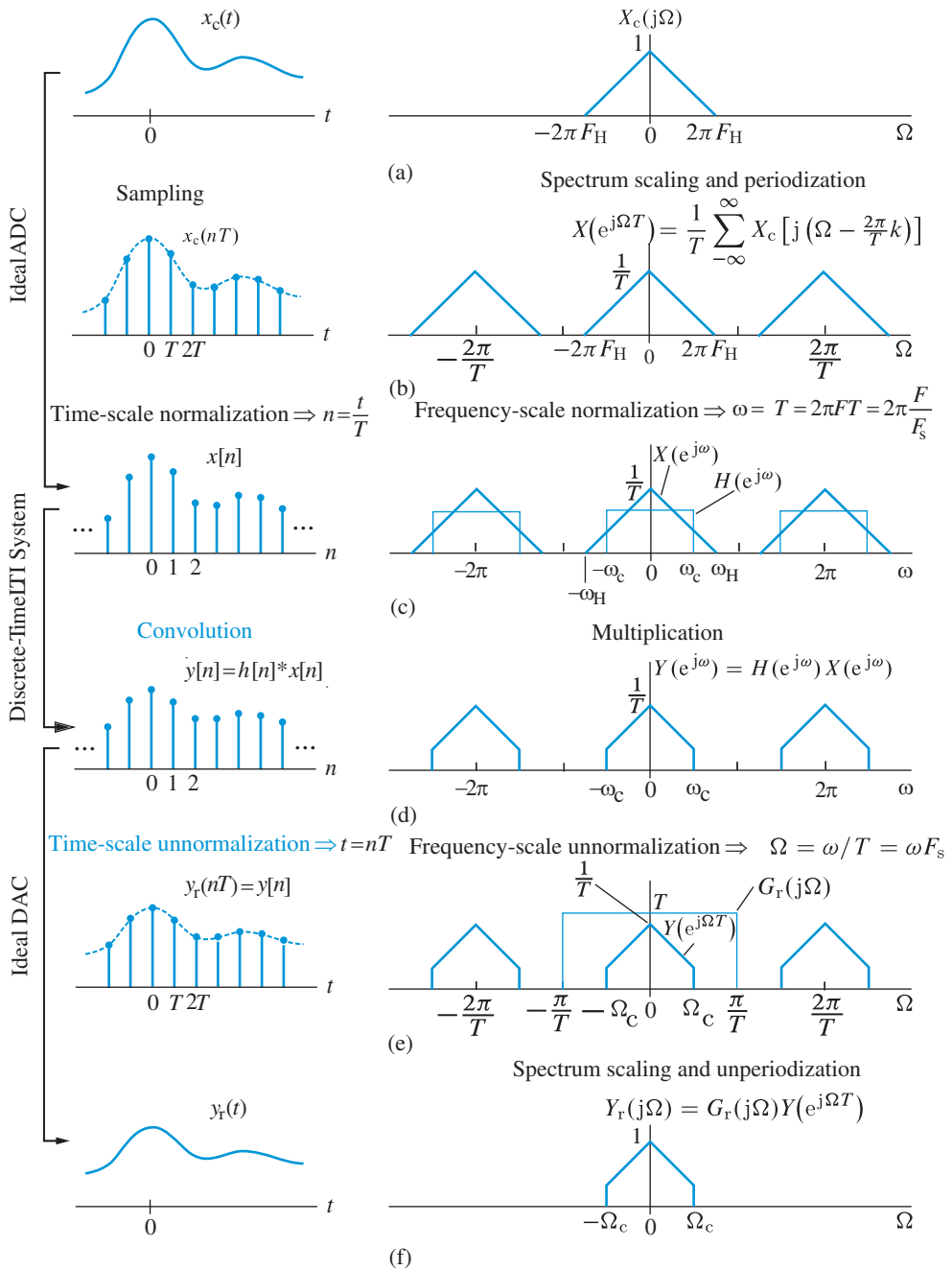
Figure 6.19 Discrete-time filtering of continuous-time signals: (a)–(c) operations performed by the ideal ADC, (c)–(d) operation performed by the discrete-time LTI filter, and (d)–(f) operations performed by ideal DAC. In the absence of aliasing the overall system is equivalent to a continuous-time LTI system.

where $h[n]$ is the impulse response of the system and $H(e^{j\omega})$ its frequency response. Figure 6.19(c)–(d) illustrates this process for the ideal lowpass filter

$$H(e^{j\omega}) = \begin{cases} 1, & |\omega| < \omega_c \\ 0. & \omega_c < |\omega| \le \pi \end{cases} \tag{6.42}$$

Ideal DAC The input to the ideal DAC is a sequence of numbers $y[n]$ and the output is a continuous-time signal $y_r(t)$. The output of the ideal DAC is given by

$$y_r(t) = \sum_{n=-\infty}^{\infty} y[n]g_{BL}(t - nT), \tag{6.43}$$

$$Y_r(j\Omega) = G_{BL}(j\Omega)Y(e^{j\Omega T}), \tag{6.44}$$

where $g_{BL}(t)$ is the ideal interpolation function, defined in (6.24), and $G_{BL}(j\Omega)$ is its Fourier transform. The discrete-time signal $y[n]$ is just an indexed sequence of numbers; absolute time information is introduced by the ideal DAC using the sampling period T provided by the user. More specifically, the ideal DAC performs the following operations:

1. Scales the time axis to absolute time ($t = nT$) and the frequency axis to absolute frequency ($\Omega = \omega/T$ or $F = \omega F_s/2\pi$).
2. Scales the input spectrum by T and removes all replicas (images) outside the fundamental frequency range $-F_s/2 < F < F_s/2$ or $-\pi/T < \Omega < \pi/T$.

These operations are illustrated in Figure 6.19(d)–(f). We note that the ideal DAC is a linear, time varying system; therefore, (6.43), which resembles a convolution sum, it is *not* a convolution operation (see Tutorial Problem 9).

Effective continuous-time filter To understand the operation of the overall system in Figure 6.19, we substitute (6.39) and (6.41) into (6.44). The result is

$$Y_r(j\Omega) = G_{BL}(j\Omega)H(e^{j\Omega T})X(e^{j\Omega T})$$

$$= G_{BL}(j\Omega)H(e^{j\Omega T})\frac{1}{T}\sum_{k=-\infty}^{\infty} X\left(j\Omega - j\frac{2\pi}{T}k\right). \tag{6.45}$$

If $X_c(j\Omega)$ is bandlimited to $2\pi F_H$, the sampling frequency satisfies $F_s > 2F_H$, and we use the ideal DAC $G_{BL}(j\Omega)$ given by (6.23), then (6.45) simplifies to

$$Y_r(j\Omega) = \begin{cases} H(e^{j\Omega T})X_c(j\Omega), & |\Omega| \le \pi/T \\ 0. & |\Omega| > \pi/T \end{cases} \tag{6.46}$$

This is equivalent to

$$Y_r(j\Omega) = H_{eff}(j\Omega)X_c(j\Omega), \tag{6.47}$$

where

$$
H_{\text{eff}}(j\Omega) = \begin{cases} H(e^{j\Omega T}), & |\Omega| \leq \pi/T \\ 0. & |\Omega| > \pi/T \end{cases} \tag{6.48}
$$

We note that although the ideal ADC and DAC are linear time-varying systems, the over-all system in Figure 6.19 is equivalent to an effective LTI continuous-time system whose frequency response is given by (6.48). This is possible only if (a) the discrete-time system is LTI, and (b) there is no aliasing during the sampling process (see Tutorial Problem 11).

As an example, for the ideal lowpass filter (6.42), we have

$$
H_{\text{eff}}(j\Omega) = \begin{cases} 1, & |\Omega| \leq \omega_c/T \\ 0. & |\Omega| > \omega_c/T \end{cases} \tag{6.49}
$$

We note that the cutoff frequency of the effective continuous-time ideal lowpass filter, $\Omega_c = \omega_c/T$, depends on both the normalized cutoff frequency ω_c and the sampling period T. Thus, we could implement a lowpass filter with variable cutoff frequency by varying the sampling rate.

Example 6.5   Ideal bandlimited differentiator

One useful application of discrete-time processing of analog systems is in the implementation of a differentiator, which is defined by $y_c(t) = dx_c(t)/dt$. The frequency response of this ideal differentiator is

$$
H_c(j\Omega) = \frac{Y_c(j\Omega)}{X_c(j\Omega)} = j\Omega, \tag{6.50}
$$

whose analog implementation is generally problematic. If the signal of interest $x_c(t)$ is bandlimited to Ω_H frequency, then we can use a bandlimited differentiator with frequency response

$$
H_c(j\Omega) = \begin{cases} j\Omega, & |\Omega| \leq \Omega_H \\ 0. & \text{otherwise} \end{cases} \tag{6.51}
$$

If we sample the impulse response $h_c(t)$ using $\Omega_s = 2\Omega_H$ to obtain the discrete-time differentiator $h[n] = h_c(nT)$, then the corresponding discrete-time frequency response from (6.39) is given by

$$
H(e^{j\omega})|_{\omega=\Omega T} = \frac{1}{T} \sum_{k=-\infty}^{\infty} H_c\left(j\Omega - j\frac{2\pi}{T}k\right), \quad T = \frac{\pi}{\Omega_H} \tag{6.52}
$$

which from (6.51), and using $\omega = \Omega T$ and $\Omega_H T = \pi$, is

$$
H(e^{j\omega}) = \frac{1}{T}H_c(j\omega/T) = \frac{j\omega}{T^2}. \quad |\omega| \leq \pi \tag{6.53}
$$

Hence the impulse response of the discrete-time differentiator is given by

$$
h[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left(\frac{j\omega}{T^2} \right) e^{j\omega n} d\omega =
\begin{cases}
0, & n = 0 \\
\frac{\cos(\pi n)}{nT^2}. & n \neq 0
\end{cases}
\tag{6.54}
$$

This impulse response can now be used as a discrete-time system in Figure 6.18 to implement an ideal bandlimited differentiator. ∎

Example 6.6   Second-order system

Consider a second-order system described by the linear differential equation (5.231)

$$
\frac{d^2 y_c(t)}{dt^2} + 2\zeta \Omega_n \frac{dy_c(t)}{dt} + \Omega_n^2 y_c(t) = \Omega_n^2 x_c(t),
\tag{6.55}
$$

where ζ is the damping ratio and Ω_n is the undamped natural frequency. The system function of this system is given by (5.232)

$$
H_c(s) = \frac{Y_c(s)}{X_c(s)} = \frac{\Omega_n^2}{s^2 + 2\zeta \Omega_n s + \Omega_n^2}.
$$

For $0 < \zeta < 1$, the system is stable and the impulse response is oscillatory, given by (5.234)

$$
h_c(t) = \frac{\Omega_n}{\sqrt{1-\zeta^2}} e^{-\zeta \Omega_n t} \sin\left[\left(\Omega_n \sqrt{1-\zeta^2} \right) t \right] u(t),
$$

which is a nonbandlimited system. We want to implement this system using the discrete-time signal processing approach for which we will obtain the impulse response of the discrete-time system in Figure 6.18 by sampling the above impulse response. Let the sampling frequency be $F_s = 1/T$. Then the required discrete-time impulse response is given by

$$
\begin{aligned}
h[n] = h_c(nT) &= \frac{\Omega_n}{\sqrt{1-\zeta^2}} e^{-\zeta \Omega_n nT} \sin\left[\left(\Omega_n \sqrt{1-\zeta^2} \right) nT \right] u(n) \\
&= \frac{\Omega_n}{\sqrt{1-\zeta^2}} \left(e^{-\zeta \Omega_n T} \right)^n \sin\left[\left(\Omega_n T \sqrt{1-\zeta^2} \right) n \right] u(n).
\end{aligned}
\tag{6.56}
$$

This approach of converting a continuous-time system into a corresponding discrete-time system is known as the *impulse-invariance* transformation and is discussed in more detail in Section 11.3.1. Using Table 3.1, the z-transform of (6.56) is given by

$$
\begin{aligned}
H(z) &= \frac{\Omega_n}{\sqrt{1-\zeta^2}} \sum_{n=0}^{\infty} \left(e^{-\zeta \Omega_n T} \right)^n \sin\left[\left(\Omega_n T \sqrt{1-\zeta^2} \right) n \right] z^{-n} \\
&= \frac{\Omega_n}{\sqrt{1-\zeta^2}} \frac{e^{-\zeta \Omega_n T} \sin\left(\Omega_n T \sqrt{1-\zeta^2} \right) z^{-1}}{1 - 2e^{-\zeta \Omega_n T} \cos\left(\Omega_n T \sqrt{1-\zeta^2} \right) z^{-1} + e^{-2\zeta \Omega_n T} z^{-2}},
\end{aligned}
\tag{6.57}
$$
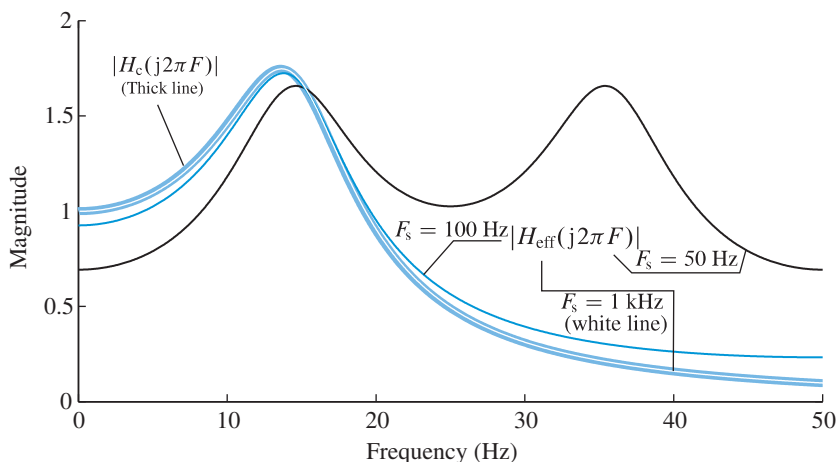
Figure 6.20 Discrete-time processing of the second-order system in Example 6.6 by sampling its impulse response. The thick line is the magnitude $|H_c(j2\pi F)|$ while the three thin lines are magnitudes $|H_{\text{eff}}(j2\pi F)|$ for $F_s = 50$, 100, and 1 kHz.

or the difference equation of the discrete-time system is

$$
y[n] = \frac{\Omega_n}{\sqrt{1-\zeta^2}} e^{-\zeta\Omega_n T} \sin\left(\Omega_n T\sqrt{1-\zeta^2}\right) x[n-1]
$$

$$
+ 2e^{-\zeta\Omega_n T} \cos\left(\Omega_n T\sqrt{1-\zeta^2}\right) y[n-1] - e^{-2\zeta\Omega_n T} y[n-2]. \tag{6.58}
$$

For example, if $\zeta = 0.3$, $\Omega_n = 30\pi$, and $F_s = 1$ kHz, then the difference equation (6.58) of the required discrete-time filter is

$$
y[n] = 8.6234x[n-1] + 1.9364y[n-1] - 0.945y[n-2].
$$

Since the continuous-time system is nonbandlimited, the frequency response of the effective filter $H_{\text{eff}}(j\Omega)$ will not match with that of the original $H_c(j\Omega)$ over the entire frequency range, especially for small values of the sampling frequency F_s. For $\zeta = 0.3$ and $\Omega_n = 30\pi$, Figure 6.20 shows magnitudes of the continuous-time system frequency responses $|H_c(j\Omega)|$ and resulting effective filter response $|H_{\text{eff}}(j\Omega)|$ for $F_s = 50$, 100, and 1000 Hz. For $F_s = 50$ Hz, the aliasing is very pronounced and results in a mismatch of two frequency responses. For $F_s = 100$ Hz the aliasing is small, but visible and the effective filter response matches that of the continuous-time filter near the resonant frequency (around 15 Hz). However, for $F_s = 1$ kHz, the two frequency responses are accurate up to $F = 50$ Hz. This example clearly shows the aliasing effect of sampling and the need for higher sampling rates for nonbandlimited signals. See also Tutorial Problem 10 for the effect on phase response in discrete-time processing. ∎

6.5   Practical sampling and reconstruction

Practical sampling and reconstruction differ from ideal sampling and reconstruction in three fundamental aspects:

1. All practical continuous-time signals are timelimited, that is, they have finite duration; therefore they are not, and cannot be, strictly bandlimited.
2. In practice, the sampled values $x[n]$ can only be described by a finite number of bits; that is, the values of $x_c(nT)$ should be quantized.
3. The ideal DAC is practically unrealizable because the interpolation kernel $g_{BL}(t) = \sin(\pi t/T)/(\pi t/T)$ has infinite duration.

The block diagram in Figure 6.21 shows a more realistic model of a practical system for digital processing of analog (continuous-time) signals. In this section we discuss individual components in terms of their functions, variations in implementation, and potential sources of signal degradation.

6.5.1   Analog-to-digital conversion

The analog-to-digital conversion process involves three essential systems: an analog low-pass filter, a sample-and-hold circuit, and an A/D converter. It is important to realize that any degradation in signal quality introduced at this stage will remain with the digitized signal.

Lowpass antialiasing filter The sole function of the analog lowpass filter before the sample-and-hold circuit is to bandlimit the input signal to the folding frequency without introducing excessive linear or nonlinear distortion, and without generating excessive
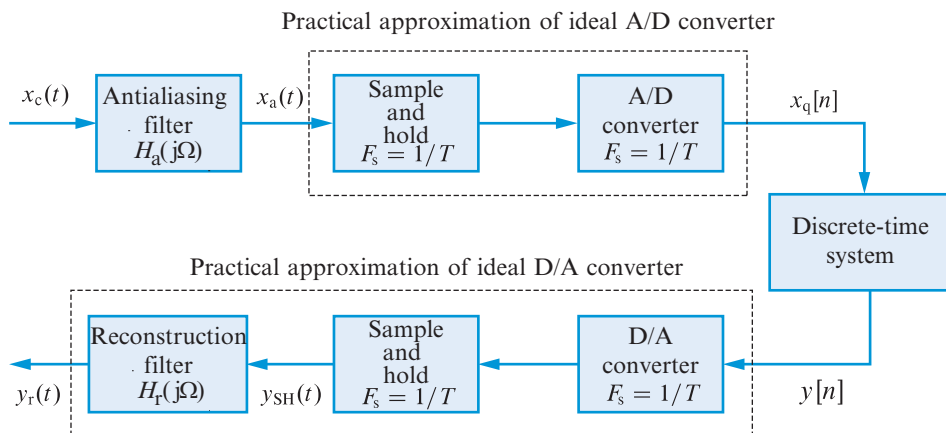
Figure 6.21 A realistic model for digital processing of continuous-time signals. The sample and hold circuit and the DAC are usually implemented as a single system; they are shown separately for conceptual reasons.

noise. This is a critical operation because when the filter's output signal is sampled, any frequency components above the folding frequency will result in aliasing distortion. We emphasize that even if the signal is bandlimited, there is always wideband additive noise which will be folded back to create aliasing. This requires an analog lowpass filter $H_a(j\Omega)$ with steep roll-off and very high attenuation of all frequency components above $F_s/2$. The design and implementation of "good" antialiasing filters for Nyquist rate sampling is difficult and expensive. However, as we will study in Chapter 12, we can use simpler and inexpensive antialiasing filters if we allow for oversampling followed by digital filtering compensation.

Sample-and-hold (S/H) circuit When an analog voltage is connected directly to the input of an ADC, the conversion process can be adversely affected if the analog voltage is changing during the conversion time. The quality of the conversion process can be improved by using a S/H circuit to hold the analog voltage constant while the A/D conversion is taking place. The simplest implementation of the S/H function, using a switch and a capacitor, is shown in Figure 6.22(a). When the switch is in the "sample" position, the S/H tracks the input signal and the capacitor charges up to the input value. When the switch is moved to the "hold" position, the capacitor has no discharge path and stores an analog quantity (charge) that represents the signal at the sampling instant. This process, which is repeated at each sampling interval, is illustrated in Figure 6.22(b). Usually, an ADC has a built-in sample-and-hold function; this type of ADC is known as a *sampling ADC*.

Since the sampling operation is performed by the S/H circuit, the role of S/H is to sample $x_c(t)$ as instantaneously as possible and to hold the sample value as constant as possible

(a)

(b)
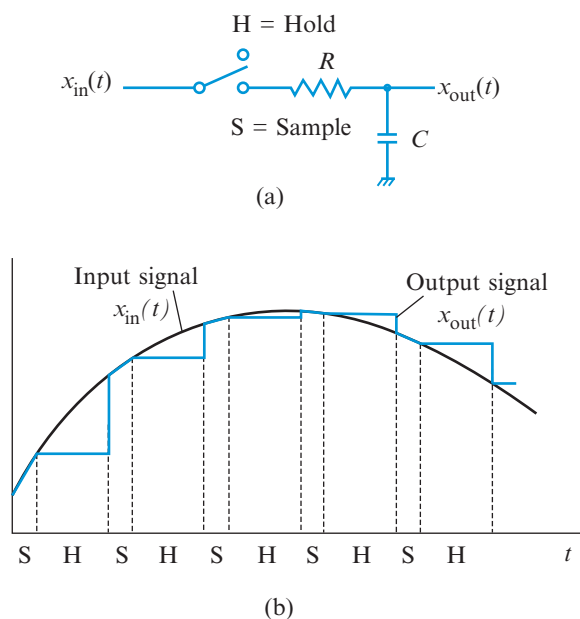
Figure 6.22 (a) Simplified diagram of a sample-and-hold circuit. (b) Example of input and output signals. Note that during the sample mode the S/H tracks the input signal.
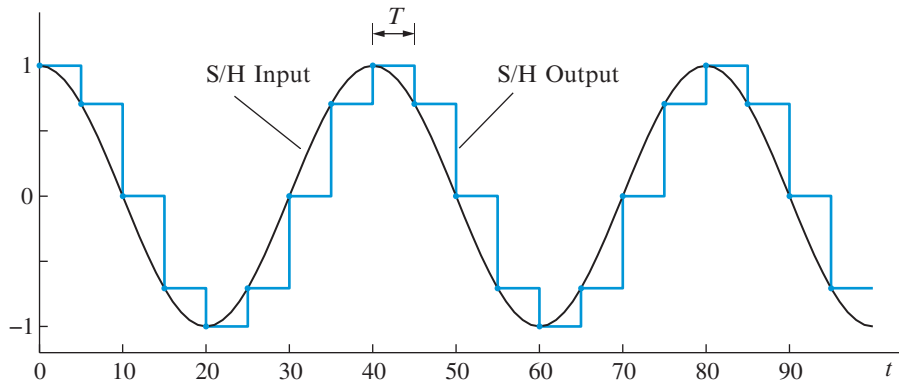
Figure 6.23 Input and output signals of an ideal sample-and-hold circuit.

until the next sample (or at least as long as it takes for the ADC to complete the conversion). Thus, the output of the S/H circuit can be modeled as a staircase waveform where each sample value is held constant until the acquisition of the next sample (see Figure 6.23). We note that although the input is a single sinusoid, the output is clearly nonsinusoidal; thus, it is *not* possible to describe a S/H by a system function or a frequency response function. As we learned in Section 5.1, it is a fundamental property of LTI systems that a sinusoidal input generates a sinusoidal output with the same frequency. The S/H system is linear but time-varying (see Tutorial Problem 13).

A/D converter The ADC is a physical device that converts the voltage or current value at its input into a binary word, which is the numerical representation of a quantized value closest to the input value. The major difference between ideal and practical conversion is that an ADC generates sample values that are known with finite precision. The ADC is the device in which both quantization and binary coding of the sampled signal take place.

The basic function of a quantizer is to electronically define a range of input values, subdivide that range into a set of subregions, and then decide within which subregion the input sample lies. The coder generates the binary word corresponding to the assigned level. The type of binary representation used is not important for the present discussion; the critical point is that a B-bit quantizer can represent 2^B different numbers. Binary number representations are discussed in Chapter 15. The basic idea is illustrated in Figure 6.24 for a uniform quantizer. It is called uniform because the input amplitude range is divided into K *quantization intervals* of equal width Δ (*quantization step*) and the output levels are uniformly spaced. Since all values within each quantization interval are represented by a single value, quantization *always* results in loss of information. This allocation of intervals to a number of discrete levels, called *quantization levels*, is illustrated in Figure 6.24(a); as a result, the input–output transfer characteristic of the uniform quantizer has the staircase form shown in Figure 6.24(b). The quantization error, that is the difference between the staircase function and the ideal straight line $y = x$, is shown in Figure 6.24(c). We note that the quantization error as a function of input signal amplitude has a triangular shape characteristic.
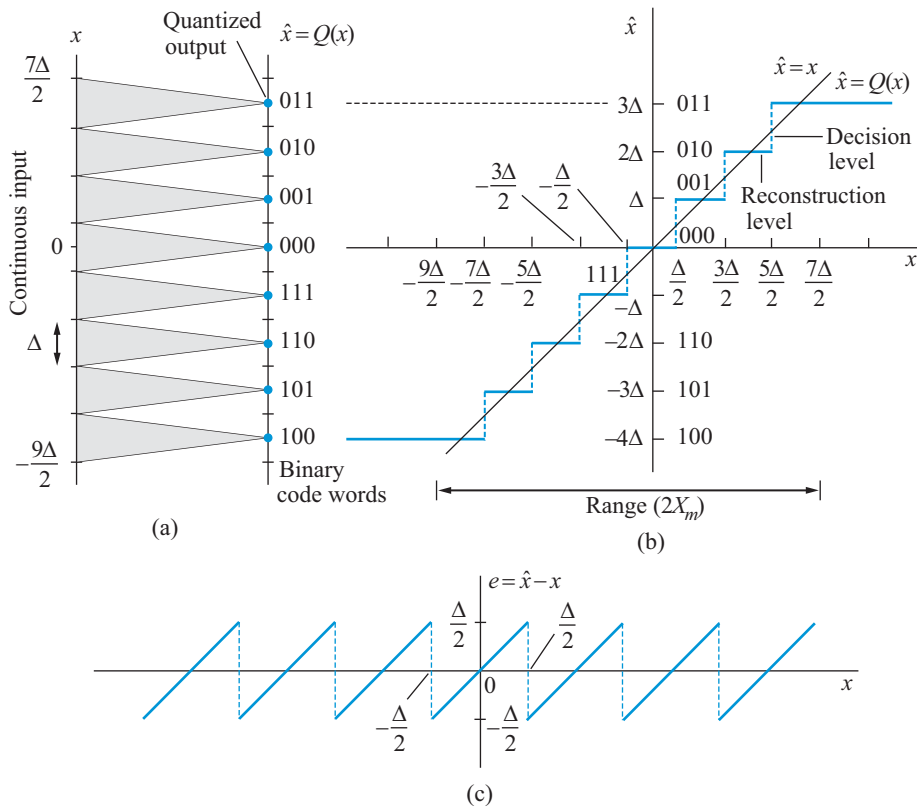
Figure 6.24 The quantization operation allocates intervals to a number of discrete levels, that is, quantization is a many-to-one mapping. (a) Allocation of levels in a 3-bit quantizer which rounds x/Δ to the closest integer. Input-output (b) and quantization error (c) transfer function of a uniform rounding quantizer.

In MATLAB several functions are available to perform quantization depending on the strategy used: the `round(x)` function quantizes x to the nearest integer; the `fix(x)` function quantizes x to the nearest integer towards 0; the `ceil(x)` function quantizes x to the nearest integer towards ∞; and the `floor(x)` function quantizes x to the nearest integer towards $-\infty$. With appropriate scaling, these functions can be used to quantize a value to any number of digits or bits.

Based on how this decision is made we can classify A/D converters into three broad families, as set out below. These A/D converters operate at the Nyquist sampling rate; a family of A/D converters that operate at much higher sampling rates (oversampling A/D converters) is discussed in Chapter 15.

Serial or integrating converters decide by examining one subregion at a time, going from one end of the range to the other (linear search); this requires 2^B clock periods (one level at a time converters). Such converters, although they have a low throughput, are widely used in instrumentation applications due to their simplicity and insensitivity to hardware imperfections.

Successive approximation converters find the subregion where the input signal lies, using a binary search approach. The input voltage is first compared with a reference value that is half the maximum. If the input voltage is greater, the most significant bit is set, and the signal is then compared with a reference level that is three-fourths the maximum to determine the next bit, and so on (one bit at a time converters). The advantage is that the result can be obtained on the order of B clock periods; however, the cost is higher because it requires a highly accurate high-speed DAC. Such converters are used in telecommunication applications.

Parallel or flash converters examine all subregions simultaneously using one comparator per subregion (one word at a time converters). The result can be obtained in the order of one clock cycle, but the cost of hardware is much greater than that for successive approximation converters. Such converters are used for high sampling rate applications, like radar, sonar, and video.

Quantization noise The two major types of error introduced by an ADC are aliasing error and quantization error. Since quantization is a nonlinear operation, analysis of quantization error is done using statistical techniques, that is, it is treated as a random signal (see Chapter 14). However, if we assume that F_s satisfies the sampling theorem there is no aliasing error. In this case we can obtain a useful formula for the power of quantization error by quantizing the continuous-time signal $x_c(t)$ instead of the discrete-time signal $x[n] = x_c(nT)$.

The basic idea is illustrated in Figure 6.25, which shows the quantization of a continuous-time sinusoidal signal. We note that if there is a large number of small quantization intervals, the quantization error resembles the triangular error characteristic shown in Figure 6.24(c). The exceptional cases occur when the signal goes through a maximum or minimum within a quantization step. Since the signal $x_c(t)$ is almost linear between quantization levels, the equation for the quantization error signal, say $e_c(t) \triangleq x_q(t) - x_c(t)$, in a typical interval is given by

$$e_c(t) = \frac{\Delta}{2\tau}t, \quad -\tau \le t \le \tau \tag{6.59}$$

where for convenience we assume that the line is centered about time zero. Then the mean squared quantization error power is
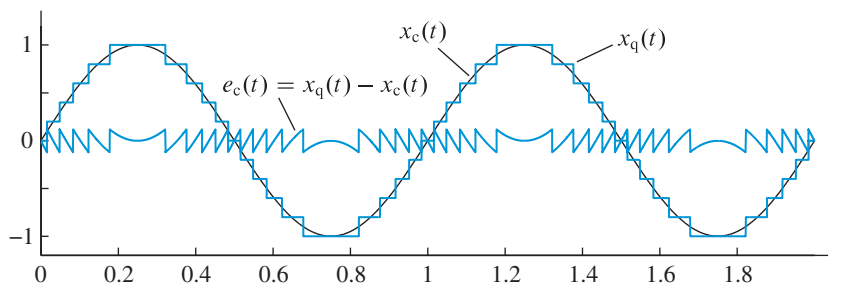
Figure 6.25 Quantization error resulting from the quantization of a continuous-time sinusoidal signal using a rounding quantizer with $\Delta = 0.2$.

$$P_Q = \frac{1}{2\tau} \int_{-\tau}^{\tau} e_c^2(t) \mathrm{d}t = \frac{\Delta^2}{12}. \tag{6.60}$$

As shown in Chapter 14, this approximation is sufficiently accurate for most signals of practical interest, as long as we have a large number of small quantization steps and the signal spans and remains within the range of the quantizer.

The average power for a sinusoidal signal $x_c(t) = X_m \sin(\frac{2\pi}{T_p}t)$, with period T_p which spans the range of the quantizer, is given by

$$P_S = \frac{1}{T_p} \int_0^{T_p} X_m^2 \sin^2\left(\frac{2\pi}{T_p}\right) \mathrm{d}t = \frac{X_m^2}{2}. \tag{6.61}$$

The universally accepted criterion of performance for an ideal quantizer is the signal-to-quantization noise ratio (SQNR), which is defined by

$$\mathrm{SQNR} \triangleq \frac{P_S}{P_Q} = \frac{3}{2} \times 2^{2B}. \tag{6.62}$$

To derive this formula we have used (6.61), (6.62), and the expression $\Delta = (2X_m)/2^B$ for the quantization step of a B-bit quantizer. Expressing the SQNR in decibels, we obtain the fundamental relation

$$\mathrm{SQNR(dB)} = 10\log_{10}\mathrm{SQNR} = 6.02B + 1.76. \tag{6.63}$$

This is a theoretical maximum which is widely used as a rule of thumb for selection of A/D converters for practical applications. The key conclusion is that each additional bit in the quantizer adds 6 dB to the SQNR.

MATLAB audio ADC functions Audio signals can be digitized in MATLAB for further processing using platform-specific ADC functions. The `wavrecord(N,Fs)` function is for use only with 32-bit Microsoft Windows operating systems and records `N` samples of an audio signal available through the PC-based audio hardware, sampled at a rate of `Fs` Hz. The standard sampling rates are 8000, 11025, 2250, and 44100 Hz, the default value being 11025 Hz. To record signals from audio input devices on other operating systems, the function `audiorecorder` is available which creates an 8000 Hz, 8-bit object in MATLAB and can support sampling rates up to 48 kHz on properly configured sound cards.

6.5.2   Digital-to-analog conversion

In Section 6.2 we showed how a bandlimited signal can be reconstructed from a sequence of samples using the ideal DAC described by

$$x_r(t) = \sum_{n=-\infty}^{\infty} x[n]g_r(t - nT), \tag{6.64}$$

where $g_r(t)$ is the ideal interpolation function $g_{BL}(t)$ in (6.24). Since $g_{BL}(t) \neq 0$ for $t < 0$ and $\int |g_{BL}(t)|dt = \infty$, the ideal DAC is a noncausal and unstable system; hence, it is not practically realizable. For a practical reconstruction system $g_r(t)$ should be zero for $t < 0$ and absolutely summable. A system that implements (6.64), for an arbitrary function $g_r(t)$, is known as a (practical) *digital-to-analog converter* (DAC). The main objective of a DAC is to "fill-in" the signal values between the sampling times $t = nT$, a process known as interpolation. The interpolation function $g_r(t)$ is known as the *characteristic pulse* of the converter. At each sample time $t = nT$, the converter generates a pulse $g_r(t - nT)$ scaled by the value of the current sample $x[n]$. In general, the continuous-time reconstructed signal $x_r(t)$ consists of an infinite sum of scaled and shifted characteristic pulses. The shape of $g_r(t)$, or equivalently of $G_r(j\Omega)$ determines the quality of the reconstructed signal.

In practice, the conversion from a digital signal to analog typically is implemented with the three devices shown in Figure 6.21. The DAC generates an analog voltage at its output which is determined by the binary word at its input. The basic idea behind the operation of a DAC is that the binary bits cause electronic switches to open or close, thus routing the electric current through an appropriate network of resistors to generate the correct voltage level. Because no counting or searching is required, D/A converters tend to be much faster than A/D converters.

The next system is a special S/H amplifier which prevents the internal switching glitches in the DAC from appearing at the output analog signal. This is done by holding the output voltage of the DAC constant for one sampling period; the result is a staircase continuous-time signal. Since the DAC simply maps the binary input word to the quantized value $x_q[n]$, the interpolation is performed by the S/H amplifier. Thus, the output of the S/H is given by

$$x_{SH}(t) = \sum_{n=-\infty}^{\infty} x_q[n]g_{SH}(t - nT). \tag{6.65}$$

The characteristic pulse of the S/H circuit and its Fourier transform are

$$g_{SH}(t) = \begin{cases} 1, & 0 \leq t \leq T \\ 0. & \text{otherwise} \end{cases} \xrightarrow{\text{CTFT}} G_{SH}(j\Omega) = \frac{2\sin(\Omega T/2)}{\Omega}e^{-j\Omega T/2} \tag{6.66}$$

Figure 6.26 compares the frequency domain behavior of the S/H circuit with that of the ideal D/A converter. The S/H circuit, unlike the ideal DAC, does *not* completely eliminate the replicated spectral images introduced by the sampling process; moreover, it introduces an amplitude distortion (known as *droop*) in the Nyquist band $|F_s| < F_s/2$. The maximum droop roll-off is $|G_{SH}(j0)/G_{SH}(j\pi/T)| = \pi/2$ or about 4 dB at $F = F_s/2$. These effects are illustrated in Figure 6.26.

To compensate for the effects of the S/H circuit we use an analog lowpass post-filter $H_r(j\Omega)$ such that $G_{SH}(F)H_r(F) = G_{BL}(F)$ as shown in Figure 6.27. This implies that

$$H_r(j\Omega) = \begin{cases} \frac{\Omega T/2}{\sin(\Omega T/2)}e^{j\Omega T/2}, & |\Omega| < \pi/T \\ 0. & \text{otherwise} \end{cases} \tag{6.67}$$

This reconstruction filter eliminates the surviving spectral images (anti-imaging filter) and compensates for the droop (equalization filter). Since the time advance of $T/2$ seconds
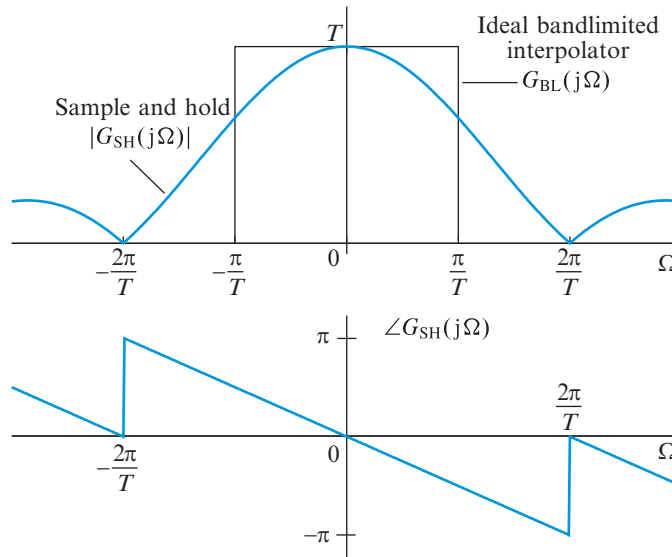
Figure 6.26 Frequency domain characteristics of the S/H system. The characteristics of the ideal bandlimited interpolator are included for comparison.

is practically unrealizable, we can only compensate for the magnitude response of the S/H circuit. Quite often the compensation for the droop is ignored because the 4 dB drop is insignificant for large values of F_s. The most economical way to compensate for droop distortion or imperfections in the antialiasing filter is by appropriate digital filtering before the DAC (see Tutorial Problem 14).

In the following example we give a complete time- and frequency-domain reconstruction analysis for a sinusoidal input signal when an ideal S/H circuit, followed by a reconstruction filter, is used in the D/A block of Figure 6.21.

Example 6.7   Practical reconstruction of sinusoidal signals

A sinusoidal signal $x_c(t) = \cos 2\pi F_0 t$ with $F_0 = 0.025$ Hz is sampled at a rate of $F_s = 1/T = 0.2$ Hz. The result is the discrete-time signal $x[n] = x_c(nT) = \cos(2\pi f_0 n)$ with $f_0 = 1/8$. The signals $x_c(t)$, $x[n]$, and their spectra are shown in Figure 6.28; for simplicity, we assume that there is no quantization.

In practice, the ideal reconstructor is approximated by a DAC followed by a S/H circuit and an anti-imaging lowpass filter. At each sampling interval, the DAC takes as input a digital word and generates a voltage equal to $x[n]$; this value is held constant for T seconds by the S/H circuit. The output $x_{SH}(t)$ of the S/H is the staircase periodic waveform; therefore, it has a discrete aperiodic spectrum given by $X_{SH}(j2\pi F) = G_{SH}(j2\pi F)X(e^{j2\pi FT})$. The S/H scales the magnitude of each input frequency component by the sinc function and introduces a time delay of $T/2$ seconds. In Figure 6.28 we only show the magnitude of $X_{SH}(j2\pi F)$ and $G_{SH}(j2\pi F)$.

The reconstruction filter removes all frequency components outside the Nyquist interval, compensates for the droop distortion, and scales the input amplitude by T. Therefore, the
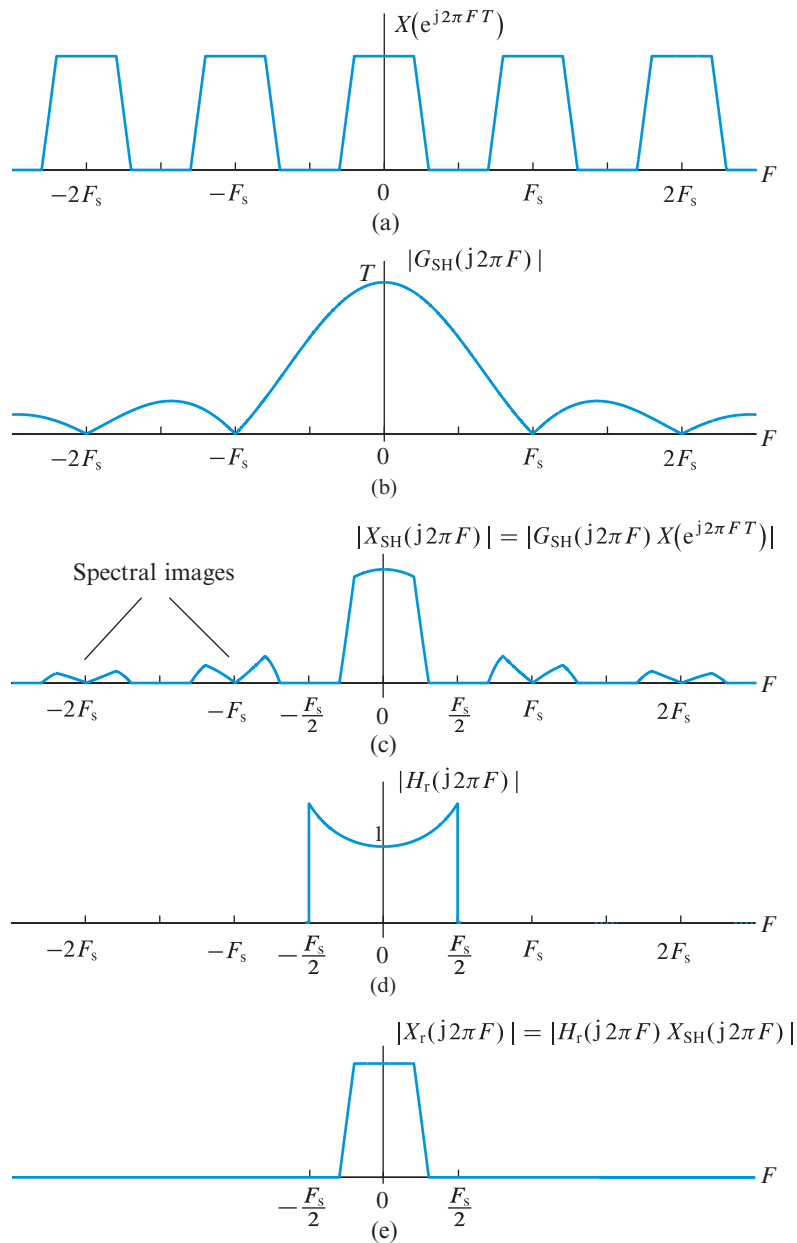
Figure 6.27 Frequency domain characteristics of S/H reconstruction.

final reconstructed signal is given by

$$x_r(t) = \tfrac{1}{2}e^{-jT/2}e^{j2\pi F_0 t} + \tfrac{1}{2}e^{jT/2}e^{-j2\pi F_0 t} = \cos(2\pi F_0 t - T/2),$$

which is identical with the input signal delayed by $T/2$ seconds. ■
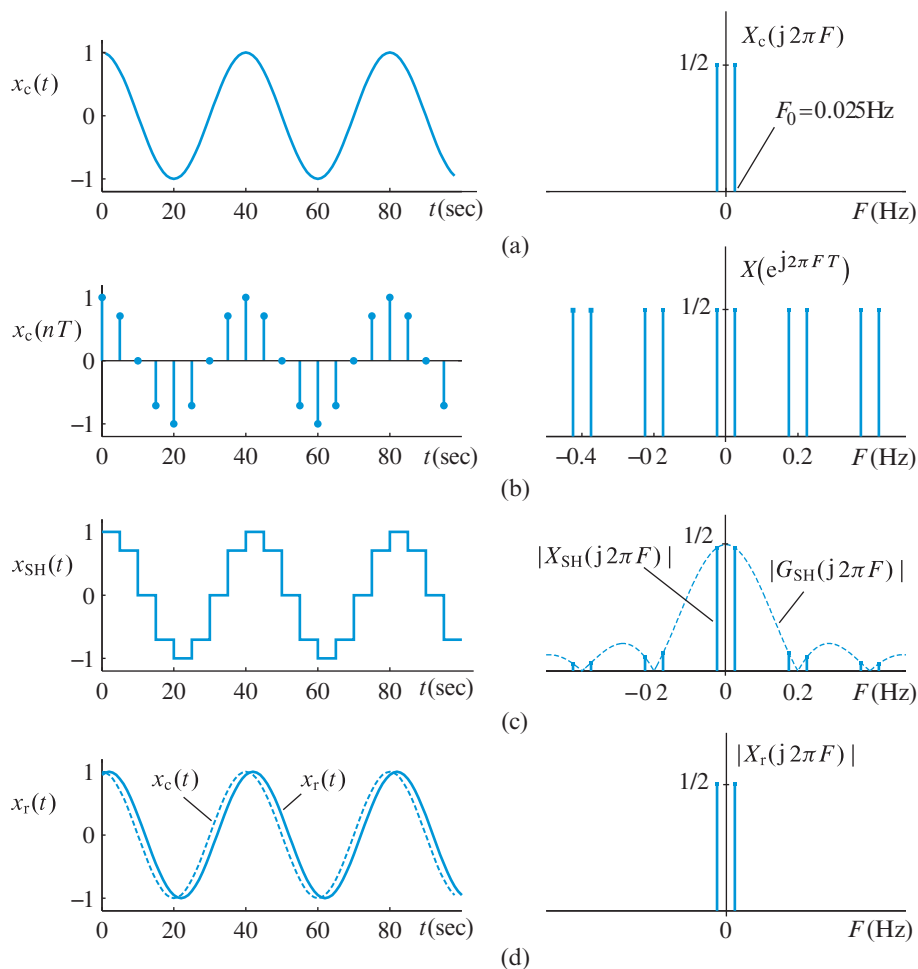
Figure 6.28 Frequency domain characteristics of S/H reconstruction. $F_s = 0.2$ Hz.

MATLAB audio DAC functions We have previously discussed the `sound` function that can play a digitized signal as an audio sound through computer speakers which are connected to audio hardware that performs the practical DAC. Similarly, the `wavplay(x,Fs)` function plays the audio signal stored in the vector `x` on a Microsoft Windows PC-based audio output device at an integer sampling rate of `Fs` Hz, the default being 11025 Hz. On other operating systems, the `player = audioplayer(x,Fs)` function can be used which creates an object `player` that can be played through audio output devices using the `play(player)` function.

6.6   Sampling of bandpass signals

Up to this point we have studied sampling of lowpass signals, that is, signals with spectra bandlimited around the zero frequency. Now we turn our attention to sampling of

bandlimited signals whose spectra are concentrated around a much higher frequency. Let $x_c(t)$ be a real-valued signal that is bandlimited to the range (F_L, F_H)

$$X_c(j\Omega) = \begin{cases} 0, & |\Omega| \leq \Omega_L = 2\pi F_L \\ 0, & |\Omega| \geq \Omega_H = 2\pi F_H \end{cases} \tag{6.68}$$

where $0 < \Omega_L < \Omega_H < \infty$. We call this a *bandpass* signal with center frequency $\Omega_C = 2\pi F_C = (\Omega_L + \Omega_H)/2$ and bandwidth (Hz)

$$B \triangleq F_H - F_L = (\Omega_H - \Omega_L)/2\pi. \tag{6.69}$$

Since $x_c(t)$ is real-valued, the function $X_c(j\Omega)$ has even magnitude and odd phase about $\Omega = 0$. Bandpass signals appear frequently in communication and radar systems, which use modulation techniques to transmit the signals of interest with electromagnetic waves. In such applications, typically, the center frequency is many times larger than the bandwidth. In this section we discuss bandpass signal sampling using a uniform sequence of samples $x[n] = x_c(nT)$, $-\infty < n < \infty$. Other approaches based on quadrature modulation techniques or second-order sampling are discussed in Vaughan *et al.* (1991) and Proakis and Manolakis (2007).

Since the highest frequency in the bandpass signal (6.68) is F_H, according to the theory of Section 6.1, a sampling rate of $F_s \geq 2F_H$ is adequate to sample $x_c(t)$ without causing any aliasing distortion. In this section, we show that a sampling rate within the range

$$2B \leq F_s \leq 4B \tag{6.70}$$

is sufficient to reconstruct a bandpass signal $x_c(t)$ from its samples without aliasing. The minimum sampling rate of $2B$ is adequate under the condition that F_H/B is an integer.

6.6.1 Integer band positioning

We first consider the special case where F_H is an integer multiple of the bandwidth B, that is

$$F_H = K(F_H - F_L) = KB, \tag{6.71}$$

where K is an integer. The spectrum of the sampled signal $x[n] = x_c(nT)$ is a scaled periodic repetition of the original bandpass spectrum

$$X(e^{j2\pi FT}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c \left[j2\pi(F - kF_s) \right]. \tag{6.72}$$

This process is illustrated in Figure 6.29 for $K = 3$ (odd band positioning). We emphasize that each band is repeated periodically with period $F_s = 2B$. For clarity, we label the replica of each band by the index k in (6.72). A band is shifted to the right if $k > 0$ and to the left if $k < 0$. Since the shifted replicas do not overlap with the original spectrum, there is no aliasing. We notice that if we multiply $X(e^{j2\pi FT})$ by the Fourier transform $G_r(j2\pi F)$
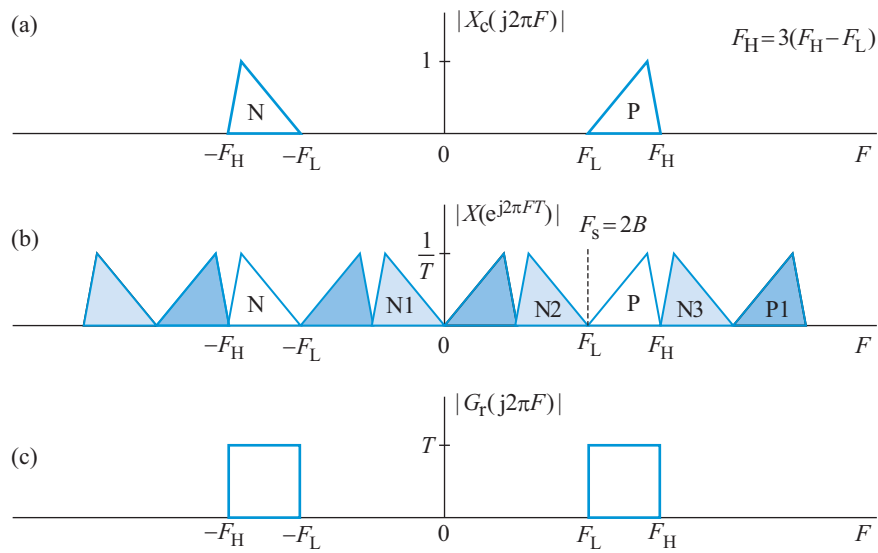
Figure 6.29 Bandpass signal sampling in the frequency domain for odd integer band positioning: (a) original bandpass signal spectrum, (b) spectrum of the sampled sequence, and (c) Fourier transform of the ideal bandpass interpolator.

we can recover $X_c(j2\pi F)$, and hence $x_c(t)$, exactly. The ideal reconstruction process is given by

$$x_c(t) = \sum_{n=-\infty}^{\infty} x_c(nT)g_r(t - nT), \tag{6.73}$$

where $g_r(t)$ is the *modulated* ideal bandlimited interpolation function (more details are provided in Tutorial Problem 15):

$$g_r(t) = \frac{\sin(\pi Bt)}{\pi Bt} \cos(2\pi F_C t), \tag{6.74}$$

where $F_C = (F_H - F_L)/2$. Figure 6.30 illustrates that perfect reconstruction is possible for $K = 4$ (even band positioning). We note that even values of K result in an inverted basedband spectrum; this may cause inconvenience in some practical applications (see Tutorial Problem 16). In conclusion, *a sampling rate of $F_s = 2(F_H - F_L)$ is adequate for alias-free sampling of a bandpass signal if the ratio $K = F_H/(F_H - F_L)$ is exactly an integer.*

With proper choice of F_C in (6.74) we can reconstruct a bandpass signal at a lower center (or intermediate) frequency, a process known as *downconversion*. Traditional radio and radar receivers obtain the equivalent (same information) baseband ($|F| < F_s/2$) signal by analog techniques; the resulting lowpass signal is then sampled for further digital processing. The current trend is to sample the analog bandpass signal as close to the antenna as possible and then use digital techniques for all subsequent processing.
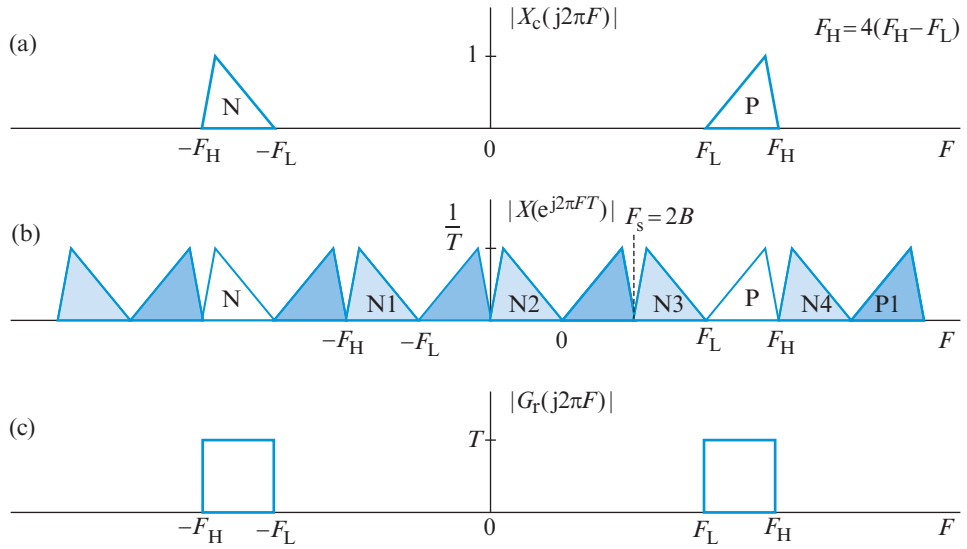
Figure 6.30 Bandpass signal sampling in the frequency domain for even integer band positioning: (a) original bandpass signal spectrum, (b) spectrum of the sampled sequence, and (c) Fourier transform of the ideal bandpass interpolator. Note the reversal of the original spectrum in the baseband region.

6.6.2   Arbitrary band positioning

If F_H is not an integer multiple of bandwidth $B = F_H - F_L$, a more general analysis is required. To this end, we consider the bandpass signal in Figure 6.31(a) and we use the symbols N and P to designate the negative and positive parts of its spectrum. These parts are repeated periodically, with period F_s, after sampling. If we choose the minimum value of F_s to satisfy $F_s \geq 2B$, then the shifted copies of P do not overlap; furthermore, the space between two P replicas can fit a copy of N without overlap. We next note that shifting N to the left cannot cause any overlap. However, shifting N to the right might cause an overlap with P. The $(k-1)$th replica of N has its right end located at a frequency F such that $F - (k-1)F_s = -F_L$ or $F = (k-1)F_s - F_L$. Similarly, the left end of the kth replica is located at a frequency F such that $F - kF_s = -F_H$ or $F = kF_s - F_H$. From Figure 6.31(b) it is clear that to avoid overlap it is necessary that

$$(k-1)F_s - F_L \leq F_L, \tag{6.75}$$

$$kF_s - F_H \geq F_H. \tag{6.76}$$

Combining these two inequalities leads to the following condition

$$\frac{2F_H}{k} \leq F_s \leq \frac{2F_L}{k-1}. \tag{6.77}$$
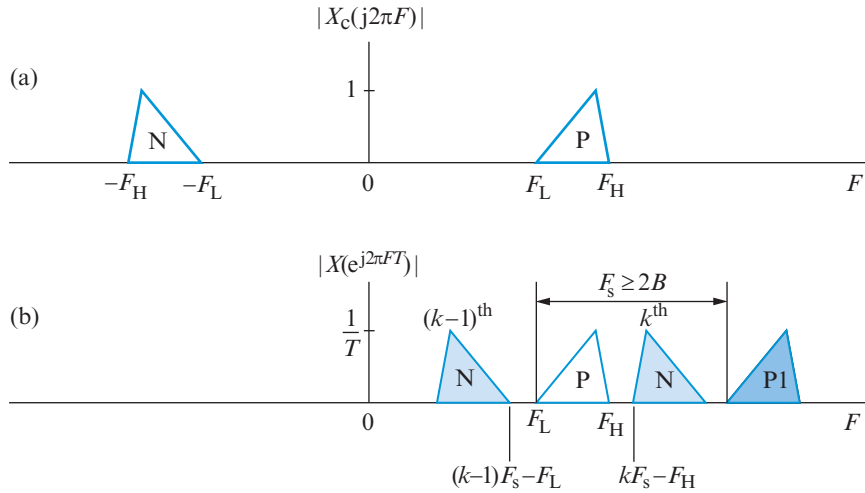
Figure 6.31 (a) Spectrum of a bandpass signal with arbitrary band positioning. (b) Spectra of the negative band (N) shifted by $(k - 1)F_s$ Hz and kF_s Hz, and spectrum of the positive band (P) shifted by F_s Hz.

From the symmetry of the original spectrum and the symmetry of the shifting involved, it is clear that the same constraints assure that there will be no overlap on N. To determine the allowed values of k, we rewrite (6.77) as

$$F_s \geq \frac{2F_H}{k} = 2B \left(\frac{F_H}{B} \right) \left(\frac{1}{k} \right).$$

Since $F_s \geq 2B$, the last relation implies that $k \leq F_H/B$. Since k is an integer, the allowed values are

$$1 \leq k \leq \left\lfloor \frac{F_H}{B} \right\rfloor, \tag{6.78}$$

where $\lfloor x \rfloor$ denotes the largest integer not exceeding x. Using (6.69), conditions (6.77) can be expressed, in terms of normalized variables, as follows:

$$\frac{2}{k} \left(\frac{F_H}{B} \right) \leq \frac{F_s}{B} \leq \frac{2}{k - 1} \left(\frac{F_H}{B} - 1 \right). \tag{6.79}$$

Figure 6.32 shows a graphical representation of (6.79) for several values of k. The unshaded regions are the regions where the constraints are satisfied, while in the shaded regions the constraints are not satisfied, and overlap (resulting in aliasing) will occur. The sawtooth line shows the locus of points with the minimum sampling rate

$$\min F_s = 2B \left(\frac{F_H}{B} \right) \bigg/ \left\lfloor \frac{F_H}{B} \right\rfloor = 2F_H / \lfloor F_H/B \rfloor, \tag{6.80}$$

which is obtained from the left inequality in (6.79) using the largest value of k. Clearly, the minimum rate of $2B$ is attained when the ratio F_H/B is an integer. Note that values of F_H slightly less than kB lead to worst-case results.
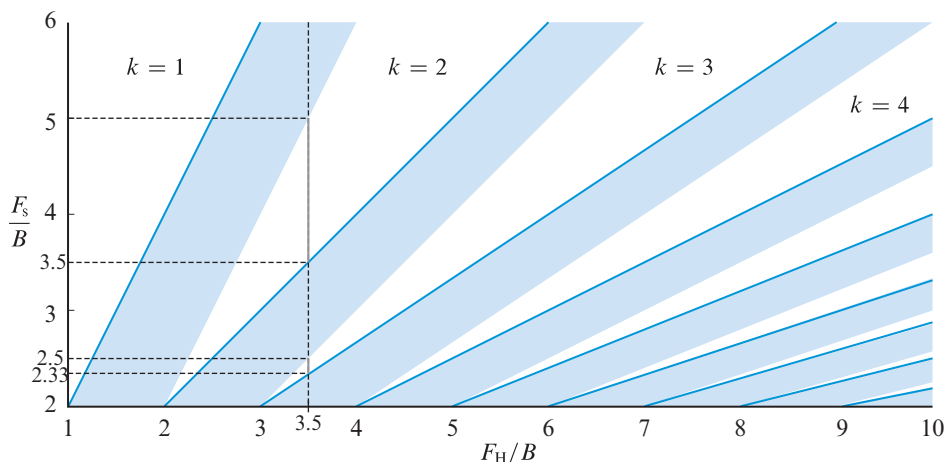
Figure 6.32 Minimum (solid sawtooth line) and permissible sampling rates (white wedges) for bandpass signal sampling.

Example 6.8

As an example, consider a bandpass signal with $F_L = 2.5$ kHz and $F_H = 3.5$ kHz. In this case, $B = F_H - F_L = 1$ kHz, $F_H/B = 3.5$, and $1 \le k \le 3$. Then from (6.80) and Figure 6.32 we see that the minimum sampling rate is $F_s = 2.33$ kHz, corresponding to $k = 3$. From (6.77), the allowable ranges of sampling rate are 2.33 kHz $\le F_s \le 2.5$ kHz for $k = 3$, 3.5 kHz $\le F_s \le 5$ kHz for $k = 2$, and $F_s \ge 2F_H = 7$ kHz for $k = 1$. This is shown in Figure 6.32 in which a dashed vertical line is drawn at $F_H/B = 3.5$ and the allowable ranges are shown as solid lines. In the last $k = 1$ region (not shown) the sampling rates correspond to a lowpass rather than a bandpass signal. ∎

6.6.3   Creating integer band positioning with guard bands

When F_H/B is not an integer, instead of selecting the sampling frequency according to (6.77), we can artificially extend the bandwidth of $x_c(t)$ or change the center frequency to achieve integer band positioning. This approach provides only the minimum allowable sampling frequency (6.80). For example, we can extend the lower band edge F_L to F'_L, such that

$$F'_L \le F_L, \tag{6.81}$$

$$F_H = k(F_H - F'_L) \triangleq kB', \tag{6.82}$$

where k is an integer. Solving (6.82) for F'_L and using (6.81) we obtain

$$F'_L = \left(\frac{k-1}{k}\right) F_H \le F_L. \tag{6.83}$$

The last inequality leads to an expression for finding the value of k:

$$k \le \frac{F_H}{F_H - F_L} = \frac{F_H}{B} \quad \text{or} \quad k = \left\lfloor \frac{F_H}{B} \right\rfloor. \tag{6.84}$$

Therefore, the allowable sampling rate is $F_s = 2B'$ or equivalently

$$F_s = 2F_H \Big/ \left\lfloor \frac{F_H}{B} \right\rfloor, \tag{6.85}$$

which is identical to the minimum sampling frequency provided by (6.80).

Example 6.9

As an example, consider a bandpass signal with $F_L = 22$ kHz and $F_H = 26$ kHz. In this case, $B = F_H - F_L = 4$ kHz and $F_H/B = 6.5$. From (6.84) $k = 6$ and hence $F_L = \frac{5}{6}F_H = 21.6667$ Hz. Therefore, the allowable sampling rate is $F_s = 2B' = 2(F_H - F_L') = 8.6667$ kHz. ∎

This idea can be used to create guard bands, on both sides of the spectrum, to protect from aliasing due to practical limitations. This procedure, which essentially moves the sampling frequency away from the tips of the wedges of Figure 6.32, is discussed in Tutorial Problem 17.

6.7   Image sampling and reconstruction
....

In digital image processing systems we deal with an array of numbers, say $s[m, n]$, obtained by sampling a physical image $s_c(x, y)$ on a rectangular grid of points $(m\Delta x, n\Delta y)$, that is,

$$s[m, n] \triangleq s_c(m\Delta x, n\Delta y), \tag{6.86}$$

where $(\Delta x, \Delta y)$ is the spacing of the grid. After processing, we obtain another array of numbers $v[m, n]$, which should be used to reconstruct a continuous image $v_r(x, y)$ for viewing.

The two-dimensional, continuous Fourier transform pair, which is a straightforward extension of the one-dimensional CTFT pair, is given by the expressions

$$S_c(F_x, F_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s_c(x, y) e^{-j2\pi(xF_x + yF_y)} \, dx dy, \tag{6.87}$$

$$s_c(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} S_c(F_x, F_y) e^{j2\pi(xF_x + yF_y)} \, dF_x dF_y, \tag{6.88}$$

where F_x and F_y are the *spatial* frequency variables in units of cycles per unit of distance and, for clarity, we have used a less-cumbersome notation $S_c(F_x, F_y) \triangleq S_c(j2\pi F_x, j2\pi F_y)$.
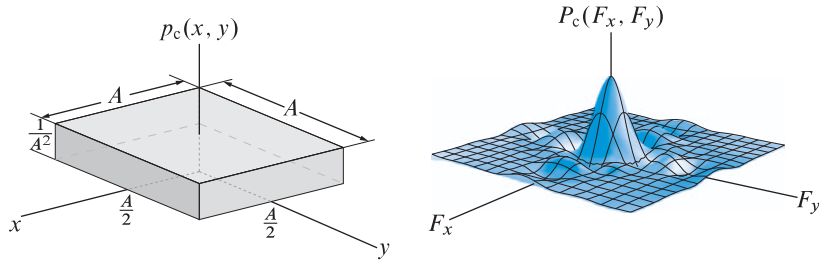
Figure 6.33 A 2-D rectangular function and a section of its spectrum about the origin. Compare with Figure 6.7 which shows the one-dimensional case.

For example, the Fourier transform of the square pulse image

$$p_c(x, y) = \begin{cases} 1/A^2, & |x| < A/2, |y| < A/2 \\ 0, & \text{otherwise} \end{cases} \tag{6.89}$$

is given by (see Tutorial Problem 18)

$$P_c(F_x, F_y) = \frac{\sin(\pi F_x A)}{\pi F_x A} \times \frac{\sin(\pi F_y A)}{\pi F_y A}. \tag{6.90}$$

Figure 6.33 shows the rectangular function and portion of its spectrum about the origin. As in the one-dimensional case, the locations of the zeros in the spectrum are inversely proportional to the values of A.

2-D sampling theorem In a manner similar to the 1-D case, the Fourier transform of the sampled image, $s[m, n]$, is given by

$$\tilde{S}(F_x, F_y) \triangleq \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s[m, n] e^{j2\pi(m\Delta x F_x + n\Delta y F_y)} \tag{6.91}$$

$$= \frac{1}{\Delta x \Delta y} \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} S_c(F_x - kF_{s_x}, F_y - \ell F_{s_y}), \tag{6.92}$$

where $F_{s_x} \triangleq 1/\Delta x$ and $F_{s_y} \triangleq 1/\Delta y$ are the spatial sampling frequencies and where again we have used a less-cumbersome notation $\tilde{S}(F_x, F_y)$ for the 2-D Fourier transform $S(e^{j2\pi F_x}, e^{j2\pi F_y})$ of $s[m, n]$. As can be seen from Figure 6.34, the spectrum of the sampled image is obtained by infinitely repeating the spectrum of the original image over the frequency plane in a rectangular grid with spacing (F_{s_x}, F_{s_y}).

It is clear from Figure 6.34(a) that if the function $s_c(x, y)$ is band-limited, that is,

$$S_c(F_x, F_y) = 0 \quad \text{for } |F_x| > B_x \text{ and } |F_y| > B_y, \tag{6.93}$$

and the spatial sampling frequencies satisfy the conditions

$$F_{s_x} \geq 2B_x \quad \text{and} \quad F_{s_y} \geq 2B_y, \tag{6.94}$$
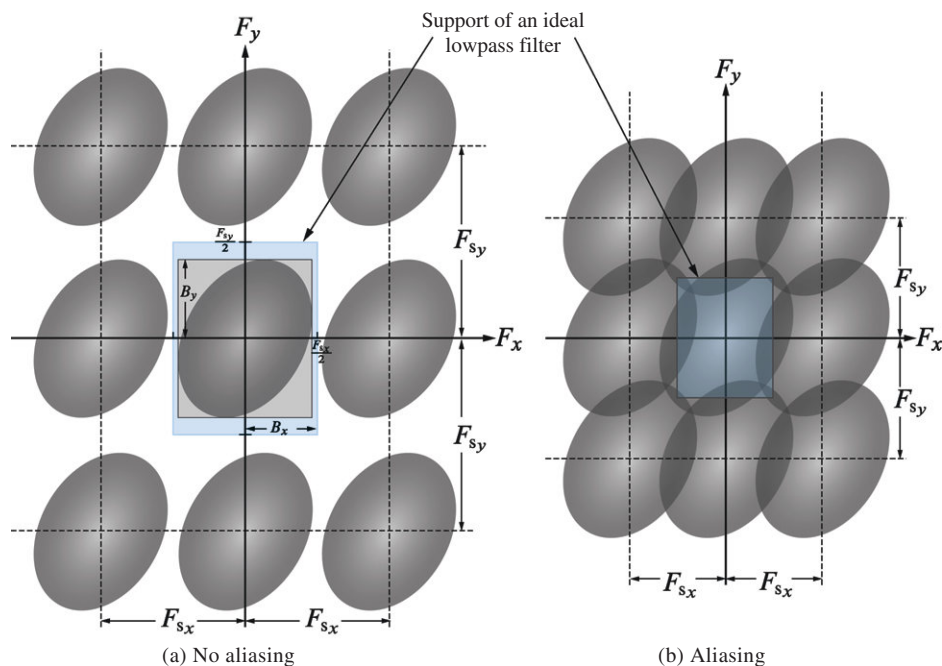
Figure 6.34 2D Sampling in the frequency domain: (a) $F_{s_x} > 2B_x$, $F_{s_y} > 2B_y$ (no aliasing); (b) $F_{s_x} < 2B_x$, $F_{s_y} < 2B_y$ (aliasing).

then there is *no* spectrum overlap. In this case, the spectrum of the original image can be recovered by multiplying $\tilde{S}(F_x, F_y)$ with the reconstruction filter

$$G_r(F_x, F_y) = \begin{cases} \Delta x \Delta y, & |F_x| \le F_{s_x}/2 \text{ and } |F_y| \le F_{s_y}/2 \\ 0. & \text{otherwise} \end{cases} \qquad (6.95)$$

Therefore, the image $s_c(x, y)$ itself can be reconstructed from the samples $s[m, n]$. This sampling condition is the 2-D counterpart of the 1-D sampling theorem discussed in Section 6.1. In physical terms, relations (6.94) require that the sampling period must be equal to or smaller than one-half the period of the finest detail within the image. If equality holds in (6.94), the image is said to be sampled at its Nyquist rate; if F_{s_x} and F_{s_y} are greater than required by the Nyquist criterion, the image is called oversampled; and if the opposite case holds, the image is undersampled.

Visual effects of sampling If there is spectral overlap resulting from undersampling, as indicated by the shaded regions in Figure 6.34(b), spurious spatial frequency components (aliasing error) will be introduced into the reconstruction. For real-world images, most prominent aliased frequency components are near folding frequencies in each dimension (that is, half the sampling frequencies), which then results in a beat pattern effect. In the field of optics, these aliasing effects are known as *Moiré patterns*.
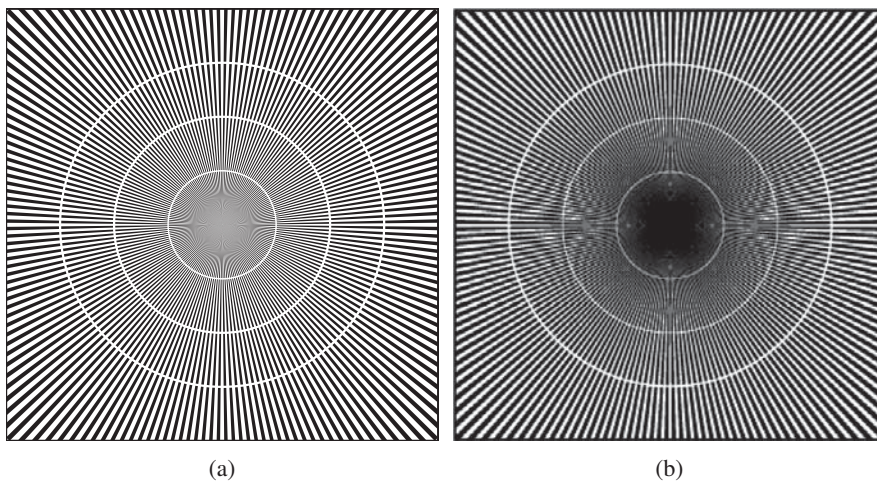
Figure 6.35 Moiré pattern due to aliasing: (a) original pattern, (b) 72 dpi pattern.

Aliasing effects in image sampling are illustrated in Figure 6.35 in which a pattern of concentric circles and radiating line-segments is used to simulate increasing spatial frequencies. Figure 6.35(a) shows an original image pattern that is generated using vector graphics while Figure 6.35(b) shows the same pattern after it has been digitized at 72 dots-per-inch (dpi) rate. The Moiré patterns due to aliasing are clearly evident.

These Moiré pattern effects of aliasing are a source of annoyance in digital images and can be reduced by slightly defocusing the scene to be digitized so that sharp details are smoothed (or equivalently, high frequencies are attenuated) and hence Moiré patterns are broken. As explained in Section 6.5, antialiasing filtering has to be done at the front-end, before the image is sampled. There are no such things as software-based antialiasing filters that can be used once an image has been sampled. Many digital cameras have true antialiasing filtering built-in, either in the lens or on the surface of the sensor itself. However, these filters are not ideal and have some attenuation within their passband that represents a loss of resolution ("detail") of the sampled image. As a result, there is a trade-off between sampled image resolution and aliasing error.

Figure 6.36 shows reduction of aliasing effects due to a smoothing operation in resampled images. An original digital image that is obtained at high sampling rate is shown in 6.36(a). It is resampled to 50% of its original size by deleting every other row and column without pre-smoothing and the result is shown in 6.36(b). The Moiré pattern is not only visible but is also quite annoying. Figure 6.36(c) shows the same image after first smoothing it using a simple averaging filter prior to resampling to 50% of its original size. Now the displayed image is visually pleasing although it has lost its original pattern and is slightly blurred.

Such digital aliasing is also clearly evident in the display of fonts on computer monitors, especially when the clear type or other antialiasing techniques are not activated. Font characters are described by vector graphics equations which are continuous in 2D space. When characters are rendered on a dot selectable device like a monitor, a bit-mapped version is created by sampling and displayed. To a discerning eye, this display appears jagged
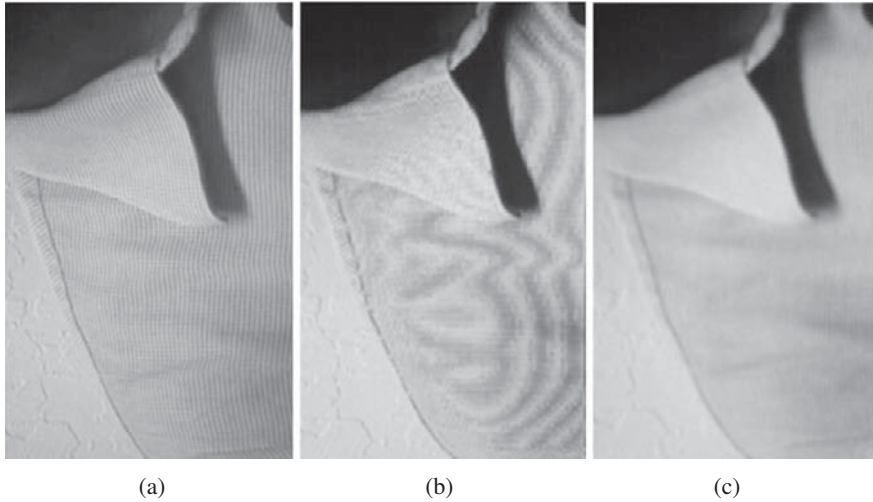
Figure 6.36 Aliasing in resampled images (digital aliasing): (a) original image, (b) resampled without pre-filtering, and (c) resampled with pre-filtering.

and blocky. When antialiasing techniques are turned on, high resolution characters are first blurred (or smoothed) and then subsampled to render a visually pleasing and improved display of characters. This aspect of antialiasing is explored in Review Problem 47.

Ideal reconstruction To achieve perfect image reconstruction in a digital image processing system, it is necessary to bandlimit the image to be sampled, spatially sample the image at the Nyquist or higher rate, and properly interpolate the image samples.

The 2-D counterpart of the 1-D reconstruction formula (6.20) is given by

$$s_{\mathrm{r}}(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} s[m, n] g_{\mathrm{r}}(x - m\Delta x, y - n\Delta y), \tag{6.96}$$

where $g_{\mathrm{r}}(x, y)$ is a 2-D interpolation function. Taking the Fourier transform of (6.96) we obtain the following counterpart of (6.22):

$$S_{\mathrm{r}}(F_x, F_y) = G_{\mathrm{r}}(F_x, F_y)\tilde{S}(F_x, F_y) \tag{6.97}$$

where $\tilde{S}(F_x, F_y)$ is the 2-D Fourier transform of the sequence $s[m, n]$ defined in (6.92). The interpolation function for the ideal reconstructor (6.95) is given by

$$g_{\mathrm{r}}(x, y) = \frac{\sin(\pi F_{\mathrm{s}_x} x)}{\pi F_{\mathrm{s}_x} x} \times \frac{\sin(\pi F_{\mathrm{s}_y} y)}{\pi F_{\mathrm{s}_y} y}, \tag{6.98}$$

which is the 2-D counterpart of the ideal bandlimited interpolator (6.25).

Sampled signal

Sample-and-hold

Signal reconstructed
by video board

Gaussian CRT
spot

Displayed signal
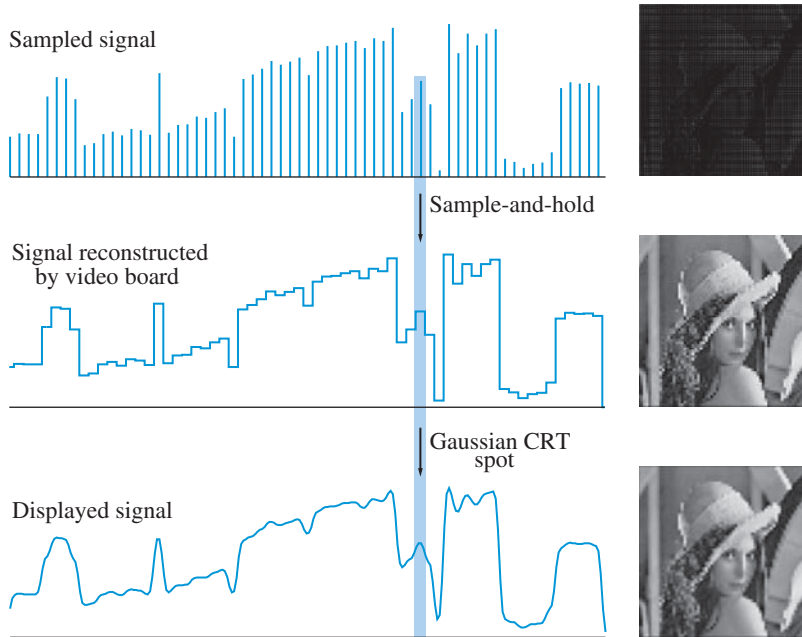
Figure 6.37 Image reconstruction by sample-and-hold and Gaussian cathode-ray tube spot.

Practical reconstruction To understand the function of a reconstruction system, we use (6.91) and (6.97) to express the spectrum of the reconstructed image as

$$S_r(F_x, F_y) = \frac{1}{\Delta x \Delta y} G_r(F_x, F_y) \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} S_c(F_x - kF_{s_x}, F_y - \ell F_{s_y}). \qquad (6.99)$$

Ideally, $G_r(F_x, F_y)$ should select the spectral component for $k = 0$, $\ell = 0$ with uniform attenuation at all spatial frequencies and should reject all other spectral components. These conditions, which are perfectly satisfied by the ideal interpolator (6.98), are impossible to achieve exactly with physical reconstruction systems. An imperfect interpolator may attenuate the frequency components of the zero-order spectra, causing a loss of image resolution, and may allow contributions from higher-order spectra, causing the introduction of high-spatial-frequency artifacts. For example, displaying a digital image in a cathode ray tube display involves two steps. First, the sample values are converted into a continuous video signal using a sample-and-hold circuit. The resulting staircase signal drives the display spot, which has a 2-D Gaussian shape; thus, the display-spot acts as an impulse response of an anti-imaging reconstruction filter. This process is illustrated in Figure 6.37 showing a horizontal scan line of the image.

Learning summary

- Any time a continuous-time signal $x_c(t)$ is uniformly sampled with sampling period $T = 1/F_s$, the spectrum of $x[n] = x_c(nT)$ is obtained by scaling the spectrum of $x_c(t)$ by $1/T$ and putting copies at all integer multiples of F_s

$$X(e^{j2\pi FT}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c[j2\pi(F - kF_s)].$$

- A bandlimited signal with $X_c(j2\pi F) = 0$ for $|F| > F_H$ can be exactly reconstructed from the sequence of samples $x_c(nT)$, where $F_s = 1/T \geq 2F_H$, using the ideal bandlimited interpolation formula

$$x_c(t) = \sum_{n=-\infty}^{\infty} x_c(nT) \frac{\sin[\pi(t - nT)/T]}{\pi(t - nT)/T}.$$

The highest frequency F_H present in $x_c(t)$ is called the Nyquist frequency. The minimum sampling rate, $2F_H$, required to avoid overlap of the repeated copies of $X_c(j2\pi F)$ (aliasing distortion) is known as the Nyquist rate.

- The sampling theorem makes possible the discrete-time processing of continuous-time signals. This is accomplished by sampling a continuous-time signal, applying discrete-time processing algorithms to the sequence of samples, and reconstructing a continuous-time signal from the resulting sequence.

- In practice, the value of each sample is represented numerically using a finite number of bits. This process, known as quantization, destroys information in the same way that adding noise destroys precision. For most signals of practical interest, the signal-to-quantization error ratio increases 6 dB for each additional bit used in the representation.

- A bandpass signal $x_c(t)$, with spectrum $X_c(j2\pi F) = 0$ outside the range $0 < F_L \leq |F| \leq F_H < \infty$, can be reconstructed from its samples without aliasing using a sampling rate in the range $2B \leq F_s \leq 4B$, where $B = F_H - F_L$, instead of the Nyquist rate $2F_H$. The minimum sampling rate of $2B$ will be adequate under the condition that F_H/B is an integer.

- Perfect reconstruction of a bandlimited image $s_c(x, y)$, from a set of samples $s_c(m\Delta x, n\Delta y)$ without aliasing is possible if both the horizontal and vertical sampling frequencies satisfy the sampling theorem.

TERMS AND CONCEPTS

Aliasing distortion A signal distortion caused by overlapping spectra of the signal samples in which frequencies higher than folding frequency are aliased into lower frequencies. Also known as aliasing.

Apparent frequency The lowest frequency of a sinusoid that has exactly the same samples as the input sinusoid and is denoted by F_a.

Arbitrary-band positioning A condition in a bandlimited bandpass signal whose highest bandwidth is not an integer multiple of its bandwidth.

Bandlimited bandpass signal A signal whose spectrum has a finite bandwidth around a center frequency that is much larger than its bandwidth.

Bandlimited lowpass signal A baseband signal whose spectrum is zero above a finite maximum frequency, called bandwidth.

Effective continuous-time filter A continuous-time system realized through the A/D converter – digital filter – D/A filter operation, that is, through a discrete-time processing.

Folding frequency The highest signal frequency that is retained in an input signal after sampling and is equal to half the sampling frequency, or $F_s/2$. All frequencies above $F_s/2$ are aliased into frequencies below $F_s/2$.

Guard band A band of frequencies created when the sampling frequency is greater than the Nyquist rate. It contains no signal spectra.

Ideal digital-to-analog converter (DAC) An idealized operation that reconstructs a continuous-time signal from its samples.

Ideal sampling An idealized operation that periodically picks values of a continuous-time signal resulting in a discrete-time signal. Also called ideal analog-to-digital conversion (ADC) or uniform sampling.

Ideal bandlimited interpolation An idealized reconstruction of a bandlimited signal from its samples using an ideal lowpass filter or using a sinc interpolating function.

Impulse-invariance transformation A procedure of converting a continuous-time filter into an equivalent discrete-time filter so that the shape of the impulse response is preserved.

Integer-band positioning A condition in a bandlimited bandpass signal whose highest bandwidth is an exact integer multiple of its bandwidth.

Interpolation An operation that fills-in values between signal samples according to a predetermined function.

Lowpass antialiasing filter A continuous-time lowpass filter that prevents aliasing by removing frequencies above the folding frequency prior to sampling.

Moiré pattern A visual sampling effect in image sampling created by frequencies close to folding frequency and results in beat-like modulation pattern.

Nyquist frequency The highest frequency in a continuous-time signal. Also called the bandwidth of the signal.

Nyquist rate The minimum sampling rate that avoids aliasing in a bandlimited signal and is equal to twice the Nyquist frequency.

Practical DAC An implementable system that converts samples into a continuous-time signal by implementing a sample-and-hold circuit followed by a carefully designed lowpass post-filter.

Quantization noise An unavoidable error created by the quantization operation in a practical ADC. It is measured via signal-to-quantization noise ratio (SQNR) in dB.

Quantization A process of approximating a continuous range of signal values by a relatively small but finite number of discrete values. Results in an error called quantization noise.

Sample-and-hold circuit A relatively simple circuit in an ADC or DAC that is designed to hold applied input value steady for one sampling interval while the converter performs some operation.

Sampling ADC A practical analog-to-digital converter that has a built-in sample-and-hold circuit.

Sampling frequency A measure of number of samples in one second, expressed in samples per second.

Sampling rate A measure of number of samples in one second, expressed in Hz.

Sampling theorem A fundamental result that states that if a signal contains no frequencies above the highest (or Nyquist) frequency F_H, then it can be completely determined by its samples spaced at-most $1/(2F_H)$ seconds apart.

Talk-through system A simple discrete-time system consisting of an ADC followed by a DAC and used for verifying correct operation of sampling and reconstruction or limitations of A/D or D/A converters.

MATLAB functions and scripts

Name	Description	Page
audiorecorder	Records sound as an object using an audio input device	323
audioplayer	Creates a player object for use with the play function	327
ceil	Quantizes a number to the nearest integer towards ∞	321
fix	Quantizes a number to the nearest integer towards 0	321
floor	Quantizes a number to the nearest integer towards $-\infty$	321
play	Plays a player object through an audio output device	327
round	Quantizes a number to the nearest integer	321
sinc	Computes the $\sin(\pi x)/(\pi x)$ interpolating function	309
sound	Plays sampled signal as a sound through speakers	327
wavrecord	Records sound through mic or input-line (PC only)	323
wavplay	Plays sampled signal as a sound through speakers (PC only)	327

FURTHER READING

1. A detailed treatment of sampling theory, at the same level as in this book, is given in Oppenheim and Schafer (2010) and Proakis and Manolakis (2007). A clear and concise discussion of the sampling theorem, including the original derivation, is given in Shannon (1949).

2. The practical aspects of A/D and D/A conversion are discussed in Hoeschele (1994) and Kester (2005). Williston (2009) provides an introduction to all practical aspects of DSP, including A/D and D/A conversion.

3. Bandpass sampling, which is used extensively in radar and communications, is discussed in Linden (1959), Vaughan *et al.* (1991), and Coulson (1995). A tutorial introduction is given in Proakis and Manolakis (2007).

4. Two-dimensional sampling is discussed in the standard image processing references by Gonzalez and Woods (2008) and Pratt (2007). The implications of sampling in computer graphics are discussed in Foley *et al.* (1995).

Review questions

1. Describe the ideal sampler and explain why it is an ideal operation.
2. In your own words explain how, in an ideal analog-to-digital conversion, the Fourier transform of samples of a continuous-time signal is related to the Fourier transform of the continuous-time signal.
3. What is an aliasing distortion, when does it happen, and how can it be avoided?
4. Explain various frequency terms used in a sampling operation.
5. What two conditions are needed to fulfill requirements of the sampling theorem?
6. Describe the general approach used in the reconstruction of a signal from its samples.
7. What is an ideal digital-to-analog converter and what result does it achieve?
8. Explain in your own words how an ideal bandlimited interpolation in the time domain achieves perfect reconstruction.
9. Describe relationships between the spectra of a continuous-time signal and the discrete-time signal obtained by periodic sampling.
10. We want to down-convert the frequency of a sinusoidal signal to half of its original frequency. Explain how this conversion can be achieved through the use of an ideal ADC followed by an ideal DAC.
11. What is an apparent frequency and how do we compute it and describe it pictorially?
12. When a periodic signal is sampled carefully to obtain a periodic sequence, how are their Fourier series coefficients related?
13. In a discrete-time processing of continuous-time signals, what is an effective continuous-time filter?
14. In which fundamental aspects does a practical sampling and reconstruction differ from the ideal one?
15. What is an antialiasing filter and why is it needed?
16. Describe a sample-and-hold circuit and explain how it helps in the sampling operation.
17. Explain the three broad categories of A/D converters and their applications.
18. What is quantization noise, how is it measured, and how is this measurement related to the number of bits in an A/D converter?
19. What is a practical digital-to-analog converter and what is the preferred method of its implementation?
20. In a practical reconstruction, we need an analog post-filter following a D/A converter. What issues does this filter solve in creating a good reconstruction?
21. In the sampling of bandpass signals, what minimum sampling frequency is needed if bands exhibit integer positioning?
22. In the sampling of bandpass signals, how is minimum sampling frequency determined if bands exhibit arbitrary positioning?
23. What is the visual aliasing effect in image sampling, why does it happen, and what is the best approach to eliminate it?
24. How is practical reconstruction achieved in an image display?

25. What will be the visual effect of coarse image sampling followed by a simple sample-and-hold reconstruction in an image display?
26. From Figure 6.34 notice that there are a number of ideal lowpass filters that could be used to achieve perfect image reconstruction. Is the ideal 2D interpolation function unique? Explain.

Problems

Tutorial problems

1. Signal $x_c(t) = 5\cos(200\pi t + \frac{\pi}{6}) + 4\sin(300\pi t)$ is sampled at a rate of $F_s = 1$ kHz to obtain the discrete-time signal $x[n]$.
   (a) Determine the spectrum $X(e^{j\omega})$ of $x[n]$ and plot its magnitude as a function of ω in $\frac{rad}{sam}$ and as a function of F in Hz. Explain whether the original signal $x_c(t)$ can be recovered from $x[n]$.
   (b) Repeat part (a) for $F_s = 500$ Hz.
   (c) Repeat part (a) for $F_s = 100$ Hz.
   (d) Comment on your results.
2. Signal $x_c(t)$ with spectrum $X_c(j\Omega) = \frac{100}{100+\Omega^2}$ is sampled at a rate of $F_s = 100$ Hz to obtain the discrete-time signal $x[n]$.
   (a) Determine the spectrum $X(e^{j\omega})$ of $x[n]$ and plot it as a function of F in Hz over $-150 \le F \le 150$ Hz.
   (b) Repeat part (a) for $F_s = 50$ Hz.
   (c) Repeat part (a) for $F_s = 25$ Hz.
   (d) For which sampling rate can the signal $X_c(t)$ be reasonably recovered from its samples $x[n]$.
3. Consider a continuous-time signal

$$x_c(t) = 2\cos(10\pi t - 60°) - 3\sin(16\pi t).$$

   It is sampled at $t = 0.05n$ to obtain $x[n]$ which is then applied to an ideal DAC to obtain another continuous-time signal $y_r(t)$.
   (a) Determine $x[n]$ and graph its samples along with the signal $x_c(t)$ in one plot.
   (b) Determine $y_r(t)$ as a sinusoidal signal. Graph and compare it with $x_c(t)$.
   (c) Repeat (a) and (b) for sampling at $t = 0.1n$. Comment on your results.
   (d) Repeat (a) and (b) for sampling at $t = 0.5n$. Comment on your results.
4. In this problem we study the effect of Nyquist-rate sampling of a cosine signal on its ideal reconstruction. Consider a sinusoidal signal $x_c(t) = \cos(2\pi F_0 t + \theta_0)$. It is sampled at a rate of $F_s = 100$ Hz and the resulting samples are applied to an ideal DAC to obtain $y_r(t)$.
   (a) Determine $y_r(t)$ if $F_0 = 10$, 20, and 40 Hz and $\theta_0 = 0$ radians. How does $y_r(t)$ compare with $x_c(t)$?
   (b) Determine $y_r(t)$ if $F_0 = 50$ Hz and $\theta_0 = 0$, $\pi/3$, $\pi/2$, $2\pi/3$, and π radians. How does $y_r(t)$ compare with $x_c(t)$?
   (c) From your observation of results in (b), express $y_r(t)$ in terms of θ_0.

5. In this problem we study the effect of Nyquist-rate sampling of a sine signal on its ideal reconstruction. Consider a continuous-time signal $x_c(t) = \sin(2\pi F_0 t + \theta_0)$. It is sampled at a rate of $F_s = 100$ Hz and the resulting samples are applied to an ideal DAC to obtain $y_r(t)$.

(a) Determine $y_r(t)$ if $F_0 = 10$, 20, and 40 Hz and $\theta_0 = 0$ radians. How does $y_r(t)$ compare with $x_c(t)$?

(b) Determine $y_r(t)$ if $F_0 = 50$ Hz and $\theta_0 = 0$, $\pi/3$, $\pi/2$, $2\pi/3$, and π radians. How does $y_r(t)$ compare with $x_c(t)$?

(c) From your observation of results in (b), express $y_r(t)$ in terms of θ_0.

6. Consider the linear FM signal $x_c(t) = \sin\left(\frac{\pi B t^2}{\tau}\right)$, $0 \leq t \leq \tau$ with $B = 10$ Hz and $\tau = 10$ s. It is applied to the talk-through system of Figure 6.10 with sampling rate of $F_s = B$ Hz to obtain sampled signal $x[n]$ and reconstructed signal $x_r(t)$. Simulate this operation in MATLAB and graph $x_c(t)$, $x[n]$, and $x_r(t)$ in one figure using sub-plots.

7. Let $x_c(t) = e^{-1000|t|}$ with its CTFT

$$X_c(j2\pi F) = \frac{0.002}{1 + (0.002\pi F)^2}.$$

(a) Graph the signal $x_c(t)$, $-5 \leq t$, (ms) ≤ 5 and its CTFT $X_c(j2\pi F)$, $-2 \leq F$, (KHz) ≤ 2.

(b) Sample $x_c(t)$ at $F_s = 1000$ sam/s to obtain $x_1[n]$. Graph the samples $x_1[n]$, $-5 \leq nT$, (ms) ≤ 5 and its DTFT $X_1(e^{j2\pi F})$, $-2 \leq F$, (KHz) ≤ 2. Compare $X_1(e^{j2\pi F})$ with $X_c(j2\pi F)$.

(c) Sample $x_c(t)$ at $F_s = 5000$ sam/s to obtain $x_2[n]$. Graph the samples $x_2[n]$, $-5 \leq nT$, (ms) ≤ 5 and its DTFT $X_2(e^{j2\pi F})$, $-2 \leq F$, (KHz) ≤ 2. Compare $X_2(e^{j2\pi F})$ with $X_c(j2\pi F)$.

(d) Using ideal DAC, obtain the reconstructed signal $y_{r1}(t)$ from samples $x_1[n]$ and graph both signals in one plot over $-5 \leq nT$, (ms) ≤ 5. Compare $y_{r1}(t)$ with $x_c(t)$.

(e) Repeat (d) using $x_2[n]$ to obtain $y_{r2}(t)$.

8. Let $x_c(t)$ be periodic with fundamental period T_0. It is sampled with $F_s = N/T_0$ to produce a periodic sequence $x[n]$ with fundamental period N. Show that the DTFS coefficients, $\tilde{c}_k$, of $x[n]$ are given by the aliasing of the CTFS coefficients, c_k, of $x_c(t)$ with respect to N, that is,

$$\tilde{c}_k = \sum_{\ell=-\infty}^{\infty} c_{k-\ell N}, \ k = 0, \pm 1, \pm 2, \ldots$$

9. Show that (6.43) can be obtained by the convolution of the impulse train $\sum_{n=-\infty}^{\infty} y[n]\delta[t - nT]$ with the impulse response $g_{BL}(t)$ thus proving that (6.43) by itself is not a convolution.

10. In Example 6.6 the continuous-time system (6.55) is converted into the discrete-time system (6.57) by sampling the continuous-time impulse response at F_s Hz. Let $\zeta = 0.3$ and $\Omega_n = 30\pi$ rad/s.

(a) Determine and graph the phase response $\angle H_c(j2\pi F)$.

(b) For $F_s = 50$, 100, and 500 Hz, determine the effective phase responses, $\angle H_{eff}(j2\pi F)$ and graph them in one plot.

(c) Compare your plots in (a) and (b) above and comment on the effect on phase response in discrete-time processing.

11. In digital processing of continuous-time signals, an appropriate digital filter is used after ADC and is followed by a DCA. If ADC and DAC are ideal operations then show that the effective continuous-time system is LTI.

12. An 8-bit ADC has an input analog range of ± 5 volts. The analog input signal is

$$x_c(t) = 2\cos(200\pi t) + 3\sin(500\pi t).$$

The converter supplies data to a computer at a rate of 2048 bits/s. The computer, without processing, supplies these data to an ideal DAC to form the reconstructed signal $y_c(t)$. Determine:

(a) the quantizer resolution (or step),

(b) the SQNR in dB,

(c) the folding frequency and the Nyquist rate,

(d) the reconstructed signal $y_c(t)$.

13. Show that the sample and hold system described by

$$x_{out}(t) = x_{in}(nT); \quad nT \leq t < (n+1)T, \ \forall n$$

is a linear but time-varying system.

14. An economical way to compensate for the droop distortion in S/H DAC is to use an appropriate digital compensation filter prior to DAC.

(a) Determine the frequency response of such an ideal digital filter $H_r(e^{j\omega})$ that will perform an equivalent filtering given by $H_r(j\Omega)$ in (6.67).

(b) One low-order FIR filter suggested in Jackson (1996) is

$$H_{FIR}(z) = -\frac{1}{16} + \frac{9}{8}z^{-1} - \frac{1}{16}z^{-2}.$$

Compare the magnitude response of $H_{FIR}(e^{j\omega})$ with that of $H_r(e^{j\omega})$ above.

(c) Another low-order IIR filter suggested in Jackson (1996) is

$$H_{IIR}(z) = \frac{9}{8 + z^{-1}}.$$

Compare the magnitude response of $H_{IIR}(e^{j\omega})$ with that of $H_r(e^{j\omega})$ above.

15. In sampling a bandpass bandlimited signal $x_c(t)$ of bandwidth B Hz with integer band positioning, the ideal reconstruction filter $G_r(j2\pi F)$ of bandwidth B can reconstruct $X_c(t)$ exactly from its samples. Show that the impulse response of this ideal filter is given by the modulated bandlimited interpolation function in (6.74).

16. The signal $x_c(t)$ be given by

$$x_c(t) = 2\cos(650\pi t) + 4\cos(700\pi t) + 6\cos(750\pi t) + 8\cos(800\pi t).$$

(a) If $x_c(t)$ is sampled at $F_s = 801 \frac{sam}{sec}$, determine and plot the spectrum of the sampled signal as a function of F Hz.

(b) If $x_c(t)$ is sampled at $F_s = 201 \frac{\text{sam}}{\text{sec}}$, determine and plot the spectrum of the sampled signal as a function of F Hz.

(c) What is your observation of the baseband signals after sampling in each of the above two cases.

17. A bandpass signal has $F_L = 105$ Hz and $F_H = 145$ Hz. Determine the minimum sampling rate so as to have a minimum guard band of 10 Hz between two spectrum replicas. Draw the resulting spectrum over $[-150, 150]$ Hz range.

18. Show that the 2D Fourier transform of the square pulse $p_c(x, y)$ in (6.89) is given by $P_c(F_x, F_y)$ in (6.90).

19. A sinusoidal signal $s_c(x, y) = 3\cos(2.4\pi x + 2.6\pi y)$ is sampled at (F_{s_x}, F_{s_y}) frequency to obtain the image $s[m, n]$. An ideal reconstruction is used on $f[m, n]$ to obtain the analog signal $s_r(x, y)$.

(a) If $F_{s_x} = 2$ sam/meter and $F_{s_y} = 3$ sam/meter, determine $s[m, n]$ and $s_r(x, y)$.

(b) If $F_{s_x} = 3$ sam/meter and $F_{s_y} = 2$ sam/meter, determine $s[m, n]$ and $s_r(x, y)$.

(c) If $F_{s_x} = 3$ sam/meter and $F_{s_y} = 3$ sam/meter, determine $s[m, n]$ and $s_r(x, y)$.

20. Image sampling described in (6.86) refers to the acquisition of samples at a point (also known as impulse sampling). Consider a finite-aperture image sampling in which image values over a rectangle of size $\Delta x \times \Delta y$ centered at $(m\Delta x, n\Delta y)$ are averaged to form image samples, where $\Delta x = 1/F_{s_x}$ and $\Delta y = 1/F_{s_y}$. Let $s[m, n]$ represent the point-sampling of a physical image $s_c(x, y)$ as in (6.86) and let $s_{fa}[m, n]$ represent its finite-aperture sampled image.

(a) Express $s_{fa}[m, n]$ in terms of $s[m, n]$ and $s_c(x, y)$.

(b) Determine the 2D Fourier transform of $s_{fa}[m, n]$ in terms of 2D continuous Fourier transform of $s_c(x, y)$. Compare it with that of $s[m, n]$.

(c) Based on your results in part (b) above, comment on the visual quality of finite-aperture sampling in terms of resolution-loss versus aliasing trade-off.

Basic problems

21. Show that a sampler is a memoryless, linear, time-varying system.

22. Signal $x_c(t) = 3 + 2\sin(16\pi t) + 10\cos(24\pi t)$ is sampled at a rate of F_s to obtain the discrete-time signal $x[n]$. For each of the following sampling rates: (i) determine the spectrum $X(e^{j\omega})$ of $x[n]$; (ii) plot its magnitude as a function of ω in $\frac{\text{rad}}{\text{sam}}$ and as a function of F in Hz; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.

(a) $F_s = 30$ Hz, (b) $F_s = 20$ Hz, (c) $F_s = 15$ Hz.

23. Signal $x_c(t) = 5e^{j40t} + 3e^{-j70t}$ is sampled periodically with T s to obtain the discrete-time signal $x[n]$. For each of the following sampling periods in seconds, determine the spectrum $X(e^{j\omega})$ of $x[n]$ and plot its magnitude and phase as a function of ω in $\frac{\text{rad}}{\text{sam}}$. Explain whether $x_c(t)$ can be recovered from $x[n]$.

(a) $T = 0.01$, (b) $T = 0.04$, and (c) $T = 0.1$.

24. Signal $x_c(t)$ with spectrum $X_c(j2\pi F)$ shown below in (a) is sampled at a rate of F_s to obtain the discrete-time signal $x[n]$. For each of the following sampling rates: (i) determine the spectrum $X(e^{j\omega})$ of $x[n]$; (ii) plot it as a function of ω in $\frac{\text{rad}}{\text{sam}}$; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.

(a) $F_s = 10$ Hz, (b) $F_s = 15$ Hz, (c) $F_s = 30$ Hz.

$X_c(j2\pi F)$ $X_c(j2\pi F)$

(a)   (b)
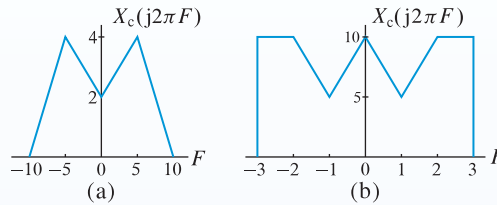
25. Signal $x_c(t)$ with spectrum $X_c(j2\pi F)$ shown above in (b) is periodically sampled at a sampling period of T to obtain the discrete-time signal $x[n]$. For each of the following sampling periods in seconds: (i) determine the spectrum $X(e^{j2\pi F/F_s})$ of $x[n]$; (ii) plot it as a function of F in Hz; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.
(a) $T = 0.2$, (b) $T = 0.25$, (c) $T = 0.5$.

26. Consider a continuous-time signal

$$x_c(t) = 3\cos(2\pi F_1 t + 45°) + 3\sin(2\pi F_2 t).$$

It is sampled at $t = 0.001n$ to obtain $x[n]$ which is then applied to an ideal DAC to obtain another continuous-time signal $y_r(t)$.

(a) For $F_1 = 150$ Hz and $F_2 = 400$ Hz, determine $x[n]$ and graph its samples along with the signal $x_c(t)$ in one plot (choose few cycles of the $x_c(t)$ signal).

(b) Determine $y_r(t)$ for the above $x[n]$ as a sinusoidal signal. Graph and compare it with $x_c(t)$.

(c) Repeat (a) and (b) for $F_1 = 300$ Hz and $F_2 = 700$ Hz. Comment on your results.

27. Telephone speech signals are sampled at 8 ksam/s and are transmitted over a 64 kbits/s digital link. Consider the following signal, which is transmitted over the telephone channel and then reconstructed using the ideal DAC:

$$x_c(t) = 5\sin(10000\pi t - \pi/2).$$

The analog range of the ADC is adjusted to avoid saturation and to minimize the quantization error. Determine:

(a) the quantizer step,

(b) the SQNR in dB,

(c) the folding frequency,

(d) the reconstructed signal $x_r(t)$.

28. A signal has power in the frequency range from 18.1 to 20 kHz. What is the minimum sampling frequency that can be used?

29. A bandpass signal has $F_L = 1002.5$ kHz and $F_H = 1046$ kHz. Determine the minimum sampling rate so as to have a minimum guard band of 4 kHz between two spectrum replicas. Draw the baseband signal spectrum after sampling.

30. Consider the continuous-space signal $s_c(x, y) = 4\cos(4\pi x)\cos(6\pi y)$. It is rectangularly sampled with sampling intervals Δx and Δy respectively, to obtain $s[m, n]$.

(a) An ideal lowpass reconstruction filter with the rectangular bandwidth of $\left(\frac{1}{2\Delta x}, \frac{1}{2\Delta y}\right)$ is used on $s[m, n]$ to obtain the reconstructed signal $s_r(x, y)$. Determine $s_r(x, y)$ if:

i $\Delta x = \Delta y = 0.5$ m,

ii $\Delta x = \Delta y = 0.2$ m.

(b) Now assume that $x_c(x, y)$ is rectangularly sampled with sampling intervals $\Delta x = \Delta y = 0.2$ m. A reconstruction filter has the frequency response of a square display spot of size 0.2×0.2 m^2 but the frequency response is also (essentially) bandlimited to the region $[-5, 5] \times [-5, 5]$ (cycles/m)^2. Determine the reconstructed signal $s_r(x, y)$ and show that it contains a significant amount of Moiré pattern.

31. Generate images which provides Moiré effect in two dimensions. For this effect experiment with various digital frequencies F_x and F_y so that they are close to (but less than) $1/2$.

Assessment problems

32. Signal $X_c(t) = 4\cos(4000\pi t) + 6\cos(6000\pi t) + \sin(14000\pi t)$ is sampled at a rate of F_s to obtain the discrete-time signal $x[n]$. For each of the following sampling rates in kHz: (i) determine the spectrum $X(e^{j\omega})$ of $x[n]$; (ii) plot its magnitude as a function of ω in $\frac{\text{rad}}{\text{sam}}$ and as a function of F in Hz; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.
    (a) $F_s = 20$, (b) $F_s = 10$, (c) $F_s = 5$.

33. Signal $x_c(t) = 8 + 12e^{-j20\pi(t-1)} + 7e^{-j40\pi(t+1)}$ is sampled periodically with the rate F_s Hz to obtain the discrete-time signal $x[n]$. For each of the following sampling rates in Hz, determine the spectrum $X(e^{j2\pi F/F_s})$ of $x[n]$ and plot its magnitude and phase as a function of F in Hz. Explain whether $x_c(t)$ can be recovered from $x[n]$.
    (a) $F_s = 50$, (b) $F_s = 20$, (c) $F_s = 10$.

34. Signal $x_c(t) = 3 + 2\sin(16\pi t) + 10\cos(24\pi t)$ is sampled at a rate of F_s to obtain the discrete-time signal $x[n]$. For each of the following sampling rates: (i) determine the spectrum $X(e^{j\omega})$ of $x[n]$; (ii) plot its magnitude as a function of ω in $\frac{\text{rad}}{\text{sam}}$ and as a function of F in Hz; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.
    (a) $F_s = 30$ Hz, (b) $F_s = 20$ Hz, (c) $F_s = 15$ Hz.

35. Signal $x_c(t) = 4e^{j5\pi t} + 6e^{j12\pi t}$ is sampled periodically with T s to obtain the discrete-time signal $x[n]$. For each of the following sampling periods in seconds, determine the spectrum $X(e^{j\omega})$ of $x[n]$ and plot its magnitude and phase as a function of ω in $\frac{\text{rad}}{\text{sam}}$. Explain whether $x_c(t)$ can be recovered from $x[n]$.
    (a) $T = 0.05$, (b) $T = 0.15$, (c) $T = 0.25$.

36. Signal $x_c(t)$ with spectrum $X_c(j\Omega) = \pi e^{-|\Omega|}$ is sampled periodically at sampling period T to obtain the discrete-time signal $x[n]$. For each of the following sampling periods in seconds: (i) determine the spectrum $X(e^{j\omega})$ of $x[n]$; (ii) plot it as a function of ω; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.
    (a) $T = \pi$, (b) $T = 0.5\pi$, (c) $T = 0.2\pi$.

37. Signal $x_c(t)$ with spectrum $X_c(j2\pi F)$ shown below is sampled at a rate of F_s to obtain the discrete-time signal $x[n]$. For each of the following sampling rates: (i) determine the spectrum $X(e^{j\omega})$ of $x[n]$; (ii) plot it as a function of ω in $\frac{\text{rad}}{\text{sam}}$; and (iii) explain whether $x_c(t)$ can be recovered from $x[n]$.
    (a) $F_s = 6$ Hz, (b) $F_s = 4$ Hz, (c) $F_s = 2$ Hz.
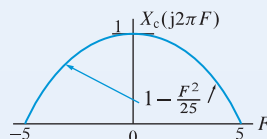
38. Show that an ideal DAC is a noncausal, linear, time-varying system.

39. Consider a continuous-time signal

$$x_c(t) = 10 + 3\sin(20\pi t + \pi/3) + 5\cos(40\pi t).$$

It is sampled at $t = 0.01n$ to obtain $x[n]$, which is then applied to an ideal DAC to obtain another continuous-time signal $y_r(t)$.

(a) Determine $x[n]$ and graph its samples along with the signal $x_c(t)$ in one plot (choose few cycles of the $x_c(t)$ signal).

(b) Determine $y_r(t)$ as a sinusoidal signal. Graph and compare it with $x_c(t)$.

(c) Repeat (a) and (b) for sampling at $t = 0.05n$. Comment on your results.

(d) Repeat (a) and (b) for sampling at $t = 0.1n$. Comment on your results.

40. A real signal has frequencies between 24 and 28 Hz. We want to sample the signal such that there would be two images of the signal between 0 and 24 Hz. Determine the required sampling rate.

41. A bandpass signal has $F_L = 76$ Hz and $F_H = 98$ Hz. Determine the minimum sampling rate so as to have a minimum guard band of 2 Hz between two spectrum replicas. Draw the resulting spectrum of the sampled signal over $[-100, 100]$ Hz range.

42. Consider a continuous-space signal $s_c(x, y) = 2\cos(98\pi x + 198\pi y)$. It is rectangularly sampled with sampling rates F_{s_x} and F_{s_y} in samples/m, respectively, to obtain $s[m, n]$.

(a) An ideal lowpass reconstruction filter with the rectangular bandwidth of $\left(\frac{F_{s_x}}{2}, \frac{F_{s_y}}{2}\right)$ is used on $s[m, n]$ to obtain the reconstructed signal $s_r(x, y)$. Determine $s_r(x, y)$ if:

   i. $F_{s_x} = 25$ samples/m and $F_{s_y} = 50$ samples/m,
   ii. $F_{s_x} = 100$ samples/m and $F_{s_y} = 200$ samples/m.

(b) Now assume that $s_c(x, y)$ is rectangularly sampled with sampling rates of $F_{s_x} = 100$ and $F_{s_y} = 200$ as in subpart ii. above. It is reconstructed using a square display spot of size 0.01×0.005 m^2 centered on samples. Assume that the frequency response of this display spot is bandlimited to the region $[-100, 100] \times [-200, 200]$ (cycles/m)^2.

   i. Determine the frequency response of the reconstruction filter.
   ii. Sketch the 2D Fourier transform of the reconstructed signal $s_r(x, y)$ over the region $[-100, 100] \times [-200, 200]$ (cycles/meter)^2. From the plot determine whether $s_r(x, y)$ contains any significant amount of Moiré pattern.

Review problems

43. The ideal D/A converter described by (6.25) cannot be constructed because it requires an infinite number of past and future samples of $x_c(t)$ for the interpolation. It can be approximated by using only a finite number of terms of (6.25). One method to accomplish this is to use only $(N + 1)$ terms of the sum in (6.25) as

$$\hat{x}_c(t) = \sum_{n=0}^{N} x[n] \frac{\sin\left[\pi(t - nT)/T\right]}{\pi(t - nT)/T}. \tag{6.100}$$

The resulting error $e_1(t) \triangleq x_c(t) - \hat{x}_c(t)$ in the interval $0 \leq t \leq NT$ will be studied in this problem. The objective of studying this approximation error is to obtain a better understanding of ideal D/A conversion. Normally, all necessary observations can be obtained with $3 \leq N \leq 20$.

(a) Develop a MATLAB function `[xhat,t] = DAC1(x,N,T)` that implements (6.100) and computes $\hat{x}_c(t)$ over the interval $0 \leq t \leq NT$ given samples $x[n]$, $0 \leq n \leq N$ and sampling interval T. To obtain a smooth graph of $\hat{x}_c(t)$, choose at least 20 points per sampling interval and provide the resulting time vector `t` as the second output parameter.

(b) Let $x_c(t)$ be a constant equal to 1. Determine and plot $x_c(t)$, $\hat{x}_c(t)$, and $e_1(t)$ for $T = 1$ and $N = 5, 10, 15,$ and 20. Explain the shapes of the plotted signals. How does the error behave as a function of t?

(c) Let the "size" of the error be the maximum value of $|e_1(t)|$ in the middle third of the plotted time interval. Determine and plot the size of error as a function of N for $3 \leq N \leq 20$. Comment on your observations.

(d) Repeat parts (b) and (c) for $x_c = \cos(2\pi F t)$ with $F = 1$ and $T = 0.1$. Compare your results.

(e) Repeat part (d) for various values of F and $T = 1/F_s$. Is the size of the error dependent on the normalized frequency $f = F/F_s$?

(f) Repeat parts (b) and (c) for $x_c = \cos(2\pi F t + \theta)$ with $F = 1$, $T = 0.1$, and various values of θ. Is the size of the error dependent on θ?

44. The ideal D/A converter described by (6.20) and (6.24) cannot be constructed because it requires an infinite number of past and future samples of $x_c(t)$ for the interpolation. It can be approximated by using only a finite number of terms of (6.20). One method to accomplish this is studied in Problem 43. Another method is to use (6.20) but to truncate the interpolation function, $g_r(t)$, as

$$g_K(t) \triangleq g_r(t)\mathcal{R}\left(\frac{t}{KT}\right) = \frac{\sin(\pi t/T)}{\pi t/T}\mathcal{R}\left(\frac{t}{KT}\right), \qquad (6.101)$$

in which

$$\mathcal{R}(\alpha) = \begin{cases} 1, & \text{for } -1 < \alpha < 1 \\ 0, & \text{otherwise} \end{cases} \qquad (6.102)$$

is the rectangular function. Note that $g_K(t) = 0$ for $|t| > KT$. Let the resulting error be

$$e_2(t) \triangleq x_c(t) - \hat{x}_c(t) \triangleq x_c(t) - \sum_{n=-\infty}^{\infty} x[n]g_K(t - nT). \qquad (6.103)$$

The error $e_2(t) = x_c(t) - \hat{x}_c(t)$ in the interval $0 \leq t \leq NT$ will be studied in this problem. The objective of studying this approximation error is to obtain a better understanding of ideal D/A conversion. Normally, all necessary observations can be obtained with $3 \leq K, N \leq 20$.

(a) Develop a MATLAB function `[xhat,t] = DAC2(x,N,K,T)` that implements (6.101), (6.102), and (6.103) and computes $\hat{x}_c(t)$ over the interval $0 \le t \le NT$ given samples $x[n]$, $-K \le n \le N + K$, $3 \le K \le 20$, and sampling interval T. To obtain a smooth graph of $\hat{x}_c(t)$, choose at least 20 points per sampling interval and provide the resulting time vector `t` as the second output parameter.

(b) Let $x_c(t)$ be a constant equal to 1. Determine and plot $x_c(t)$, $\hat{x}_c(t)$, and $e_2(t)$ for $T = 1$ and $N = 5$, 10, 15, and 20. Explain the shapes of the plotted signals. How does the error behave as a function of t?

(c) Let the "size" of the error be the maximum value of $|e_2(t)|$ in the middle third of the plotted time interval. Determine and plot the size of error as a function of N for $3 \le K \le 10$. Comment on your observations.

(d) Repeat parts (b) and (c) for $x_c = \cos(2\pi F t)$ with $F = 1$ and $T = 0.1$. Compare your results.

(e) Repeat part (d) for various values of F and $T = 1/F_s$. Is the size of the error dependent on the normalized frequency $f = F/F_s$?

(f) Repeat parts (b) and (c) for $x_c = \cos(2\pi F t + \theta)$ with $F = 1$, $T = 0.1$, and various values of θ. Is the size of the error dependent on θ?

45. In this problem we will study quantization error distribution due to the `round` function in MATLAB. Let $x[n] = \cos(n/11)$ which is a nonperiodic signal. Hence its samples are continuously distributed over the fundamental period of its envelope. For the following parts use 500 000 signal samples. Each sample of $x[n]$ is quantized to B-bits using a rounding operation to obtain the sequence $x_q[n]$, and let $e[n] = 2^B(x[n]-x_q[n])$ be the normalized quantization error between -0.5 and 0.5.

(a) Develop a MATLAB function
   `[eH,e,eavg,evar] = QuantR(xn,B,N)`
   that computes a normalized histogram of $e[n]$ in array `eH` at `e` bins given `N` samples of `xn` sequence quantized to `B` bits using rounding. The scalars `eavg` and `evar` should contain the computed values of mean and variance of $e[n]$. The sum of `eH` array elements is one in a normalized histogram.

(b) Quantize $x[n]$ to 1, 2, 4, and 6 bits and plot the resulting distributions for $e[n]$. Comment on your plots.

(c) Repeat part (a) for $x[n] = \frac{1}{2}[\cos(n/11) + \cos(n/17) + \sin(n/31)]$.

(d) Repeat part (a) for $x[n]$ obtained using the `rand` function.

46. In this problem we will study quantization error distribution due to the truncation `fix` function in MATLAB. Let $x[n] = \cos(n/11)$ which is a nonperiodic signal. Hence its samples are continuously distributed over the fundamental period of its envelope. For the following parts use 500 000 signal samples. Each sample of $x[n]$ is quantized to B-bits using truncation to obtain the sequence $x_q[n]$ and let $e[n] = 2^B(x[n] - x_q[n])$ be the normalized quantization error between -0.5 and 0.5.

(a) Develop a MATLAB function
   `[eH,e,eavg,evar] = QuantF(xn,B,N)`
   that computes a normalized histogram of $e[n]$ in array `eH` at `e` bins given `N` samples of `xn` sequence quantized to `B` bits using truncation. The scalars `eavg` and `evar` should contain the computed values of mean and variance of $e[n]$. The sum of `eH` array elements is one in a normalized histogram.

(b) Quantize $x[n]$ to 1, 2, 4, and 6 bits and plot the resulting distributions for $e[n]$. Comment on your plots.

(c) Repeat part (a) for $x[n] = \frac{1}{2}[\cos(n/11) + \cos(n/17) + \sin(n/31)]$.

(d) Repeat part (a) for $x[n]$ obtained using the rand function.

47. This problem uses a 100×300 image file containing letters "DSP" and is available at the book website as dsp.png file. Access this file in MATLAB and store it as a variable xc.

(a) Sample xc by taking every 10th pixel horizontally and vertically to create a sampled image x of size 10×30. Rescale x to 100×300 to obtain xs. Display images xc, xs, and x and comment on their appearance.

(b) First blur the image xc using a 5×5 averaging filter to obtain filtered image yc, then sample it by taking every 10th pixel horizontally and vertically to create a sampled image y of size 10×30, and finally rescale y to 100×300 to obtain ys. Display images yc, ys, and y. Compare their appearance with those in (a) and comment on the antialiasing effects on font display.

(c) Experiment on font-antialiasing in (b) above with various blurring filter types and sizes and comment on your observations.

7

The Discrete Fourier Transform

This chapter is primarily concerned with the definition, properties, and applications of the Discrete Fourier Transform (DFT). The DFT provides a unique representation using N coefficients for any sequence of N consecutive samples. The DFT coefficients are related to the DTFS coefficients or to equally spaced samples of the DTFT of the underlying sequences. As a result of these relationships and the existence of efficient algorithms for its computation, the DFT plays a central role in spectral analysis, the implementation of digital filters, and a variety of other signal processing applications.

Study objectives

After studying this chapter you should be able to:

- Understand the meaning and basic properties of DFT and how to use the DFT to compute the DTFS, DTFT, CTFS, and CTFT transforms.
- Understand how to obtain the DFT by sampling the DTFT and the implications of this operation on how accurately the DFT approximates the DTFT and other transforms.
- Understand the symmetry and operational properties of DFT and how to use the property of circular convolution for the computation of linear convolution.
- Understand how to use the DFT to compute the spectrum of continuous-time signals and how to compensate for the effects of windowing the signal to finite-length using the proper window.

7.1   Computational Fourier analysis

The basic premise of Fourier analysis is that any signal can be expressed as a linear super-position, that is, a sum or integral of sinusoidal signals. The exact mathematical form of the representation depends on whether the signal is continuous-time or discrete-time and whether it is periodic or aperiodic (see Table 7.1). Note that in this chapter we use the tilde ($\tilde{\ }$) to emphasize that a sequence or function is periodic. Indeed, $\tilde{x}[n + N] = \tilde{x}[n]$, $\tilde{x}_c(t + T_0) = \tilde{x}_c(t)$, $\tilde{c}_{k+N} = \tilde{c}_k$, and $\tilde{X}(e^{j\Omega+j2\pi}) = \tilde{X}(e^{j\Omega})$.

Careful inspection of Table 7.1 reveals that the equations for the DTFS involve computation of a finite number of coefficients or samples using finite sums of products. Therefore, they can be *exactly* evaluated by numerical computation. All other series or transforms can be computed only *approximately* because they involve infinite summations, integrals, and computation of signals or spectra at a continuous range of values. To illustrate these issues we discuss how we compute the CTFT, DTFT, and CTFS in practice.

Computing the CTFT A simple numerical approximation of $X_c(j\Omega)$ can be obtained by first sampling $x_c(t)$ and then replacing the Fourier integral by a sum

$$X_c(j\Omega) = \int_{-\infty}^{\infty} x_c(t)e^{-j\Omega t}dt \approx \sum_{n=-\infty}^{\infty} x_c(nT)e^{-j\Omega nT}(T) \triangleq \hat{X}_c(j\Omega). \quad (7.1)$$

Table 7.1 Summary of direct and inverse Fourier transforms and the key computational operations required for their evaluation. The presence of an infinite sum or integral prevents exact numerical computation of the corresponding transform

	Direct transform (spectral analysis)	Inverse transform (signal reconstruction)	Exact computation
DTFS	$\tilde{c}_k = \dfrac{1}{N}\sum\limits_{n=0}^{N-1} \tilde{x}[n]e^{-j\frac{2\pi}{N}kn}$ finite summation	$\tilde{x}[n] = \sum\limits_{k=0}^{N-1} \tilde{c}_k e^{j\frac{2\pi}{N}kn}$ finite summation	yes
DTFT	$\tilde{X}(e^{j\Omega}) = \sum\limits_{n=-\infty}^{\infty} x[n]e^{-j\Omega n}$ infinite summation	$x[n] = \dfrac{1}{2\pi}\int_0^{2\pi} \tilde{X}(e^{j\Omega})e^{j\Omega}d\omega$ integration	no
CTFS	$c_k = \dfrac{1}{T_0}\int_0^{T_0} \tilde{x}_c(t)e^{-jk\Omega_0 t}dt$ integration	$\tilde{x}_c(t) = \sum\limits_{k=-\infty}^{\infty} c_k e^{jk\Omega_0 t}$ infinite summation	no
CTFT	$X_c(j\Omega) = \int_{-\infty}^{\infty} x_c(t)e^{-j\Omega t}dt$ integration	$x_c(t) = \dfrac{1}{2\pi}\int_{-\infty}^{\infty} X_c(j\Omega)e^{j\Omega t}d\Omega$ integration	no

7.1 Computational Fourier analysis

The approximation of the integral by a sum, which is based on the definition of Riemann integration, improves as the spacing T between the samples decreases. To gain insight into the nature of this approximation we resort to a signal processing interpretation. We first note that the Riemann sum in (7.1) can be expressed as

$$\hat{X}_c(j\Omega) = T\tilde{X}(e^{j\omega})|_{\omega=\Omega T}, \tag{7.2}$$

where $\tilde{X}(e^{j\omega})$ is the DTFT of $x[n] = x_c(nT)$. Due to this sampling, the "true" spectrum $X_c(j\Omega)$ and the "estimated" spectrum $\tilde{X}(e^{j\omega})$ are related by (see (6.12))

$$\tilde{X}(e^{j\Omega T}) = \frac{1}{T} \sum_{\ell=-\infty}^{\infty} X_c\left(j\Omega - j\ell\frac{2\pi}{T}\right). \tag{7.3}$$

A close inspection of (7.1) and (7.3) reveals that the aperiodic spectrum of interest, $X_c(j\Omega)$, is approximated by a periodic spectrum, $\hat{X}_c(j\Omega)$, which is formed by a periodic repetition of $X_c(j\Omega)$. Hence, the approximation provided by (7.1) is meaningful only in the range $|\Omega| < \pi/T$. If $X_c(j\Omega) = 0$ for $|\Omega| > \pi/T$, the periodic repetition of $X_c(j\Omega)$ does not create any aliasing error. In this case, we have

$$X_c(j\Omega) = \begin{cases} T\tilde{X}(e^{j\Omega T}), & |\Omega| < \pi/T \\ 0. & \text{otherwise} \end{cases} \tag{7.4}$$

If there is aliasing, we have $X_c(j\Omega) \approx T\tilde{X}(e^{j\Omega T})$. This approximation improves as $T \to 0$ or equivalently $F_s \to \infty$. Since there is nothing we can do to mitigate the aliasing distortion after sampling, we shall focus on the computation of $\tilde{X}(e^{j\Omega T})$.

Computing the CTFS If we sample a periodic signal $\tilde{x}_c(t + T_0) = \tilde{x}_c(t)$ with sampling period $T = T_0/N$, the resulting signal $\tilde{x}[n] = \tilde{x}_c(nT)$, is a periodic sequence $\tilde{x}[n + N] = \tilde{x}[n]$. Using rectangular integration, the CTFS can be approximated by the finite sum

$$c_k \approx \frac{1}{T_0} \sum_{n=0}^{N-1} \tilde{x}_c(nT)e^{-jk\Omega_0 nT}(T) = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[n]e^{-j\frac{2\pi}{N}kn} = \tilde{c}_k, \tag{7.5}$$

where $\Omega_0 = 2\pi/T_0 = (2\pi/N)T$. Since the interval of integration is finite, the only source of error is aliasing (see Example 6.4). If there is no aliasing error, we have $c_k = \tilde{c}_k$. The case $T \neq T_0/N$, when $\tilde{x}[n] = \tilde{x}_c(nT)$ is not periodic, is discussed in Example 7.9.

Computing the DTFT The DTFT of a sequence $x[n]$ is given by

$$\tilde{X}(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \tag{7.6}$$

There are two major issues with the computation of (7.6). First, the infinite summation has to be evaluated for all nonzero samples of $x[n]$; this would be impossible for

infinite duration signals. In this case we are forced to approximate $\tilde{X}(e^{j\omega})$ with the finite summation

$$\tilde{X}(e^{j\omega}) \approx \sum_{n=0}^{N-1} x[n]e^{-j\omega n} \triangleq \tilde{X}_N(e^{j\omega}). \tag{7.7}$$

This approximation will be reasonable if the values of $x[n]$ outside the interval $0 \leq n \leq N-1$ are either zero or negligibly small. Again, to understand the nature of this approximation we resort to a signal processing interpretation. We note that $\tilde{X}_N(e^{j\omega})$ is *not* the spectrum of $x[n]$, but the spectrum of the sequence

$$x_N[n] \triangleq x[n]p_N[n], \tag{7.8}$$

where $p_N[n]$ is the rectangular pulse sequence

$$p_N[n] \triangleq \begin{cases} 1, & 0 \leq n \leq N-1 \\ 0. & \text{otherwise} \end{cases} \tag{7.9}$$

This *windowing* operation, which extracts a *finite segment* of the signal, has the biggest impact on the accuracy of the estimated spectrum (see Section 7.6).

Second, we can only compute the function $\tilde{X}_N(e^{j\omega})$ of the continuous variable ω at a finite set of frequencies $0 \leq \omega_k < 2\pi$, $0 \leq k \leq K-1$. If we define the quantities

$$X[k] \triangleq \tilde{X}_N(e^{j\omega_k}), \quad k = 0, 1, \ldots, K-1 \tag{7.10}$$

we can express the set of equations (7.7) in matrix form as

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[K-1] \end{bmatrix} = \begin{bmatrix} e^{j\omega_0 0} & e^{j\omega_0 1} & \cdots & e^{j\omega_0(N-1)} \\ e^{j\omega_1 0} & e^{j\omega_1 1} & \cdots & e^{j\omega_1(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ e^{j\omega_{K-1} 0} & e^{j\omega_{K-1} 1} & \cdots & e^{j\omega_{K-1}(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}, \tag{7.11}$$

or in more compact form as

$$X = Wx. \tag{7.12}$$

If $K = N$ we have a linear system of N equations with N unknowns. In this case, we can exactly determine the values of $x[0], \ldots, x[N-1]$ from the spectral samples $X[0], \ldots, X[N-1]$ by solving a linear system of equations (7.12). If the $N \times N$ matrix W is nonsingular, its inverse exists, and the solution is formally expressed as

$$x = W^{-1}X. \tag{7.13}$$

Solving the linear system (7.12) requires in the order of N^3 floating point operations. The solution of (7.13) can be simplified if we use N equally spaced frequencies

$$\omega_k = \frac{2\pi}{N}k. \quad k = 0, 1, \ldots, N-1 \tag{7.14}$$

Indeed, substituting (7.14) into (7.7) yields

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{K}kn}, \quad k = 0, 1, \ldots, N-1 \tag{7.15}$$

If we set $\tilde{x}[n] = x[n]$ and we compare (7.15) with the DTFS formula, we obtain

$$\tilde{c}_k = \frac{1}{N} X[k]. \tag{7.16}$$

Substituting (7.16) into the IDTFS formula yields

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{K}kn}, \quad n = 0, 1, \ldots, N-1 \tag{7.17}$$

which provides an explicit computation of (7.13) without matrix inversion.

Summary Equations (7.15) and (7.17) provide the basis for computational Fourier analysis. Given a set of samples $x[n]$, $0 \le n \le N-1$, we can use (7.15) to compute a set of coefficients $X[k]$, $0 \le k \le N-1$. The N signal samples can always be exactly recovered from the N coefficients using (7.17). However, the meaning or interpretation of the coefficients depends on the "origin" of the N signal samples:

- If $x[n]$ has finite-length N, that is, $x[n] = 0$ outside the range $0 \le n \le N-1$, then we have (see (7.6))

$$X[k] = \tilde{X}\big(e^{j\frac{2\pi}{N}k}\big). \tag{7.18}$$

- If $x[n]$ has finite-length $L > N$ or infinite length, then we have (see (7.7))

$$X[k] = \tilde{X}_N\big(e^{j\frac{2\pi}{N}k}\big). \tag{7.19}$$

- If $x[n]$, $0 \le n \le N-1$ is a period from a periodic sequence, then (see (7.5))

$$X[k] = N\tilde{c}_k. \tag{7.20}$$

We conclude that formulas (7.15) and (7.17) can be used to compute, either exactly or approximately, all Fourier decompositions (DTFS, CTFS, DTFT, CTFT). This is one of the main reasons for defining the pair of reversible operations (7.15) and (7.17) as a transform in its own right. This new transform, which is known as the *Discrete Fourier Transform*, is discussed in the next section.

7.2   The Discrete Fourier Transform (DFT)

In this section we develop the DFT as a transform in its own right using both an algebraic and a matrix formulation. These equivalent approaches, besides offering additional insight,

demonstrate that the DFT is a powerful reversible operation for *finite segments* of discrete-time sequences. The DFT and its inverse can be efficiently computed using a family of algorithms collectively known as the Fast Fourier Transform or FFT (see Chapter 8). The discovery of FFT algorithms established the DFT as one of the fundamental tools in digital signal processing.

7.2.1   Algebraic formulation of DFT

Given N consecutive samples $x[n]$, $0 \leq n \leq N - 1$ of a periodic or aperiodic sequence, the N-point *Discrete Fourier Transform (DFT)* $X[k]$, $0 \leq k \leq N - 1$ is defined by

$$X[k] \triangleq \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}. \tag{7.21}$$

Given N DFT coefficients $X[k]$, $0 \leq k \leq N - 1$, the N sample values $x[n]$, $0 \leq n \leq N - 1$ can be recovered using the N-point *inverse DFT (IDFT)* given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] e^{j\frac{2\pi}{N}kn}. \tag{7.22}$$

Since $X[k]$ is a function of the discrete frequency index k, which corresponds to a discrete set of frequencies $\omega_k = (2\pi/N)$, $k = 0, 1, \ldots, N - 1$, we say that (7.21) and (7.22) form a *Discrete Fourier Transform (DFT) pair*. This name is used to emphasize the fact that we have a Fourier-like transform which is discrete *both* in time and frequency. For convenience in notation, we often express the DFT equations in the following form

$$
\begin{array}{cc}
\textbf{Analysis equation} & \textbf{Synthesis equation} \\
\end{array}
$$

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \xrightleftharpoons[N]{\text{DFT}} x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}, \tag{7.23}$$

where the complex quantity W_N, known as the *twiddle factor*, is defined by

$$W_N \triangleq e^{-j\frac{2\pi}{N}}. \tag{7.24}$$

The DFT uniquely describes N consecutive samples $x[0], x[1], \ldots, x[N - 1]$ of a sequence in terms of N transform coefficients $X[0], X[1], \ldots, X[N - 1]$. To show that (7.23) is a valid reversible transform, we change the index of summation in the IDFT sum from k to m, and we insert the result into the DFT formula. This yields

$$X[k] = \sum_{n=0}^{N-1} \frac{1}{N} \sum_{m=0}^{N-1} X[m] W_N^{-mn} W_N^{kn}$$

$$= \sum_{m=0}^{N-1} X[m] \frac{1}{N} \sum_{n=0}^{N-1} W_N^{(k-m)n}. \tag{7.25}$$

To proceed, we need the following identity, which expresses the orthogonality of complex exponentials (see (4.22) for a proof)

$$\frac{1}{N} \sum_{n=0}^{N-1} W_N^{(k-m)n} = \frac{1}{N} \sum_{n=0}^{N-1} e^{j\frac{2\pi}{N}(k-m)n} = \begin{cases} 1, & k-m = rN \\ 0. & \text{otherwise} \end{cases} \tag{7.26}$$

From (7.26) we see that the only term in the sum on m in (7.25) that is nonzero is the term corresponding to $m = k$ for any k in the range $0 \le k \le N - 1$. This establishes that the right hand side of (7.25) is in fact equal to $X[k]$.

Roots of unity Consider the N complex numbers W_N^{-k}, for $k = 0, 1, \dots, N - 1$. They are called the Nth roots of unity because they satisfy the equation

$$\left(W_N^{-k}\right)^N = \left(e^{j\frac{2\pi}{N}k}\right)^N = e^{j2\pi k} = 1, \tag{7.27}$$

and therefore are zeros of the polynomial $z^N - 1$. Since $e^{j\frac{2\pi}{N}k}$ is periodic with period N, we can choose W_N^{-k} for any set of N consecutive values of k as the roots of unity. We usually choose the values W_N^{-k}, $k = 0, 1, \dots, N - 1$. As shown in Figure 7.1 these roots are complex numbers equally spaced around the unit circle. The angular spacing between them is $2\pi/N$ radians. This diagram explains why the sum $W_8^{0m} + W_8^{1m} + \cdots + W_8^{7m}$ equals eight for $m = 0$ and zero for $m = 1$.
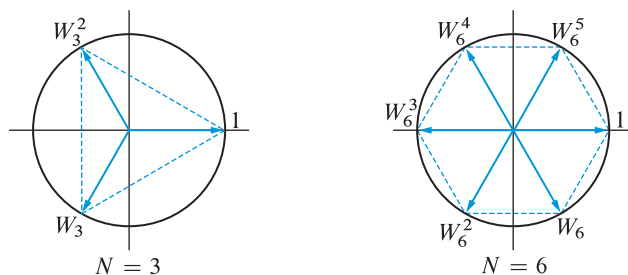
Figure 7.1 Representation of the Nth roots of unity, that is, the solutions of the polynomial equation $z^N = 1$ for $N = 3$ and $N = 6$.

Example 7.1   DFT of unit sample pulse

The N-point DFT of the unit sample sequence is given by

$$X[k] = \sum_{n=0}^{N-1} \delta[n]W_N^{kn} = W_N^0 = 1. \quad 0 \le k \le N-1 \tag{7.28}$$

Substituting (7.28) in the inverse DFT formula, we have

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} W_N^{-kn} = \frac{1}{N}\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}kn}. \quad 0 \le n \le N-1 \tag{7.29}$$

From the orthogonality relation (7.26) we conclude that $x[0] = 1$ and $x[n] = 0$ for $1 \le n \le N-1$, as expected. We stress that the uniqueness of DFT allows only the recovery of the N samples used in the computation of the N DFT coefficients.   ∎

7.2.2   Matrix formulation of DFT

The N equations for the DFT coefficients can be expressed in matrix form as

$$\begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & W_N & \cdots & W_N^{N-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & \cdots & W_N^{(N-1)(N-1)} \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix}, \tag{7.30}$$

or in compact form as

$$X_N = W_N x_N. \quad \text{(DFT)} \tag{7.31}$$

The elements of the *DFT matrix* W_N, the signal vector x_N, and the DFT coefficient vector X_N, are easily determined by comparing (7.30) to (7.21). Equation (7.30) shows that the DFT can be viewed as a matrix operator or as a linear transformation from a signal vector x_N to a DFT coefficient vector X_N. We note that the DFT matrix is symmetric, that is, $W_N = W_N^T$. If the inverse of W_N exists, we obtain

$$x_N = W_N^{-1} X_N. \tag{7.32}$$

This solution in (7.32) provides an inefficient formula for the computation of IDFT.

Let w_k denote the kth column of the DFT matrix, that is,

$$w_k \triangleq \begin{bmatrix} 1 & W_N^k & \cdots & W_N^{(N-1)k} \end{bmatrix}^T. \tag{7.33}$$

The conjugate transpose of w_k, denoted by w_k^H, is obtained by taking the complex conjugate of each element of w_k^T. Using (7.26) we can show that the inner product of the kth and mth columns is given by

$$w_k^H w_m = \sum_{n=0}^{N-1} W_N^{-kn} W_N^{mn} = \begin{cases} N, & k = m \\ 0. & k \neq m \end{cases} \tag{7.34}$$

Therefore, the columns of the DFT matrix are mutually orthogonal vectors. We can easily see that the elements of matrix $C = W_N^H W_N$ are given by $c_{km} = w_k^H w_m$. This leads to the identity

$$W_N^H W_N = N I_N, \tag{7.35}$$

where I_N is the $N \times N$ identity matrix. Multiplying both sides of (7.35) from the right by W_N^{-1}, and recalling that W_N is symmetric, we obtain

$$W_N^{-1} = \frac{1}{N} W_N^H = \frac{1}{N} W_N^*. \tag{7.36}$$

Thus, the rows of the inverse matrix are obtained by conjugating the columns of the DFT matrix and then dividing by N. Substituting (7.36) into (7.32) yields the matrix form of the IDFT (7.22):

$$x_N = \frac{1}{N} W_N^H X_N = \frac{1}{N} W_N^* X_N. \quad \text{(IDFT)} \tag{7.37}$$

The normalized matrix $\overline{W}_N \triangleq W_N / \sqrt{N}$ is unitary because it satisfies the condition $\overline{W}_N^H \overline{W}_N = I_N$. The columns of $\overline{W}_N$ are orthonormal vectors because $\bar{w}_k^H \bar{w}_m = \delta[k - m]$, that is, they are mutually orthogonal and have unit length. Additional properties of the DFT matrix are discussed in Tutorial Problem 4.

The IDFT (7.37) can be written in terms of the columns of W_N^* as follows:

$$x_N = \frac{1}{N} \sum_{k=0}^{N-1} X[k] w_k^*. \tag{7.38}$$

This formula provides a decomposition of the signal vector x_N into an orthogonal basis specified by the columns of the DFT matrix. Each basis vector is a finite-length complex exponential sequence; its contribution to the representation is equal to the corresponding DFT coefficient (see Tutorial Problem 6).

Computation of DFT Computing the DFT or its inverse involves a matrix-by-vector multiplication, which requires $O(N^2)$ operations. The computation of the DFT matrix is typically done up-front and the results are stored in memory. Direct computation of W_N in MATLAB can be done by the statement

$$\mathtt{W = exp(-i*(2*pi/N)*(1:N-1)*(1:N-1)')} \tag{7.39}$$

MATLAB uses the function $W = \text{dftmtx}(N)$, which includes the single statement

$$W = \text{fft}(\text{eye}(N)). \tag{7.40}$$

From (7.30) we see that if $x[k] = 1$ and $x[n] = 0$ for all other n, the DFT is equal to the kth column of $\mathbf{W}_N$. The input signal vectors for $k = 0, 1, \ldots, N - 1$ form the columns of the identity matrix $\text{eye}(N)$. Since the MATLAB fft function computes the DFT of each column, the result of (7.40) is the DFT matrix.

Example 7.2

Use (7.31) to determine the DFT coefficients of the four point segment $x[0] = 0, x[1] = 1,$ $x[2] = 2, x[3] = 3$ of a sequence $x[n]$.

Solution We first compute the entries of the matrix $\mathbf{W}_4$ using the property

$$W_N^{k+N} = W_N^k = e^{-j\frac{2\pi}{N}k} = \cos\left(\frac{2\pi}{N}k\right) - j\sin\left(\frac{2\pi}{N}k\right). \tag{7.41}$$

The result is a complex matrix given by

$$\mathbf{W}_4 = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^4 & W_4^6 \\ W_4^0 & W_4^3 & W_4^6 & W_4^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4^1 & W_4^2 & W_4^3 \\ 1 & W_4^2 & W_4^0 & W_4^2 \\ 1 & W_4^3 & W_4^2 & W_4^1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}.$$

The DFT coefficients are evaluated by the matrix-by-vector multiplication (7.30) as

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ -2 + j2 \\ -2 \\ -2 - j2 \end{bmatrix}.$$

In MATLAB these computations are done using the commands:

```
x = [0 1 2 3]'; W = dftmtx(4); X = W*x;
```

The inverse DFT, $x = \text{conj}(W)*X/N$, can also be evaluated using matrix inversion $x = \text{inv}(W)*X$ or by solving the linear system $x = W\backslash X$. ∎

7.2.3   Inherent periodicity of DFT and IDFT

We have shown that the DFT is a finite $N \times N$ linear reversible transformation that relates any N consecutive samples $\{x[0], x[1], \ldots, x[N-1]\}$ of a signal $x[n]$, $-\infty < n < \infty$ to N DFT coefficients $\{X[0], X[1], \ldots, X[N-1]\}$. Clearly, *no* information about the *unused* samples of $x[n]$ can be obtained from the inverse DFT. We can assign values to the unavailable samples, only if we have *additional* information:

- If it is known a priori that $x[n]$ has finite length N, we can assign the values $x[n] = 0$ for $n < 0$ and $n \geq N$.
- If $x[n]$ is known to be periodic with period N, then we can use periodic extension $x[n + mN] = x[n]$ to obtain the values for all n.

We now explain what happens if we allow the indices k and n to take values outside the range $0 \leq k, n \leq N - 1$. The twiddle factor $W_N^{kn} = e^{-j\frac{2\pi}{N}kn}$ is periodic in k and periodic in n with fundamental period N, that is

$$W_N^{(k+N)n} = W_N^{kn} \quad \text{and} \quad W_N^{k(N+n)} = W_N^{kn}. \tag{7.42}$$

This double periodicity has the following fundamental implications:

- If we allow k to take values outside the set $\{0, 1, \ldots, N - 1\}$, the DFT coefficients $X[k]$ will repeat with fundamental period N. This periodic extension can be termed the *Discrete Fourier Series* (DFS) and denoted by $\tilde{X}[k]$ to emphasize its periodic nature, that is,

$$\tilde{X}[k + N] = \tilde{X}[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad \text{for all } k \tag{7.43}$$

where the primary period is equal to the DFT $X[k] = \tilde{X}[k]p_N[k]$. Note carefully that DFS (which is the inherent periodic extension of DFT) is not the same as DTFS. These two quantities, however, are related (see (7.53)).

- If we try to recover $x[n]$ from $X[k]$ using the inverse DFT and we allow n to take values outside the set $\{0, 1, \ldots, N - 1\}$, the values of $x[n]$ will repeat with fundamental period N. This periodic sequence can be termed the *Inverse Discrete Fourier Series* (IDFS)

$$\tilde{x}[n + N] = \tilde{x}[n], \quad \text{for all } n \tag{7.44}$$

where the primary period is equal to the initial segment $x[n] = \tilde{x}[n]p_N[n]$.

These periodicities are an inherent property of DFT and stem from the discrete nature of time and frequency variables. They have fundamental implications for both the properties of DFT and its meaning when used for computational Fourier analysis. The significance and implications of this inherent periodicity cannot be overemphasized (see Figure 7.2). Basically, all signals with the same samples in the range $0 \leq n \leq N - 1$, but different values elsewhere (a) have the same DFT coefficients, and (b) are treated by the N-point DFT as periodic sequences with period N. An intuitive explanation for this periodicity is given in the next section.

7.3   Sampling the Discrete-Time Fourier Transform

As we discussed in Section 6.1, sampling a continuous-time signal every T seconds causes periodic repetition of its Fourier transform every $2\pi/T$ radians/second. We next show that sampling the DTFT $\tilde{X}(e^{j\omega})$ of a sequence $x[n]$ every $2\pi/N$ radians, to obtain the DFT, causes periodic repetition of $x[n]$ every N samples.

Sequence "seen" by user

$x[n]$

DFT

IDFT

$\tilde{x}[n]$ Sequence "seen" by DFT
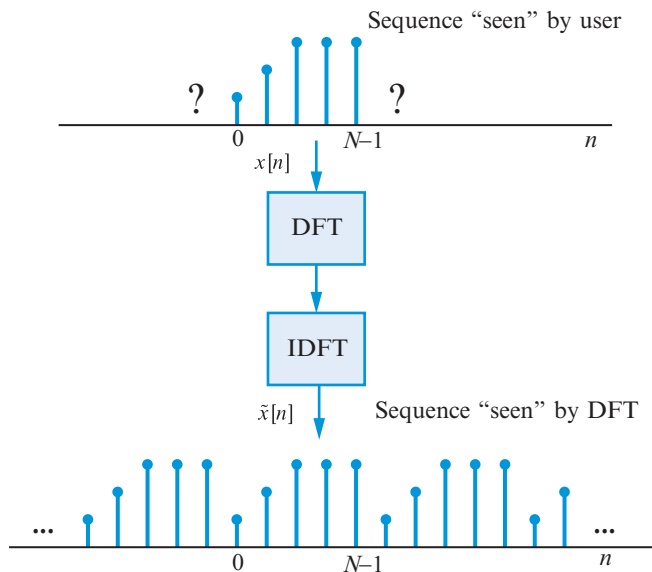
Figure 7.2 The inherent periodicity of DFT should always be taken into consideration to obtain a correct and meaningful interpretation of the results.

7.3.1   Frequency-domain sampling

We first recall that all samples of an aperiodic sequence $x[n]$, $-\infty < n < \infty$, can be uniquely recovered from its DTFT $\tilde{X}(e^{j\omega})$ using the inverse DTFT

$$x[n] = \frac{1}{2\pi} \int_0^{2\pi} \tilde{X}(e^{j\omega})e^{-j\omega n}d\omega. \tag{7.45}$$

Suppose now that we sample $\tilde{X}(e^{j\omega})$ at N equally spaced frequencies, that is,

$$X[k] \triangleq \tilde{X}\left(e^{j\frac{2\pi}{N}k}\right) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\frac{2\pi}{N}kn}, \tag{7.46}$$

where $k = 0, 1, \ldots, N-1$. The result of this sampling operation is a set of N DFT coefficients $X[k]$, $0 \le k \le N-1$. Since the DTFT is equal to the z-transform evaluated on the unit circle, it follows that the same result can be obtained by sampling $X(z)$ at N equally spaced points on the unit circle. Thus,

$$X[k] = X(z)|_{z=e^{j(2\pi/N)k}} = \tilde{X}\left(e^{j\frac{2\pi}{N}k}\right). \tag{7.47}$$

This interpretation, which is illustrated in Figure 7.3, demonstrates the inherent periodicity of the N-point DFT.

If we know all values of the DTFT $\tilde{X}(e^{j\omega})$ in the continuous range $0 \le \omega \le 2\pi$, we can recover all values of $x[n]$, from $n = -\infty$ to $n = \infty$, using the IDTFT (4.90). This result is illustrated in Figure 7.4(a). We now ask the question: How many samples of the

$$X[k] = X(z)|_{z=e^{j(2\pi/N)k}}$$

$$X(z)|_{z=e^{j\omega}}$$
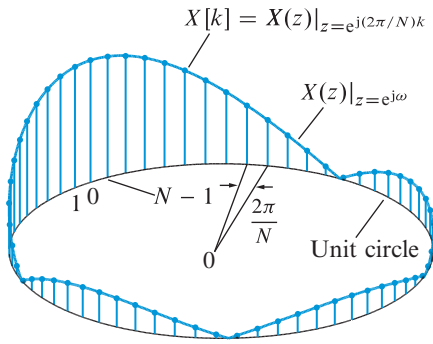
Figure 7.3 The N-point DFT is obtained by sampling the z-transform $X(z)$ at N equally spaced points $z_k = e^{j(2\pi/N)k}$, $0 \le k \le N - 1$, on the unit circle. The figure demonstrates the inherent periodicity of the N-point DFT sequence $X[k]$.
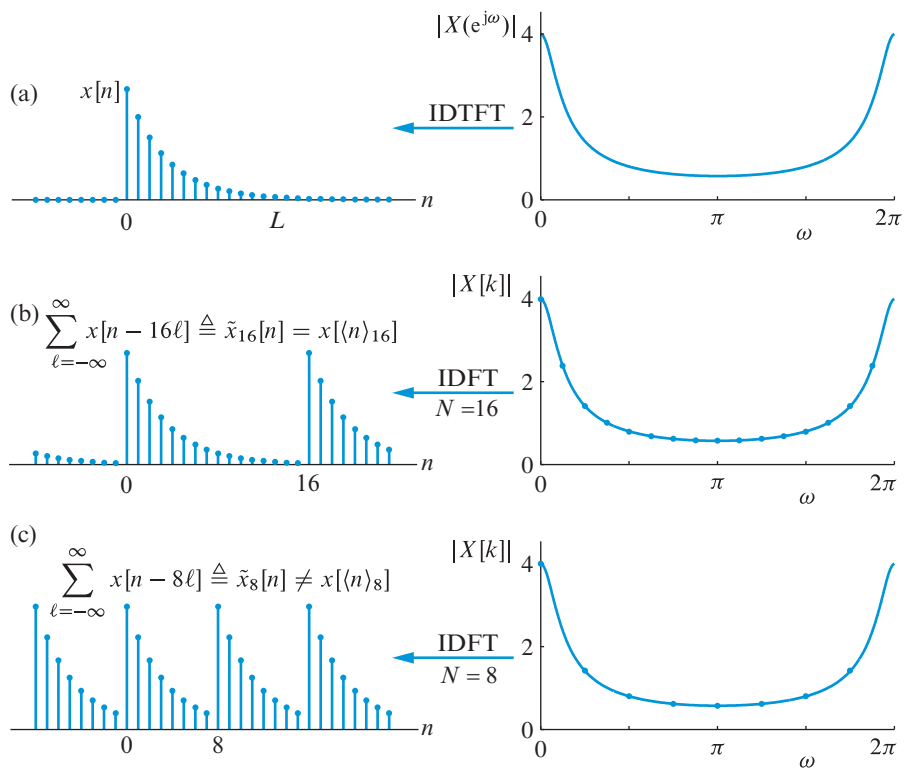
Figure 7.4 (a) Finite-length ($L = 12$) sequence $x[n]$ and its DTFT. (b) Periodic replication of $x[n]$ corresponding to sampling the DTFT with $N = 16$ (no time-domain aliasing). (c) Periodic replication of $x[n]$ corresponding to sampling the DTFT with $N = 8$ (time-domain aliasing).

sequence $x[n]$ can we recover from the samples (7.46) of the DTFT? We stress that this is different from the problem of recovering an N-point sequence from its N-point DFT; see (7.21) and (7.22). To address this question, we first recall that the DFS sequence $\tilde{X}[k]$ corresponding to DFT $X[k]$ is periodic in k with period N. Therefore, there should be a periodic sequence $\tilde{x}[n + N] = \tilde{x}[n]$ with Fourier coefficients related to the samples of DTFT. To find this relationship we divide the infinite summation (7.46) into an infinite number of finite summations, each of length N, as follows:

$$X[k] = \sum_{\ell=-\infty}^{\infty} \sum_{n=\ell N}^{\ell N+N-1} x[n]e^{-j\frac{2\pi}{N}kn}. \quad 0 \le k \le N-1 \tag{7.48}$$

We next change the index of the inner summation from n to $n - \ell N$ and interchange the order of summation. The result is

$$X[k] = \sum_{n=0}^{N-1} \left(\sum_{\ell=-\infty}^{\infty} x[n - \ell N] \right) e^{-j\frac{2\pi}{N}kn}. \quad 0 \le k \le N-1 \tag{7.49}$$

The term inside the parentheses is a sequence $\tilde{x}[n]$ defined by

$$\tilde{x}[n] \triangleq \sum_{\ell=-\infty}^{\infty} x[n - \ell N]. \tag{7.50}$$

This process is called *periodic extension* or *periodic replication* or *periodization*. Since $\tilde{x}[n]$ is obtained by periodic repetition of $x[n]$ every N samples, it is periodic with fundamental period N. Therefore, we can express $\tilde{x}[n]$ as a Fourier series

$$\tilde{x}[n] = \sum_{k=0}^{N-1} \tilde{c}_k e^{j\frac{2\pi}{N}kn}, \tag{7.51}$$

in which the Fourier coefficients are given by

$$\tilde{c}_k = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{x}[n]e^{-j\frac{2\pi}{N}kn}. \tag{7.52}$$

Comparison of (7.52) with (7.49) yields

$$\tilde{c}_k = \frac{1}{N}X[k] = \frac{1}{N}\tilde{X}\left(e^{j\frac{2\pi}{N}k}\right). \quad 0 \le k \le N-1 \tag{7.53}$$

If we substitute (7.53) into (7.51), we obtain the expression

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}\left(e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn} = \frac{1}{N} \sum_{k=0}^{N-1} X[k]W_N^{-kn}, \tag{7.54}$$

which shows how to compute the periodic sequence $\tilde{x}[n]$ from the samples $X[k]$ of $\tilde{X}(e^{j\omega})$ using the IDFT. To recover the original aperiodic sequence $x[n]$ from $\tilde{x}[n]$ we have to use (7.50). Since $\tilde{x}[n]$ is periodic with period N, we can recover N samples of $x[n]$, at most. This should be expected because we only use N samples of DTFT. Alternatively, if we create a finite-length N-point sequence from one period of a periodic sequence, that is, $x_N[n] \triangleq \tilde{x}[n]p_N[n]$, the Fourier series coefficients of $\tilde{x}[n]$ and the Fourier transform of $x_N[n]$ are related by (7.53) (see also (4.93)).

7.3.2   Time-domain aliasing

Figure 7.4(b) illustrates generation of the periodic extension $\tilde{x}[n]$ of a sequence $x[n]$ according to (7.54). We note that if $x[n]$ is zero outside the range $0 \le n \le L - 1$ and $N \ge L$, the shifted replicas of $x[n]$ do *not* overlap. In this case, we can exactly recover $x[n]$ from its periodic extension $\tilde{x}[n]$ using the formula

$$x[n] = \tilde{x}[n]p_N[n]. \tag{7.55}$$

On the other hand, if $N < L$, it is *not* possible to recover $x[n]$ from its periodic extension $\tilde{x}[n]$ because the shifted replicas of $x[n]$ overlap. This case is illustrated in Figure 7.4(c). This phenomenon, which is known as *time-domain aliasing*, is the counterpart of the frequency domain aliasing discussed in Section 6.3. Time-domain aliasing can be avoided only if $x[n]$ is time-limited to $0 \le n \le L - 1$ and the frequency-domain sampling period $\Delta\omega = 2\pi/N$ satisfies the condition $N \ge L$.

Example 7.3   Sampling and reconstruction of DTFT

A causal exponential sequence and its Fourier transform are given by

$$x[n] = a^n u[n], \quad 0 < a < 1 \xleftarrow{\text{DTFT}} \tilde{X}(e^{j\omega}) = \frac{1}{1 - ae^{-j\omega}}. \tag{7.56}$$

The spectrum $\tilde{X}(e^{j\omega})$ is sampled at frequencies $\omega_k = (2\pi/N)k, 0 \le k \le N - 1$. Compute and plot the reconstructed spectrum $\tilde{X}_N(e^{j\omega})$ when $N = 16$ for $a = 0.9$.

Solution In a typical practical application, where the sequence $x[n]$ is unavailable, we first compute $\tilde{x}[n]$ for $0 \le n \le N - 1$, using (7.54), that is,

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}\left(e^{j\frac{2\pi}{N}k}\right) e^{j\frac{2\pi}{N}kn}. \tag{7.57}$$

Then, we evaluate the reconstructed spectrum $\tilde{X}_N(e^{j\omega})$ at a set of equally spaced frequencies $\omega_k = (2\pi/K)k, 0 \le k \le K - 1$, where $K \ge N$, by

$$\tilde{X}_N\left(e^{j\frac{2\pi}{K}k}\right) = \sum_{n=0}^{N-1} \tilde{x}[n]e^{-j\frac{2\pi}{K}kn}. \tag{7.58}$$
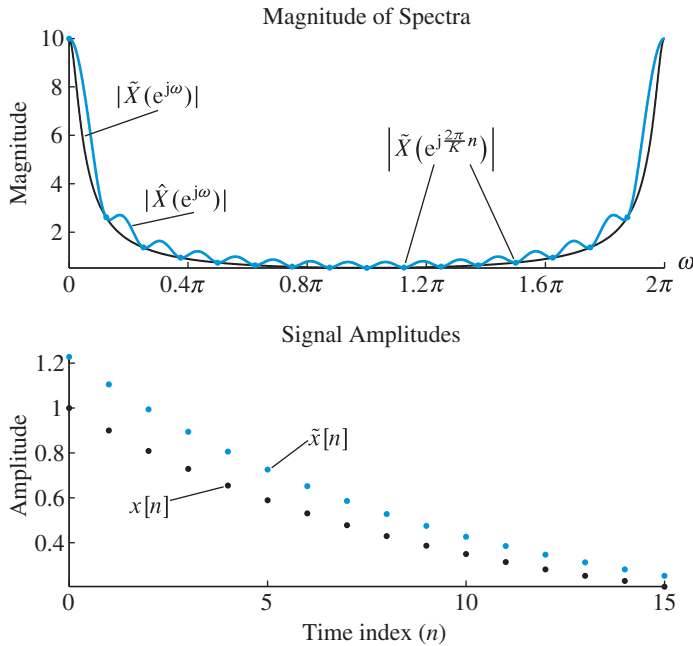
Figure 7.5 Reconstructing the DTFT from equally spaced samples. The effect of time-domain aliasing is evident both on the reconstructed DTFT and the corresponding discrete-time signal obtained by one period of the IDFT.

Figure 7.5 shows plots of $\tilde{X}(e^{j\omega})$, $\tilde{X}_N(e^{j\omega})$, $x[n]$, and $\tilde{x}[n]$ for $N = 16$ and $a = 0.9$. Since $x[n]$ is available, we can better understand the results obtained using (7.50) to find the relationship between $x[n]$ and its periodic extension $\tilde{x}[n]$. We first note that, since $x[n] = 0$ for $n < 0$, the shifted replicas $x[n - \ell N]$ for $\ell \geq 1$ do not affect the values of $\tilde{x}[n]$ in the range $0 \leq n \leq N - 1$. Thus, we have

$$\tilde{x}[n] = \sum_{\ell=-\infty}^{0} a^{n-\ell N} = a^n \sum_{\ell=0}^{\infty} (a^N)^\ell = \frac{a^n}{1 - a^N} = \frac{x[n]}{1 - a^N}, \quad 0 \leq n \leq N - 1$$

where the factor $1/(1 - a^N)$ represents the effect of time-domain aliasing. For $a = 0.9$, we have $1/(1 - 0.9^{16}) = 1.23$, which results in a significant amount of time-domain aliasing or equivalently in a large spectrum reconstruction error. Since $0 < a < 1$, the aliasing error tends to zero as $a \to 0$ or $N \to \infty$ (see Tutorial Problem 9). The reconstructed Fourier transform is given by

$$\tilde{X}_N(e^{j\omega}) = \sum_{n=0}^{N-1} \tilde{x}[n] e^{-j\omega n} = \frac{1}{1 - a^N} \frac{1 - a^N e^{-j\omega N}}{1 - a e^{-j\omega}}. \tag{7.59}$$

Note that although $\tilde{X}_N(e^{j\omega}) \neq \tilde{X}(e^{j\omega})$, the values of the two transforms at the sampling points $\omega_k = (2\pi/N)k$ are identical, as expected.   ∎

Reconstruction of DTFT $\tilde{X}(e^{j\omega})$

If there is no time-domain aliasing then $x[n]$ is an N-point sequence and hence we can determine $\tilde{X}(e^{j\omega})$ from the sequence $x[n]$ given by (7.6) as

$$\tilde{X}(e^{j\omega}) = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}. \tag{7.60}$$

However, it is possible to express $\tilde{X}(e^{j\omega})$ directly in terms of its samples $\tilde{X}(e^{j\frac{2\pi}{N}k})$. To derive an interpolation formula we first substitute (7.57) in (7.55). The result is

$$x[n] = \left[\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(e^{j\frac{2\pi}{N}k}) e^{j\frac{2\pi}{N}kn} \right] p_N[n]. \tag{7.61}$$

Taking the DTFT of (7.61) via (7.60) and using the modulation property (4.140), we obtain

$$\tilde{X}(e^{j\omega}) = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(e^{j\frac{2\pi}{N}k}) \tilde{P}_N\left[e^{j(\omega - \frac{2\pi}{N}k)} \right], \tag{7.62}$$

which is valid for sequences with length $L \leq N$. The interpolation function $\tilde{P}_N(e^{j\omega})$ is a phase-shifted version of the Dirichlet function (4.80)

$$\tilde{P}_N(e^{j\omega}) = \frac{\sin(\omega N/2)}{N \sin(\omega/2)} e^{-j\omega(N-1)/2}. \tag{7.63}$$

Since $\tilde{P}_N(e^{j0}) = 1$ and $\tilde{P}_N(e^{j\frac{2\pi}{N}k}) = 0$ for $k = 1, 2, \ldots, N-1$ (see Figure 4.21), the interpolation formula (7.62) gives exactly the original sample values $\tilde{X}(e^{j\frac{2\pi}{N}k})$ for $\omega = (2\pi/N)k$. The values of $\tilde{X}(e^{j\omega})$ for $\omega \neq (2\pi/N)k$ are obtained from (7.62) using a properly weighted linear combination of the original sample values. Reconstruction formula (7.62) is useful for a conceptual qualitative understanding of the frequency-domain interpolation process. In practice, we interpolate at a denser set of frequencies using the DFT and a procedure known as zero-padding.

Zero-padding Suppose we are given samples $X[k] \triangleq \tilde{X}(e^{j\frac{2\pi}{N}k})$, $0 \leq k \leq N-1$, of the DTFT of a finite-length sequence $x[n]$. As we note from (7.54), we can use the N-point IDFT to obtain the sequence $x[n]$, $0 \leq n \leq N-1$. Then, we can evaluate $\tilde{X}(e^{j\omega})$ at a set of K equispaced frequencies $\omega_k = (2\pi/K)k$ by

$$\tilde{X}(e^{j\frac{2\pi}{K}k}) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{K}kn}. \quad 0 \leq k \leq K-1 \tag{7.64}$$

We usually choose K much larger than N, so that the plot of $\tilde{X}(e^{j\frac{2\pi}{K}k})$ appears to be continuous. If we define the "zero-padded" sequence

$$x_{zp}[n] \triangleq \begin{cases} x[n], & 0 \le n \le N-1 \\ 0, & N \le n \le K-1 \end{cases} \tag{7.65}$$

then we can easily see that

$$\tilde{X}(e^{j\frac{2\pi}{K}k}) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{K}kn} = \sum_{n=0}^{K-1} x_{zp}[n]W_K^{kn} = X_{zp}[k], \tag{7.66}$$

that is, the K-point DFT $X_{zp}[k]$ of the zero-padded sequence provides values of the DTFT at a finer grid of equispaced frequencies. The result is a better representation of $\tilde{X}(e^{j\omega})$ for display purposes; there is *no* additional information that can be exploited by signal processing algorithms. The distinction between increased spectral resolution and better visualization is discussed in Section 7.6.

Example 7.4   Zero-padding of a rectangular pulse

To illustrate the concept of zero-padding, we consider the finite duration rectangular pulse sequence

$$x[n] = u[n] - u[n-N] = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{otherwise} \end{cases} \tag{7.67}$$

with $N = 4$. Figure 7.6(a) shows the sequence $x[n]$ and the magnitude of its DTFT. Figures 7.6(b)–(d) show the K-point DFT of $x[n]$ for $K = 4, 8,$ and 16 samples. Since $x[n] = 0$ for $n \ge N$, the K-point DFT for $K > N$ is the DFT of the original sequence padded by $(K - N)$ zeros. The N-point DFT is sufficient to uniquely represent the N-samples of the original sequence. However, it does *not* provide a "good picture" of the spectral composition of $x[n]$. This better picture is obtained by padding the N-point sequence $x[n]$ by $(K - N)$ zeros and computing the K-point DFT.

To explain this improvement in visual appearance with increasing K, we note that the spectral composition of $x[n]$ is provided by its DTFT, which is given by

$$\tilde{X}(e^{j\omega}) = \sum_{n=0}^{N-1} e^{-j\omega n} = \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} = \frac{\sin(\omega N/2)}{\sin(\omega/2)} e^{-j\omega(N-1)/2}. \tag{7.68}$$

According to (7.66), the K-point DFT of the zero-padded sequence provides samples of $\tilde{X}(e^{j\omega})$ at $\omega_k = (2\pi/K)k$, that is,

$$X[k] = \tilde{X}(e^{j\frac{2\pi}{K}k}) = \frac{\sin(\pi Nk/K))}{\sin(\pi k/K)} e^{-j\pi(N-1)k/K}. \tag{7.69}$$

If we select $K = N$, the DFT (7.69) becomes

$$\tilde{X}[k] = \begin{cases} N, & k = 0 \\ 0. & k = 1, 2, \ldots, K-1 \end{cases} \tag{7.70}$$
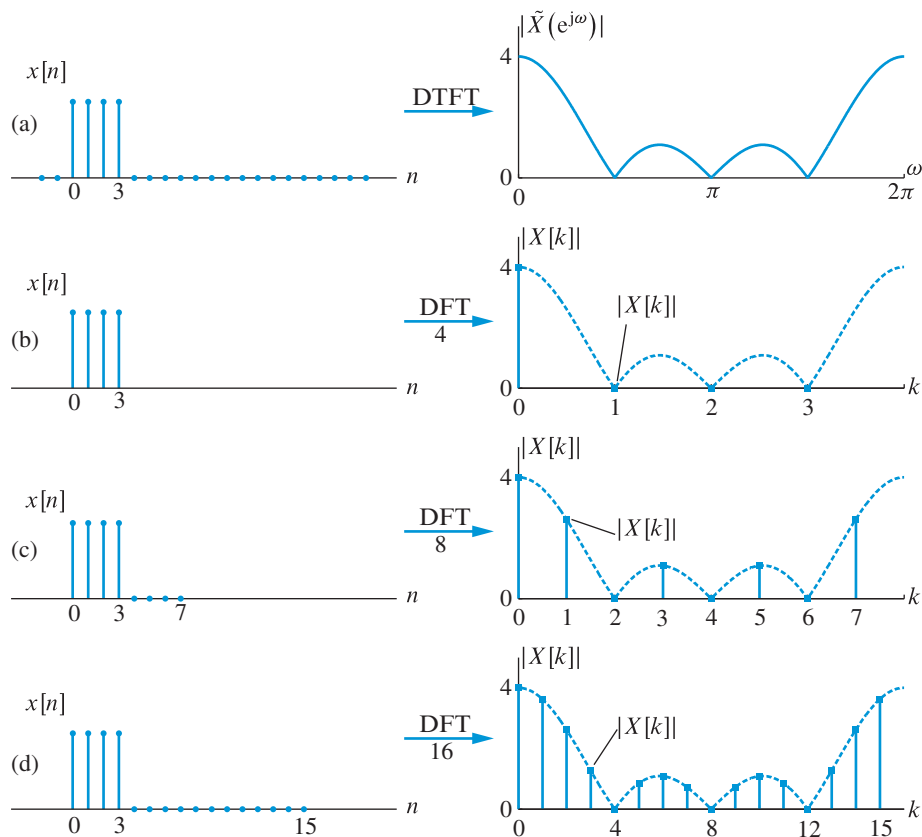
Figure 7.6 (a) A finite-length ($N = 4$) rectangular pulse $x[n]$ and the magnitude of its DTFT. (b)–(d) Magnitude of the K-point DFT of $x[n]$ for $K = 4, 8$, and 16, obtained by using zero-padding.

This corresponds to sampling the DTFT at its zero crossings (except at $\omega = 0$). Figure 7.7 shows the magnitude and phase of $\tilde{X}(e^{j\omega})$ when $N = 8$ and the samples are evaluated by the K-point DFT with zero-padding for $K = 8$ and $K = 64$. Clearly, zero-padding helps to make the shape of the DTFT more evident by evaluating samples at a denser frequency grid. ∎

Reconstruction of $X(z)$ We observed that when $x[n]$ is zero outside the range $0 \leq n \leq L - 1$ and $L \leq N$, we can recover the values of $x[0], x[1], \ldots, x[N - 1]$ from the samples $\tilde{X}(e^{j\frac{2\pi}{N}k})$, $0 \leq k \leq N - 1$ of its DTFT. Consequently, we can uniquely determine the z-transform of $x[n]$ using

$$X(z) = \sum_{n=0}^{N-1} x[n]z^{-n}. \tag{7.71}$$
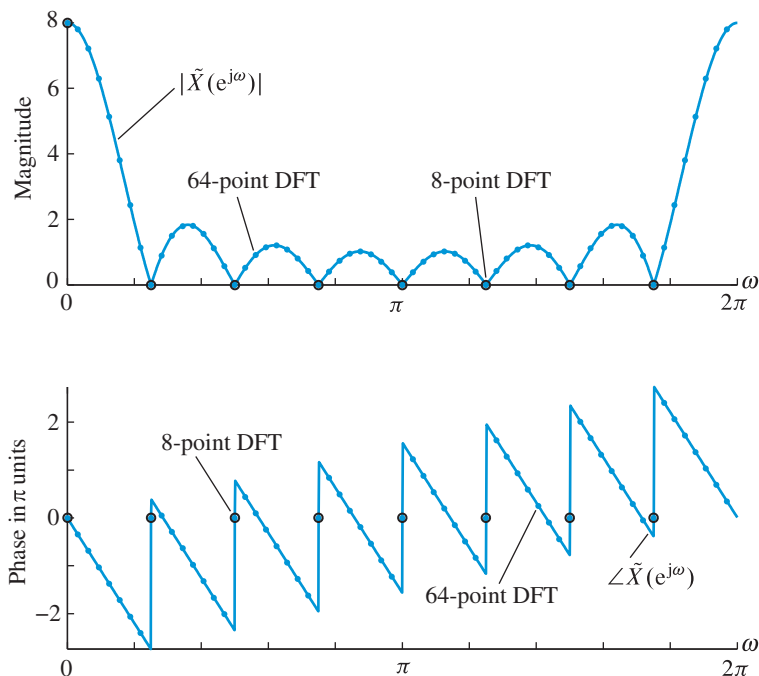
Figure 7.7 The magnitude and phase of the DTFT $\tilde{X}(e^{j\omega})$ given by (7.68) when $N = 8$ and the samples are evaluated by the 8-point DFT and the 64-point DFT using zero-padding.

Substituting (7.61) into (7.71), we can express $X(z)$ as a function of $\tilde{X}(e^{j\frac{2\pi}{N}k})$ as

$$X(z) = \sum_{n=0}^{N-1} \left[\frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(e^{j\frac{2\pi}{N}k}) e^{j\frac{2\pi}{N}kn} \right] z^{-n}$$

$$= \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}(e^{j\frac{2\pi}{N}k}) \sum_{k=0}^{N-1} \left(e^{j\frac{2\pi}{N}k} z^{-1} \right)^n.$$

Computing the last summation using the geometric sum formula (2.120) yields

$$X(z) = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{\tilde{X}(e^{j\frac{2\pi}{N}k})}{1 - e^{j\frac{2\pi}{N}k} z^{-1}}. \tag{7.72}$$

For $z = e^{j\omega}$, formula (7.72) takes a form known as *polynomial Lagrange interpolation*. This form can be reduced to (7.62) by simple algebraic manipulations (see Tutorial Problem 10).

7.3.4   Relationships between CTFT, DTFT, and DFT

There are two important properties that make the DFT so eminently useful in signal processing. First, the N-point DFT provides a unique representation of the N-samples of a

finite duration sequence. Second, the DFT provides samples of the DTFT of the sequence at a set of equally spaced frequencies. This sampling process results in the inherent periodicity of the DFT. Understanding the underlying periodicity of DFT is absolutely critical for the correct application of DFT and meaningful interpretation of the results obtained.

To understand the relationship between the CTFT and the DFT, we consider the illustration in Figure 7.8. Suppose that we are given a continuous-time signal $x_c(t)$ with Fourier transform $X_c(j\Omega)$. Application of discrete-time signal processing starts by uniformly sampling $x_c(t)$ at $t = nT$. This results in a discrete-time signal $x[n] = x_c(nT)$ with DTFT specified by

$$\tilde{X}(e^{j\Omega T}) = \sum_{n=-\infty}^{\infty} x_c(nT)e^{-j\Omega Tn} = \frac{1}{T}\sum_{m=-\infty}^{\infty} X_c\left(j\Omega - j\frac{2\pi}{T}m\right). \qquad (7.73)$$

Since $\omega = \Omega T$, the N-point DFT $X[k]$ is obtained by sampling the DTFT $\tilde{X}(e^{j\omega})$ at $\omega = 2\pi k/N$ or $\tilde{X}(e^{j\Omega T})$ at $\Omega = 2\pi k/NT$ for $0 \leq k \leq N-1$. The result is

$$X[k] = \frac{1}{T}\sum_{m=-\infty}^{\infty} X_c\left(j\frac{2\pi k}{NT} - j\frac{2\pi}{T}m\right). \quad k = 0, 1, \ldots, N-1 \qquad (7.74)$$
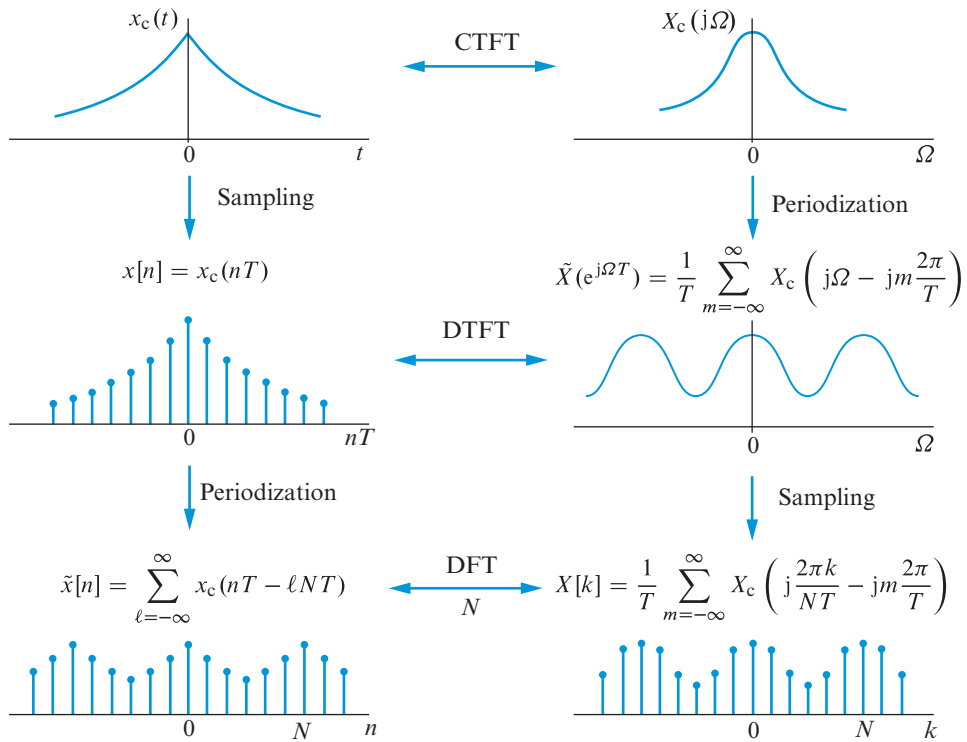
Figure 7.8 Operations and steps required to establish the relationship between CTFT, DTFT, and DFT. We note that sampling in one domain is equivalent to periodization in the other domain. Periodic replication may cause frequency-domain or time-domain aliasing.

Sampling the DTFT of $x[n]$ is equivalent to the periodic repetition of $x[n]$ with period N or equivalently of $x_c(nT)$ with period NT. The result is a periodic sequence

$$\tilde{x}[n] = \sum_{\ell=-\infty}^{\infty} x_c(nT - \ell NT). \tag{7.75}$$

From the discussion in Section 7.2 it follows that $X[k]$ is the N-point DFT of $\tilde{x}[n]$. Therefore, we have the following N-point DFT pair

$$\sum_{\ell=-\infty}^{\infty} x_c(nT - \ell NT) \xrightarrow[N]{\text{DFT}} \frac{1}{T} \sum_{m=-\infty}^{\infty} X_c\left(j\frac{2\pi k}{NT} - j\frac{2\pi}{T}m\right), \tag{7.76}$$

where $0 \leq n \leq N - 1$ and $0 \leq k \leq N - 1$.

We conclude that sampling a continuous-time signal $x_c(t)$ at $t = nT$ and then Fourier transforming the resulting sequence $x[n] = x_c(nT)$ at $\Omega = 2\pi k/NT$ results in an N-point DFT pair. Equation (7.76) reveals the frequency-domain aliasing caused by time-domain sampling and the time-domain aliasing caused by frequency-domain sampling, which in turn explains the inherent periodicity of the DFT. The relationship in (7.76), which is illustrated in Figure 7.8, demonstrates the potential pitfalls in computation of CTFT using the DFT; this important topic is discussed in Section 7.6.

7.4   Properties of the Discrete Fourier Transform

In Section 7.2 we showed that the N-point DFT provides a unique frequency-domain representation of N consecutive samples of a discrete-time signal. We have also established relationships between the DFT, Fourier series, Fourier transforms, and the z-transform. As a result, we expect the DFT to have properties which resemble the properties of these other transforms. However, there are also some important differences due to the inherent periodicity of DFT. Understanding these properties is very important for the correct use of DFT in signal processing applications. The discussion parallels the discussion of Section 4.5 for the DTFT.

7.4.1   Linearity

Let $x_1[n]$ and $x_2[n]$ be finite-duration sequences having lengths N_1 and N_2, respectively. The linear combination

$$y[n] \triangleq a_1 x_1[n] + a_2 x_2[n] \tag{7.77}$$

has maximum length $N_y = \max(N_1, N_2)$. The N-point DFT of $y[n]$, where the length should satisfy the condition $N \geq N_y$, is given by

$$Y[k] = a_1 \sum_{n=0}^{N_1-1} x_1[n]W_N^{kn} + a_2 \sum_{n=0}^{N_2-1} x_1[n]W_N^{kn}. \quad 0 \leq k \leq N-1 \quad (7.78)$$

The first summation in (7.78) is the DFT of $x_1[n]$ padded by $(N-N_1)$ zeros and the second summation is the DFT of $x_2[n]$ padded by $(N-N_2)$ zeros (see page 369 for a discussion on zero-padding). Thus, we have

$$Y[k] = a_1 X_1[k] + a_2 X_2[k], \quad 0 \leq k \leq N-1 \quad (7.79)$$

where $X_1[k]$ and $X_2[k]$ are the N-point DFTs of $x_1[n]$ and $x_2[n]$, respectively. We again stress that N should satisfy the condition $N \geq \max(N_1, N_2)$.

7.4.2   Periodic, circular, and modulo-N operations

From the discussion in Section 7.2.3 (see Figure 7.2), it is evident that the DFT treats the N-point signal $x[n]$ and its DFT coefficients $X[k]$ as primary periods of periodic sequences $\tilde{x}[n]$ and $\tilde{X}[k]$, respectively. This point of view is illustrated in Figures 7.9(a) and 7.9(b), which show a finite-length sequence $x[n]$ and its periodic extension $\tilde{x}[n]$. Another way to informally visualize this inherent periodicity is to wrap the N-point sequence $x[n]$ around a cylinder with a circumference equal to the sequence length. This approach is illustrated in 7.9(c). Traveling around the circle once, starting at $n = 0$, yields the N-point sequence $x[n]$. If we keep repeatedly going around the circle, we obtain the periodic extension $\tilde{x}[n]$ of $x[n]$.

Geometrically, the finite-duration sequence $x[n]$ is recovered by unwrapping the cylinder and laying it flat so that the circular time-axis is mapped on the linear time-axis. The relation between the linear time-index and the circular time-index, known as *modulo-N operation*, is defined by

$$n = \ell N + r, \ 0 \leq r \leq N-1 \Rightarrow \langle n \rangle_N \triangleq n \text{ modulo } N = r, \quad (7.80)$$

that is, given n and N, we choose ℓ so that the index r is always an integer between 0 and $N-1$. This compact notation (7.80) allows us to express the periodic extension $\tilde{x}[n]$ of a finite-length N-point sequence $x[n]$, $0 \leq n \leq N-1$, as

$$\tilde{x}[n] = x[\langle n \rangle_N], \quad \text{for all } n \quad (7.81)$$

using the circular indexing operation on n. We note that all these properties and interpretations also hold for the N-point DFT sequence $X[k]$. Thus the periodic DFS in terms of DFT is given by

$$\tilde{X}[k] = X[\langle k \rangle_N]. \quad \text{for all } k \quad (7.82)$$

In MATLAB, the modulo-N operation is performed by the function

$$\texttt{m=mod(n,N)}. \quad (7.83)$$
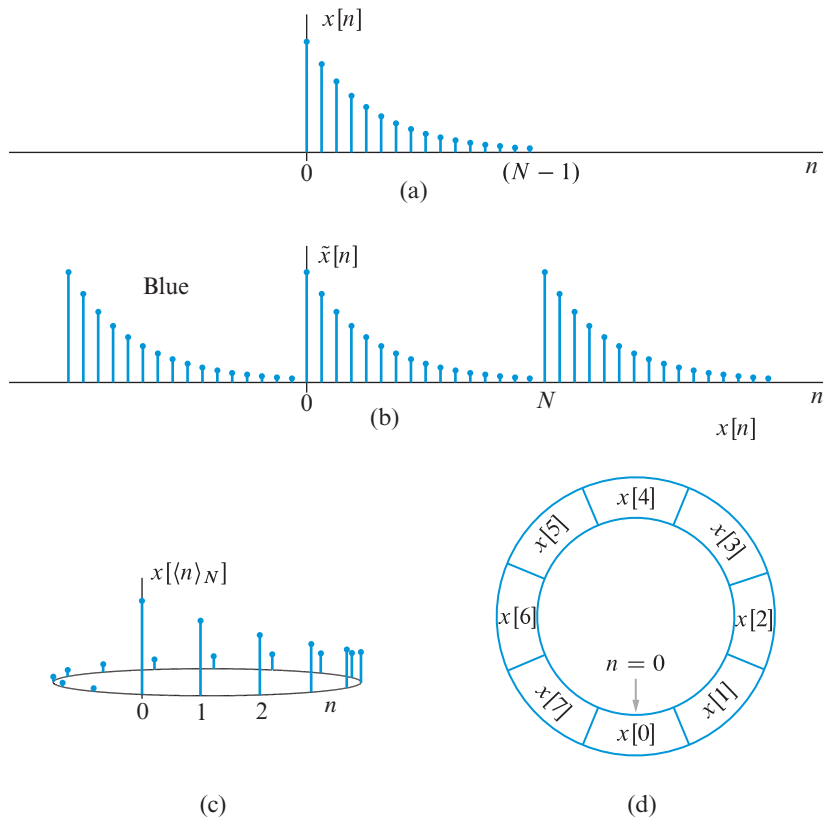
Figure 7.9 Periodic extension and circular wrapping: (a) finite-length sequence $x[n]$, (b) periodic extension sequence $\tilde{x}[n]$ formed by replicating $x[n]$ every N samples, (c) wrapping the sequence $x[n]$ around a cylinder with circumference N and using modulo-N addressing results to the periodic extension $\tilde{x}[n]$, (d) representation of a circular buffer with modulo-N indexing.

Circular buffer These ideas lead to the concept of a circular buffer, where the data are stored counterclockwise in a circular fashion, as shown in Figure 7.9(d). Circular buffers use modulo-N addressing and the index moves from $(N-1)$ to 0 by wrapping around the circle. All operations that modify the index n of a finite-length sequence in DFT applications should be interpreted in the context of circular or modulo-N indexing.

Important note In summary, we emphasize that the inherent periodicity imposed by the N-point DFT $X[k]$ of a finite-length N-point sequence $x[n]$ can be dealt with using either the periodic extension ($\tilde{x}[n]$ or $\tilde{X}[k]$) or the circular mapping ($x[\langle n \rangle_N]$ or $X[\langle k \rangle_N]$) operations. We will use one of these two approaches as convenient.

Circular folding (or reversal) The operation $z[n] = x[-n]$ of time-reversal (or time-folding) is not defined when $x[n]$ is unavailable outside the range $0 \le n \le N-1$. However, based on the inherent periodicity of the DFT, we can define this operation in a mathematically consistent way using the periodic extension $\tilde{x}[n]$. This process, which is illustrated
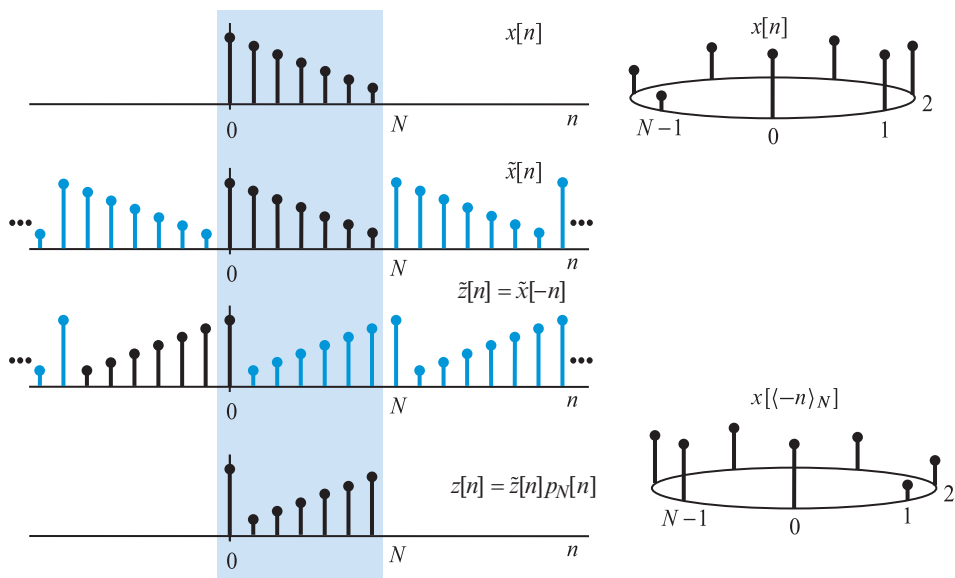
Figure 7.10 Circular folding using periodic extension, linear time-reversal, and extraction of the primary period.

```
function y=circfold(x,N)
% Circular time reversal (folding)
if length(x) > N; error('N < length(x)'); end
x=[x zeros(1,N-length(x))];
n=(0:1:N-1);
y=x(mod(-n,N)+1);
```

Figure 7.11 MATLAB function for circular time-reversal (folding) of a finite-length sequence.

in Figure 7.10, involves three steps: (a) generation of the periodic extension $\tilde{x}[n]$, (b) time-reversal about $n = 0$ to obtain $\tilde{z}[n] = \tilde{x}[-n]$, and (c) extraction of a single period of $\tilde{z}[n]$ to obtain the desired sequence $z[n] = \tilde{z}[n]p_N[n]$. The same result is obtained, in a single step, by wrapping the sequence $x[n]$ around a cylinder clockwise. This operation, which is known as *circular folding*, is defined by

$$z[n] = x[\langle -n \rangle_N] \triangleq \begin{cases} x[0], & n = 0 \\ x[N - n]. & 1 \le n \le N - 1 \end{cases} \tag{7.84}$$

We note that, as expected, this definition keeps the time-reversed sequence in the range $0 \le n \le N - 1$. The sample $x[0]$ remains at its position and the remaining samples are arranged in reverse order, that is, we have $x[0], x[N - 1], \ldots, x[2], x[1]$. A MATLAB function, circfold, to implement the circular folding operation (7.84) is given in Figure 7.11.

The circular time-reversal property of the DFT is (see Tutorial Problem 11)

$$x[\langle -n \rangle_N] \xleftarrow[N]{\text{DFT}} X[\langle -k \rangle_N], \tag{7.85}$$

which also results in a circular folding (or frequency-reversal) of $X[k]$ and is analogously defined, that is,

$$X[\langle -k \rangle_N] \triangleq \begin{cases} X[0], & k = 0 \\ X[N-k]. & 1 \le k \le N-1 \end{cases} \tag{7.86}$$

Circular symmetry In general, a sequence has even symmetry if time-reversal results in an identical sequence; if time-reversal changes the signs of the samples, the sequence has odd symmetry. Thus, the difference between linear and circular time-reversal has implications on the definition of symmetry for finite-length sequences. For sequences defined for all n, symmetry is determined about the point $n = 0$. In the circular framework, symmetry is determined with respect to the circle diameter passing through the point $n = 0$. Thus, for a finite-length real-valued sequence $x[n]$, circular symmetry is defined by the conditions:

$$x[n] = x[\langle -n \rangle_N], \qquad \text{(circular even symmetry)} \tag{7.87}$$

$$x[n] = -x[\langle -n \rangle_N]. \qquad \text{(circular odd symmetry)} \tag{7.88}$$

If we use (7.84), we can define circular symmetry by avoiding the modulo-N computation of indices. The equivalent conditions are given by

$$x[n] = \begin{cases} x[0], & n = 0 \\ x[N-n], & 1 \le n \le N-1 \end{cases} \qquad \text{(circular even symmetry)} \tag{7.89}$$

$$x[n] = \begin{cases} 0, & n = 0 \\ -x[N-n]. & 1 \le n \le N-1 \end{cases} \qquad \text{(circular odd symmetry)} \tag{7.90}$$

Note that we check the sequence $x[1], x[2], \ldots, x[N-1]$ for even or odd symmetry about the point $N/2$; the value of sample $x[0]$ should be zero to assure odd circular symmetry (see Figure 7.12).

7.4.3   Symmetry properties of the DFT

As we recall from Section 4.5.2, the DTFT of real-valued sequences has some useful symmetry properties. A similar set of properties holds for the DFT if we interpret symmetry in the circular context defined by (7.89) and (7.90). To develop these properties, we start with the most general case of a complex-valued N-point sequence $x[n]$ with its complex-valued DFT $X[k]$. The complex numbers $x[n]$ and $X[k]$ can be expressed in rectangular form as

$$x[n] = x_R[n] + jx_I[n], \quad 0 \le n \le N-1 \tag{7.91}$$

$$X[k] = X_R[k] + jX_I[k]. \quad 0 \le k \le N-1 \tag{7.92}$$
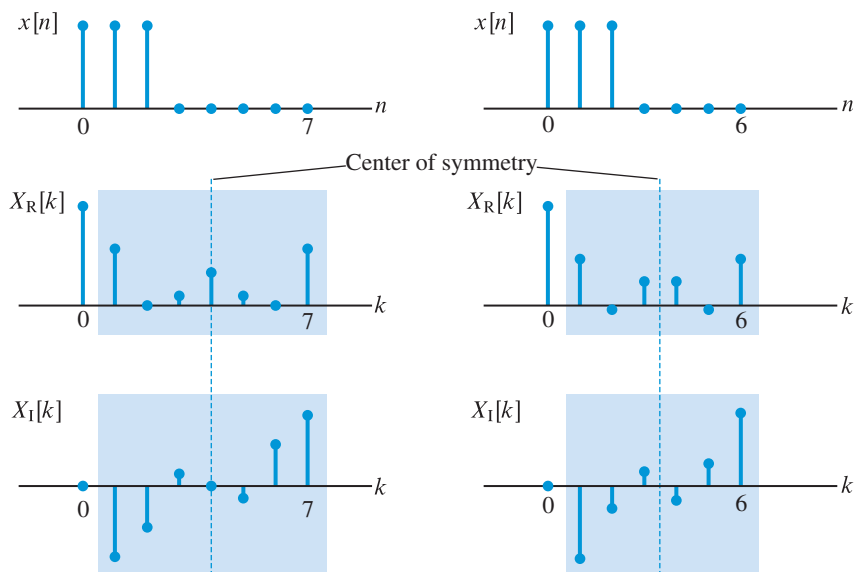
Figure 7.12 Symmetries of real and imaginary parts of the DFT of a real-valued sequence with even and odd number of samples (length). Note that we check the values for $k = 1, 2, \ldots, N - 1$ for even or odd symmetry about the point $N/2$; for odd symmetry the value at $k = 0$ should be zero.

Substituting (7.91) into the definition (7.21) of the DFT, we obtain

$$X_R[k] = \sum_{n=0}^{N-1} \left[x_R[n] \cos\left(\frac{2\pi}{N}kn\right) + x_I[n] \sin\left(\frac{2\pi}{N}kn\right) \right], \quad 0 \le k < N \tag{7.93}$$

$$X_I[k] = \sum_{n=0}^{N-1} \left[x_R[n] \sin\left(\frac{2\pi}{N}kn\right) - x_I[n] \cos\left(\frac{2\pi}{N}kn\right) \right]. \, 0 \le k < N \tag{7.94}$$

Similarly, substituting (7.92) into the inverse DFT (7.22), we obtain

$$x_R[n] = \frac{1}{N} \sum_{k=0}^{N-1} \left[X_R[k] \cos\left(\frac{2\pi}{N}kn\right) - X_I[k] \sin\left(\frac{2\pi}{N}kn\right) \right], 0 \le n < N \tag{7.95}$$

$$x_I[n] = \frac{1}{N} \sum_{k=0}^{N-1} \left[X_R[k] \sin\left(\frac{2\pi}{N}kn\right) + X_I[k] \cos\left(\frac{2\pi}{N}kn\right) \right]. \, 0 \le n < N \tag{7.96}$$

Real-valued sequences If $x[n]$ is real, setting $x_I[n] = 0$ in (7.93) and (7.94) yields

$$X_R[k] = \sum_{n=0}^{N-1} x_R[n] \cos\left(\frac{2\pi}{N}kn\right) = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi}{N}kn\right), \tag{7.97}$$

$$X_I[k] = \sum_{n=0}^{N-1} x_R[n] \sin\left(\frac{2\pi}{N}kn\right) = \sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi}{N}kn\right), \tag{7.98}$$

which, using the inherent periodicity (see Important note on page 376), implies that $\tilde{X}_R[-k] = \tilde{X}_R[k]$ and $\tilde{X}_I[-k] = -\tilde{X}_I[k]$. Therefore, we have

$$\tilde{X}^*[k] = \tilde{X}_R[k] - j\tilde{X}_I[k] = \tilde{X}_R[-k] + j\tilde{X}_I[-k] = \tilde{X}[-k] \tag{7.99}$$

for all k. Now using the circular indexing to represent the DFT, we can write

$$X_R[k] = X_R[\langle -k \rangle_N] = \begin{cases} X_R[0], & k = 0, \\ X_R[N - k], & 1 \le k \le N - 1. \end{cases} \tag{7.100}$$

$$X_I[k] = -X_I[\langle -k \rangle_N] = \begin{cases} 0, & k = 0, \\ -X_I[N - k], & 1 \le k \le N - 1. \end{cases} \tag{7.101}$$

$$X^*[k] = X[\langle -k \rangle_N] = \begin{cases} X[0], & k = 0, \\ X[N - k], & 1 \le k \le N - 1. \end{cases} \tag{7.102}$$

for $0 \le k \le N - 1$. Figure 7.12 shows an N-point sequence and its DFT for $N = 8$ (even) and $N = 7$ (odd). We note that $X_R[k]$, $1 \le k \le N - 1$ and $X_I[k]$, $1 \le k \le N - 1$ have even and odd symmetry about the point $N/2$, respectively. The center of symmetry $N/2$ is a sample of the sequence, if N is even, and half-way between two samples if N is odd. For $k = 0$ we obtain $X_R[0] = X_R[0]$ and $X_I[0] = -X_I[0]$. The last condition implies that we will always have $X_I[0] = 0$. A natural explanation of these relations is provided by the concepts of periodic or circular symmetry.

Decomposition into circular-even and -odd components Any N-point real sequence $x[n]$ can be decomposed into a sum of circularly-even and circularly-odd components as

$$x[n] = x^{ce}[n] + x^{co}[n], \quad 0 \le n \le N - 1 \tag{7.103}$$

where

$$x^{ce}[n] \triangleq \frac{x[n] + x[\langle -n \rangle_N]}{2} = x^{ce}[\langle -n \rangle_N] \tag{7.104a}$$

$$= \begin{cases} \frac{1}{2}(x[n] + x[N - n]), & 1 \le n \le N - 1 \\ x[0], & n = 0 \end{cases} \tag{7.104b}$$

and

$$x^{co}[n] \triangleq \frac{x[n] - x[\langle -n \rangle_N]}{2} = -x^{co}[\langle -n \rangle_N] \tag{7.105a}$$

$$= \begin{cases} \frac{1}{2}(x[n] - x[N - n]), & 1 \le n \le N - 1 \\ 0. & n = 0 \end{cases} \tag{7.105b}$$

Real sequences with circular symmetries If $x[n]$ is real and circularly-even, that is,

$$x[n] = x[\langle -n \rangle_N] = \begin{cases} x[0], & n = 0 \\ x[N - n], & 1 \le n \le N - 1 \end{cases} \tag{7.106}$$

then (7.98) yields $X_I[k] = 0$. This leads to the following DFT pair:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cos\left(\frac{2\pi}{N}kn\right) \xleftrightarrow[N]{\text{DFT}} X[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{2\pi}{N}kn\right), \qquad (7.107)$$

which shows that the DFT of a real and circularly-even sequence is also real and circularly-even.

If $x[n]$ is real and circularly-odd, that is,

$$x[n] = -x[\langle -n \rangle_N] = \begin{cases} 0, & n = 0 \\ -x[N-n], & 1 \le n \le N-1 \end{cases} \qquad (7.108)$$

then (7.97) yields $X_R[k] = 0$. This leads to the following DFT pair:

$$x[n] = j\frac{1}{N} \sum_{k=0}^{N-1} X[k] \sin\left(\frac{2\pi}{N}kn\right) \xleftrightarrow[N]{\text{DFT}} X[k] = -j \sum_{n=0}^{N-1} x[n] \sin\left(\frac{2\pi}{N}kn\right), \qquad (7.109)$$

which shows that the DFT of a real and circularly-odd sequence is imaginary and circularly-odd.

Imaginary sequences with circular symmetries If $x[n]$ is purely imaginary, that is, $x[n] = jx_I[n]$, then (7.93) and (7.94) yield

$$X_R[k] = \sum_{n=0}^{N-1} x_I[n] \sin\left(\frac{2\pi}{N}kn\right) \quad \text{and} \quad X_I[k] = -\sum_{n=0}^{N-1} x_I[n] \cos\left(\frac{2\pi}{N}kn\right), \qquad (7.110)$$

which show that $X_R[k]$ is circularly-odd and $X_I[k]$ is circularly-even. Furthermore, if $x_I[n]$ is circularly-odd, then $X_I[k] = 0$ and hence $X[k]$ is purely real. On the other hand, if $x_I[n]$ is circularly-even, then $X_R[k] = 0$ and hence $X[k]$ is purely imaginary.

Complex-valued sequences Using (7.91) and (7.103) we can show that any complex-valued sequence can be decomposed as follows:

$$x[n] = x_R^{\text{ce}}[n] + x_R^{\text{co}}[n] + jx_I^{\text{ce}}[n] + jx_I^{\text{co}}[n]. \qquad (7.111)$$

A similar decomposition holds for the DFT coefficients

$$X[k] = X_R^{\text{ce}}[k] + X_R^{\text{co}}[k] + jX_I^{\text{ce}}[k] + jX_I^{\text{co}}[k]. \qquad (7.112)$$

Similarly to (7.103)–(7.105), any complex sequence $x[n]$ can be decomposed into a sum of a circularly-conjugate-even and a circularly-conjugate-odd component as

$$x[n] = x^{\text{cce}}[n] + x^{\text{cco}}[n], \qquad (7.113)$$

Table 7.2 Special cases of the DFT for real and imaginary signals. The symmetries below should be interpreted as circular symmetries

$$x[n] = x_R^{ce}[n] + x_R^{co}[n] + jx_I^{ce}[n] + jx_I^{co}[n]$$

$$X[k] = X_R^{ce}[k] + X_R^{co}[k] + jX_I^{ce}[k] + jX_I^{co}[k]$$

N-point sequence	*N*-point DFT
Real	real part is even - imaginary part is odd
Imaginary	real part is odd - imaginary part is even
Real and even	real and even
Real and odd	imaginary and odd
Imaginary and even	imaginary and even
Imaginary and odd	real and odd

where

$$x^{cce}[n] \triangleq \frac{x[n] + x^*[\langle -n \rangle_N]}{2} \tag{7.114a}$$

$$= \begin{cases} x_R[0], & n = 0 \\ \frac{1}{2}(x[n] + x^*[N - n]), & 1 \le n \le N - 1 \end{cases} \tag{7.114b}$$

and

$$x^{cco}[n] \triangleq \frac{x[n] - x^*[\langle -n \rangle_N]}{2} \tag{7.115a}$$

$$= \begin{cases} jx_I[0], & n = 0 \\ \frac{1}{2}(x[n] - x^*[N - n]). & 1 \le n \le N - 1 \end{cases} \tag{7.115b}$$

We note that $x^{cce}[n] = x_R^{ce}[n] + jx_I^{ce}[n]$ and $x^{cco}[n] = x_R^{co}[n] + jx_I^{co}[n]$. Using these decompositions and the uniqueness property of the DFT, we obtain all the symmetry properties summarized in Tables 7.2 and 7.3. These properties can be exploited to facilitate efficient computation of DFT in some practical applications.

DFTs of two real-valued sequences An interesting use of the symmetry and decomposition properties of the DFT is in an efficient computation of DFTs of two real-valued sequences using one DFT operation. Let $x_1[n]$ and $x_2[n]$ be two real-valued N-point sequences with DFTs $X_1[k]$ and $X_2[k]$, respectively. If we form a complex-valued sequence $x[n]$ as

$$x[n] = x_1[n] + jx_2[n], \tag{7.116}$$

Table 7.3 Symmetry properties of the DFT

N-point Sequence	*N*-point DFT
Complex signals	
$x^*[n]$	$X^*[\langle -k \rangle_N]$
$x^*[\langle -n \rangle_N]$	$X^*[k]$
$x_R[n]$	$X^{cce}[k] = \frac{1}{2}(X[k] + X^*[\langle -k \rangle_N])$
$jx_I[n]$	$X^{cco}[k] = \frac{1}{2}(X[k] - X^*[\langle -k \rangle_N])$
$x^{cce}[n] = \frac{1}{2}(x[n] + x^*[\langle -n \rangle_N])$	$X_R[k]$
$x^{cco}[n] = \frac{1}{2}(x[n] - x^*[\langle -n \rangle_N])$	$jX_I[k]$
Real signals	
{Any real $x[n]$}	$\begin{cases} X[k] = \tilde{X}^*[\langle -k \rangle_N] \\ X_R[k] = X_R[\langle -k \rangle_N] \\ X_I[k] = -X_I[\langle -k \rangle_N] \\ \|X[k]\| = \|X[\langle -k \rangle_N]\| \\ \angle X[k] = -\angle X[\langle -k \rangle_N] \end{cases}$

with DFT $X[k]$, then using Table 7.3, we can show that (see Tutorial Problem 12)

$$X_1[k] = X^{cce}[k] \quad \text{and} \quad jX_2[k] = X^{cco}[k]. \tag{7.117}$$

Thus, using one DFT computation followed by a conjugate-symmetry decomposition gives two required DFTs.

7.4.4   Circular shift of a sequence

Consider a sequence $x[n]$ with discrete-time Fourier transform $X(e^{j\omega})$. As we observed in Section 4.5.3, $e^{-j\omega m}X(e^{j\omega})$ is the Fourier transform of the time-shifted sequence $x[n-m]$. It is then natural to ask what happens to an N-point sequence $x[n]$ if we multiply its N-point DFT $X[k]$ by $W_N^{mk} = e^{-j(2\pi m/N)k}$. The result is a sequence $z[n]$ obtained by the inverse DFT of $Z[k] = W_N^{mk}X[k]$. Hence, we have

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]W_N^{-kn}, \tag{7.118}$$

$$z[n] = \frac{1}{N} \sum_{k=0}^{N-1} W_N^{kn}X[k]W_N^{-kn} = \frac{1}{N} \sum_{k=0}^{N-1} X[k]W_N^{-k(n-m)}. \tag{7.119}$$

A superficial comparison of (7.118) and (7.119) yields $z[n] = x[n-m]$. However, because $x[n]$ is unavailable outside the interval $0 \leq n \leq N-1$, we cannot obtain $z[n]$ by a simple
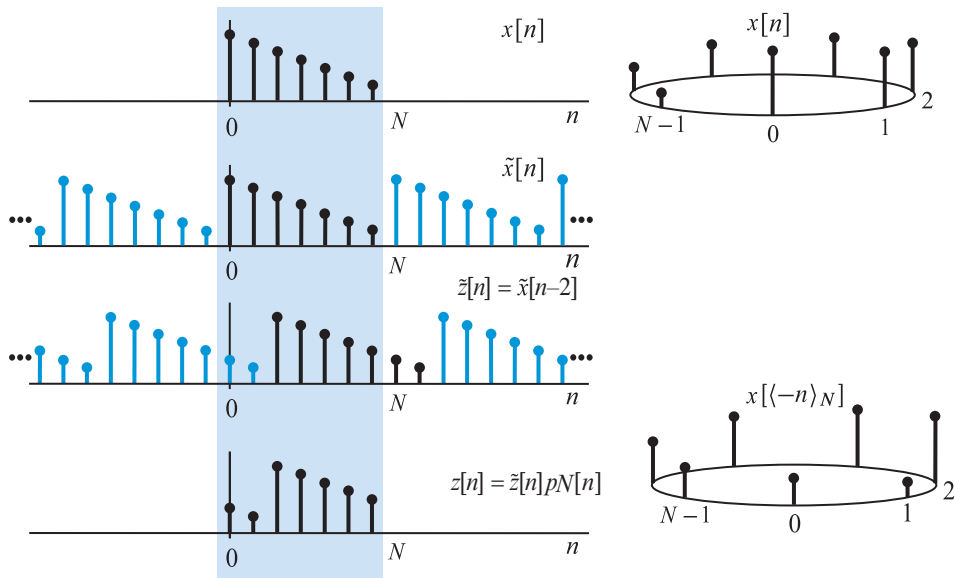
Figure 7.13 Circular shifting using the periodic extension and circular buffer interpretations.

```
function y=cirshift0(x,k,N)
% Circular shift of a sequence
if length(x) > N; error('N < length(x)'); end
x=[x zeros(1,N-length(x))];
n=(0:1:N-1); y=x(mod(n-k,N)+1);
```

Figure 7.14 MATLAB function for circular shifting of a finite-length sequence.

time shift of $x[n]$. To obtain the correct result, we recall that $W_N^{-k(n-m)} = W_N^{-k(n-m+\ell N)}$. This property implies that we can replace the index $(n - m)$ in (7.119) by the modulo-N index $\langle n - m \rangle_N$. Therefore, the correct answer is given by

$$z[n] = x[\langle n - m \rangle_N]. \quad 0 \le n \le N - 1 \tag{7.120}$$

This operation, which is called *circular shift* or *rotation*, is illustrated in Figure 7.13 using both the periodic extension and circular mapping interpretations. We note that if $m > 0$, $\tilde{x}[n - m]$ is shifted to the right and $x[\langle n - m \rangle_N]$ is rotated counterclockwise; the opposite operations take place when $m < 0$.

In conclusion, the circular shift property of the DFT is

$$x[\langle n - m \rangle_N] \xleftrightarrow[N]{\text{DFT}} W_N^{km} X[k]. \tag{7.121}$$

A MATLAB function called `cirshift0` to implement (7.121) is given in Figure 7.14.

7.4.5 Circular convolution

In Section 4.5.3, we showed that multiplying the DTFTs of two sequences corresponds to the DTFT of their *linear* convolution, that is,

$$y[n] = \sum_{m=-\infty}^{\infty} h[m]x[n-m] \xleftrightarrow{\text{DTFT}} \tilde{Y}(\text{e}^{\text{j}\omega}) = \tilde{H}(\text{e}^{\text{j}\omega})\tilde{X}(\text{e}^{\text{j}\omega}). \qquad (7.122)$$

In practice, the linear convolution sum is obtained by carrying out the summation. Computing the convolution sum by taking the inverse DTFT of the product of the two transforms cannot be carried out numerically. We now ask: what kind of operation corresponds to the multiplication of two N-point DFTs?

To answer this question we start by evaluating the product of two N-point DFTs $H[k]$ and $X[k]$. The result is the N-point DFT sequence

$$Y[k] = H[k]X[k]. \quad 0 \le k \le N-1 \qquad (7.123)$$

We wish to express the inverse DFT of $Y[k]$, which is an N-point sequence $y[n]$, in terms of the sequences $h[n]$ and $x[n]$. Using (7.21), (7.22), and (7.24), we obtain

$$
\begin{aligned}
y[n] &= \frac{1}{N}\sum_{k=0}^{N-1} H[k]X[k]W_N^{-kn} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\left[\sum_{m=0}^{N-1} h[m]W_N^{km}\right]\left[\sum_{\ell=0}^{N-1} x[\ell]W_N^{k\ell}\right]W_N^{-kn} \\
&= \sum_{m=0}^{N-1} h[m]\sum_{\ell=0}^{N-1} x[\ell]\left[\frac{1}{N}\sum_{k=0}^{N-1} W_N^{k(m+\ell-n)}\right].
\end{aligned}
\qquad (7.124)
$$

The last relation was obtained by rearranging the order of the three finite summations. Using the orthogonality condition (7.26), the last summation is given by

$$\frac{1}{N}\sum_{k=0}^{N-1} W_N^{k(m+\ell-n)} = \begin{cases} 1, & m+\ell-n = rN \\ 0. & \text{otherwise} \end{cases} \qquad (7.125)$$

Thus, the summations in (7.125) are nonzero if the index ℓ satisfies the condition

$$\ell = n - m + rN = (n-m) \text{ modulo } N = \langle n-m \rangle_N. \qquad (7.126)$$

In this case, we can substitute (7.126) into (7.124) to obtain the desired expression

$$y[n] = \sum_{m=0}^{N-1} h[m]x[\langle n-m \rangle_N]. \quad 0 \le n \le N-1 \qquad (7.127)$$

If we ignore the modulo-N operation, relation (7.127) resembles the linear convolution operation described in Section 2.4. However, the presence of the modulo operation

```
function y=circonv(h,x)
% Circular convolution of equal length sequences
N=length(x); y=zeros(N,1);
x=circfold(x,N);
y(1)=h'*x;
for n=2:N
    x=cirshift0(x,1,N);
    y(n)=h'*x;
end
```

Figure 7.15 MATLAB function for computation of circular convolution of two finite-length sequences of equal length.

$\langle m - n \rangle_N$ implies that the time-reversal (folding) and time-shifting operations required to obtain $x[\langle n - m \rangle_N]$ are performed in a circular manner. For this reason, the operation described by (7.127) is known as *circular convolution*. The N-point circular convolution (7.127) will be denoted by

$$y[n] \triangleq h[n] \, \textcircled{N} \, x[n], \tag{7.128}$$

to emphasize the effect of N on the convolution operation in (7.126). A MATLAB function called `circonv` that uses the `circfold` and `cirshift0` functions is given in Figure 7.15. In summary, the circular convolution property of the DFT is

$$y[n] = h[n] \, \textcircled{N} \, x[n] \xleftarrow[N]{\text{DFT}} Y[k] = H[k]X[k]. \tag{7.129}$$

The operations of circular folding and circular shift were illustrated in Figures 7.10 and 7.13, respectively. Circular shift is further illustrated in Figure 7.16 using the concept of a circular buffer. The N signal samples are stored in a circular fashion as illustrated in 7.16(a) for $N = 8$. To obtain $x[\langle n - 2 \rangle_N]$ we can physically shift the samples counterclockwise by two locations as shown in 7.16(b). However, as shown in 7.16(c), circular buffers can avoid this time consuming operation by adjusting the start address pointer without physically shifting any data samples in memory. This operation, which is known as *circular* or *modulo addressing*, is widely used in digital signal processors.

The mechanics of circular convolution are shown in the following examples.

Example 7.5   Computing circular convolution in the time-domain

Determine the 4-point circular convolution of the following 4-point sequences:

$$h[n] = \{h[0] \ h[1] \ h[2] \ h[3]\} = \{1 \ 0 \ 1 \ 1\},$$
$$x[n] = \{x[0] \ x[1] \ x[2] \ x[3]\} = \{0 \ 1 \ 2 \ 3\}.$$
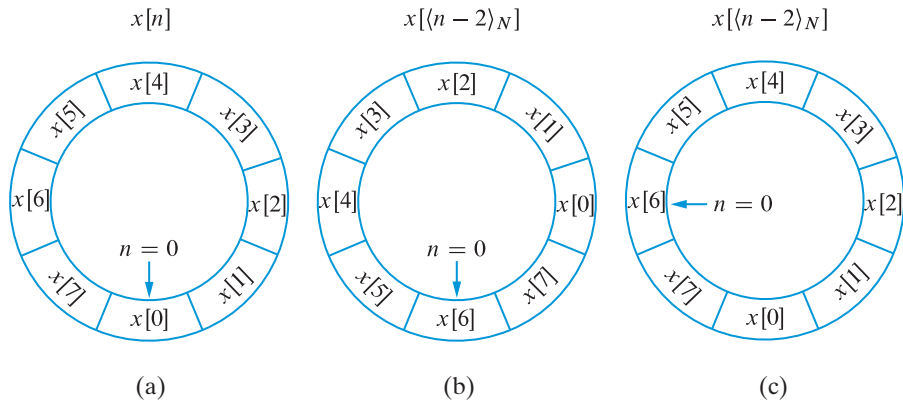
Figure 7.16 Circular shifting operation approaches: (a) sequence $x[n]$, (b) sequence $x[\langle n - 2\rangle_N]$ using circular mapping, and (c) sequence $x[\langle n - 2\rangle_N]$ using circular addressing.
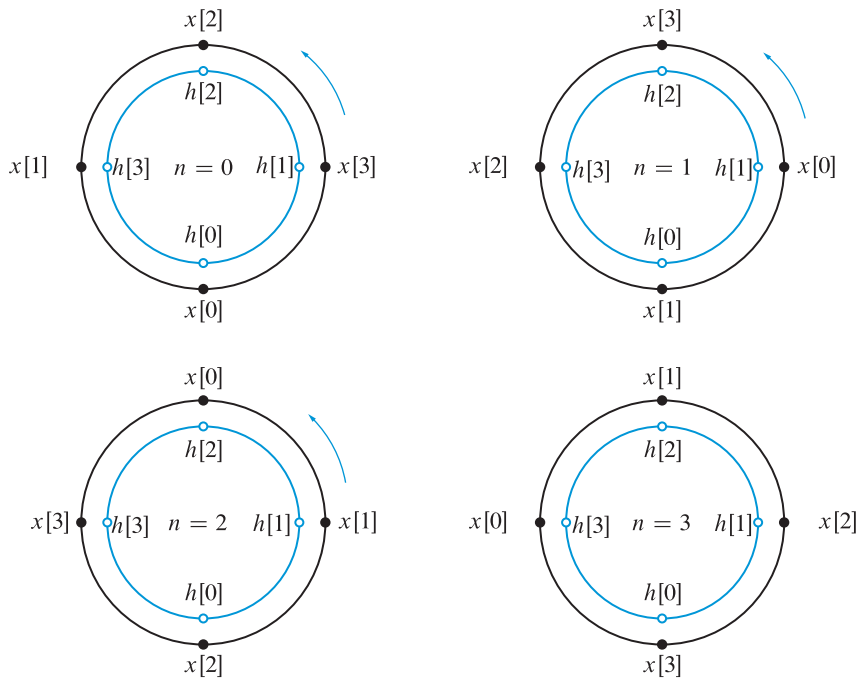
Figure 7.17 Graphical interpretation of computation of circular convolution. Note that the sequence $h[m]$ (fixed) is arranged counterclockwise, whereas the "rotating" sequence $x[m]$ is arranged clockwise because of the required circular time-reversal (folding) operation.

Solution The first step is to change the time index from n to m. For $n = 0$ we need the samples of sequences $h[m]$ and $x[\langle -m\rangle_4]$, which we arrange on two concentric circles as shown in Figure 7.17. As expected, the samples of the time-reversed sequence are placed counterclockwise. The sum of pairwise products gives the value of circular convolution for $n = 0$, that is

$$y[0] = h[0]x[0] + h[1]x[3] + h[2]x[2] + h[3]x[1] = 3. \tag{7.130}$$

For $n = 1$, we rotate $x[\langle -m \rangle_4]$ by one sample to obtain $x[\langle 1 - m \rangle_4]$. This yields

$$y[1] = h[0]x[1] + h[1]x[0] + h[2]x[3] + h[3]x[2] = 6. \tag{7.131}$$

The values of circular convolution for $n = 2, 3$ are obtained in a similar manner

$$y[2] = h[0]x[2] + h[1]x[1] + h[2]x[0] + h[3]x[3] = 5, \tag{7.132}$$

$$y[3] = h[0]x[3] + h[1]x[2] + h[2]x[1] + h[3]x[0] = 4. \tag{7.133}$$

The 4-point circular convolution equations (7.130)–(7.133) can be written in matrix form as

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \end{bmatrix} = \begin{bmatrix} x[0] & x[3] & x[2] & x[1] \\ x[1] & x[0] & x[3] & x[2] \\ x[2] & x[1] & x[0] & x[3] \\ x[3] & x[2] & x[1] & x[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ h[2] \\ h[3] \end{bmatrix}. \tag{7.134}$$

We note that the first row of the matrix is obtained by circularly reversing the sequence $x[n]$. To obtain the second row, we shift the first row one entry to the right. The last entry $x[1]$, which exits the matrix row from the right, enters the row from the left; this is essentially the meaning of circular shift. The third and forth columns are generated in a similar manner. A matrix generated by this process is called a *circulant* matrix. We note that every circulant matrix is Toeplitz, that is, all elements on the diagonals parallel to the main diagonal are equal; however, a Toeplitz matrix is not necessarily circulant. It is easy to see that the circulant matrix in (7.134) can also be generated by circularly shifting its first column (that is, the sequence $x[n]$). Finally, it is easy to show that (7.130)–(7.133) can be expressed in matrix form by using $h[n]$ to form the circulant matrix and $x[n]$ the right hand side vector.

In MATLAB the matrix form (7.134) of circular convolution can be implemented using the function `toeplitz` as follows:

$$\texttt{y=toeplitz(x,circfold(x,N))*h}. \tag{7.135}$$

The reader can use this function to verify the results given by (7.130)–(7.133). ∎

Example 7.6   Computation of circular convolution using the DFT
Compute the 4-point circular convolution of the sequences in Example 7.5 using the DFT:

$$h[n] = \{h[0]\ h[1]\ h[2]\ h[3]\} = \{1\ 0\ 1\ 1\},$$

$$x[n] = \{x[0]\ x[1]\ x[2]\ x[3]\} = \{0\ 1\ 2\ 3\}.$$

Solution The first step is to compute the 4-point DFTs of the two sequences. This is done using the matrix-by-vector multiplication formulas given in Example 7.2. The 4-point DFT of $x[n]$ is obtained by (see Example 7.2)

$$\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 6 \\ -2+j2 \\ -2 \\ -2-j2 \end{bmatrix}.$$

Similarly, the 4-point DFT of $h[n]$ is given by

$$\begin{bmatrix} H[0] \\ H[1] \\ H[2] \\ H[3] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ j \\ 1 \\ -j \end{bmatrix}.$$

In the second step we compute the product $Y[k] = H[k]X[k]$, $k = 0, 1, 2, 3$ of the two DFTs. The result is the 4-point DFT

$$Y[0] = 18, \; Y[1] = -2 - j2, \; Y[2] = -2, \; Y[3] = -2 + j2.$$

The circular convolution is obtained by computing the inverse DFT of $Y[k]$ using the formula $y = (1/4)W_4^* Y$ (see Example 7.2). The result is

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & j & -1 & -j \\ 1 & -1 & 1 & -1 \\ 1 & -j & -1 & j \end{bmatrix} \begin{bmatrix} 18 \\ -2-j2 \\ -2 \\ -2+j2 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 5 \\ 4 \end{bmatrix}.$$

The matrix approach presented in this example is used for pedagogical purposes. In practice, we use the following MATLAB implementation

$$y = \mathtt{ifft(fft(h).*fft(x))}, \tag{7.136}$$

which can be used to verify the results of this example.   ∎

Although circular convolution is the natural operation when working with DFTs, the operation required by signal processing applications is linear convolution. A logical question is, can we use circular convolution to obtain linear convolution? The answer to this question, which is yes, is discussed in Section 7.5.

7.4.6   Circular correlation

The circular correlation of two N-point sequences $x[n]$ and $y[n]$ is defined by

$$r_{xy}[\ell] \triangleq \sum_{n=0}^{N-1} x[n]y[\langle n - \ell \rangle_N]. \quad 0 \le \ell \le N-1 \tag{7.137}$$

Comparison of (7.137) and (7.127) reveals that circular correlation requires circular shifting but not circular folding. Using this idea, we can show that (see Tutorial Problem 16)

$$r_{xy}[\ell] = x[n] \, \textcircled{N} \, y[\langle -n \rangle_N] \xleftrightarrow[N]{\text{DFT}} R_{xy}[k] = X[k]Y^*[k]. \tag{7.138}$$

The definition (7.137) also provides the correlation of periodic sequences with the same period. Correlation of aperiodic sequences (linear correlation) and its meaning has been discussed in Section 4.5.4.

7.4.7   DFT of stretched and sampled sequences

Given a sequence $x[n]$, $0 \le n \le N - 1$, we can construct a new sequence $x^{(L)}[n]$ by inserting $(L-1)$ zeros between consecutive samples. This *stretched* or *expanded* sequence is formally defined by

$$x^{(L)}[n] \triangleq \begin{cases} x[n/L], & n = 0, L, \ldots, (N-1)L \\ 0. & \text{otherwise.} \end{cases} \tag{7.139}$$

The LN-point DFT of the stretched sequence $x^{(L)}[n]$ is given by L consecutive periods of the DFS $\tilde{X}[k]$. Similarly, the inverse DFT of the stretched sequence $X^{(L)}[k]$ is equal to L consecutive periods of $\tilde{x}[n]$ (see Tutorial Problem 17). That is, we have

$$x^{(L)}[n] \xleftrightarrow[NL]{\text{DFT}} \tilde{X}[k] = X[\langle k \rangle_N], \tag{7.140}$$

$$\frac{1}{L} x[\langle n \rangle_N] = \frac{1}{L} \tilde{x}[n] \xleftrightarrow[NL]{\text{DFT}} X^{(L)}[k]. \tag{7.141}$$

Given a sequence $x[n]$, $0 \le n \le N - 1$ and a positive integer M which divides N, we define the sampled sequence $x_{(M)}[n]$ by

$$x_{(M)}[n] = x[nM]. \quad 0 \le n \le \frac{N}{M} - 1 \tag{7.142}$$

The (N/M)-point DFT of $x_{(M)}[n]$ is obtained by overlapping and summing the DFT of $x[n]$ using the formula (see Tutorial Problem 18)

$$x_{(M)}[n] \xleftrightarrow[N/M]{\text{DFT}} \frac{1}{M} \sum_{m=0}^{M-1} X\left[k + m\frac{N}{M}\right]. \tag{7.143}$$

Similarly, the inverse DFT of $X_{(M)}[k]$ is obtained by overlapping and summing the sequence $x[n]$ as follows

$$\frac{1}{M} \sum_{m=0}^{M-1} x\left[n + m\frac{N}{M}\right] \xleftrightarrow[N/M]{\text{DFT}} X_{(M)}[k]. \tag{7.144}$$

The overlapping and summing operations in (7.143) and (7.144) may result in frequency-domain aliasing or time-domain aliasing, respectively. These properties are useful for the understanding and interpretation of fast algorithms used for the computation of DFT.

Table 7.4 Operational properties of the DFT

	Property	N-point sequence	N-point DFT
		$x[n], h[n], v[n]$	$X[k], H[k], V[k]$
		$x_1[n], x_2[n]$	$X_1[k], X_2[k]$
1.	Linearity	$a_1 x_1[n] + a_2 x_2[n]$	$a_1 X_1[k] + a_2 X_2[k]$
2.	Time shifting	$x[\langle n - m \rangle_N]$	$W_N^{km} X[k]$
3.	Frequency shifting	$W_N^{-mn} x[n]$	$X[\langle k - m \rangle_N]$
4.	Modulation	$x[n] \cos(2\pi/N)k_0 n$	$\frac{1}{2} X[\langle k + k_0 \rangle_N] + \frac{1}{2} X[\langle k - k_0 \rangle_N]$
5.	Folding	$x[\langle -n \rangle_N]$	$X^*[k]$
6.	Conjugation	$x^*[n]$	$X^*[\langle -k \rangle_N]$
7.	Duality	$X[n]$	$Nx[\langle -k \rangle]_N]$
8.	Convolution	$h[n] \, \text{Ⓝ} \, x[n]$	$H[k]X[k]$
9.	Correlation	$x[n] \, \text{Ⓝ} \, y[\langle -n \rangle_N]$	$X[k]Y^*[k]$
10.	Windowing	$v[n]x[n]$	$\frac{1}{N} V[k] \, \text{Ⓝ} \, X[k]$
11.	Parseval's theorem	$\displaystyle\sum_{n=0}^{N-1} x[n]y^*[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]Y^*[n]$	
12.	Parseval's relation	$\displaystyle\sum_{n=0}^{N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |X[k]|^2$	

7.4.8   Summary of properties of the DFT

Table 7.4 summarizes the operational properties of the DFT; the symmetry properties were summarized in Table 7.3. We emphasize that the presence of the modulo-N indexing operator ensures that all sequences and their DFTs are specified in the range from 0 to $N - 1$. The fact that the DFT and inverse DFT formulas differ only in a factor of $1/N$ and in the sign of the exponent of W_N, results in a strong duality between the two transforms. To establish this duality, we first replace n by $-n$ in the inverse DFT formula to obtain

$$Nx[-n] = \sum_{k=0}^{N-1} X[k] W_N^{kn}. \tag{7.145}$$

Interchanging the roles of n and k in (7.145) yields

$$Nx[-k] = \sum_{n=0}^{N-1} X[n] W_N^{nk}. \tag{7.146}$$

The sum in (7.146) is the DFT of the N-point sequence obtained from the DFT coefficients of $x[n]$. To ensure that $x[-k]$ is specified in the range $0 \leq k \leq N-1$, we use the modulo-N operator. This leads to the following duality property:

$$x[n] \xleftrightarrow[N]{\text{DFT}} X[k] \Rightarrow X[n] \xleftrightarrow[N]{\text{DFT}} Nx[\langle -k \rangle_N]. \tag{7.147}$$

Duality can be used to derive several properties of the DFT. For example, it can be shown (see Tutorial Problem 19) that the DFT of the product of two N-point sequences is the circular convolution of their DFTs, that is,

$$v[n]x[n] \xleftrightarrow[N]{\text{DFT}} \frac{1}{N}V[k] \, \textcircled{N} \, X[k]. \tag{7.148}$$

7.5   Linear convolution using the DFT

In this section we discuss techniques for the computation of linear convolution using circular convolution implemented using the DFT, as shown in Figure 7.18. This approach is computationally more efficient for FIR filters with long impulse responses if we use fast algorithms for computation of the DFT (see Chapter 8).

7.5.1   Linear convolution using circular convolution

The linear convolution of two finite-length sequences, say, $x[n]$, $0 \leq n \leq L-1$ and $h[n]$, $0 \leq n \leq M-1$, is a sequence $y[n]$ of length $L+M-1$, given by

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]. \quad 0 \leq n \leq L+M-2 \tag{7.149}$$

The convolution sequence $y[n]$ has Fourier transform given by

$$Y(e^{j\omega}) = H(e^{j\omega})X(e^{j\omega}). \tag{7.150}$$
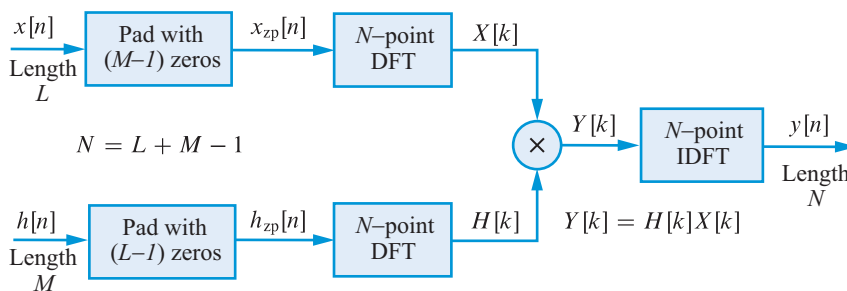
Figure 7.18 Computation of the linear convolution $y[n] = h[n] * x[n]$ of two finite-length sequences $h[n]$ and $x[n]$ using the DFT.

If we sample $Y(e^{j\omega})$ at frequencies $\omega_k = 2\pi k/N$, where $N \geq L + M - 1$, we can uniquely recover the sequence $y[n]$, $0 \leq n \leq N - 1$, from the DFT coefficients

$$Y[k] = Y(e^{j2\pi k/N}). \quad 0 \leq k \leq N - 1 \tag{7.151}$$

The inverse DFTs of $H[k] = H(e^{j2\pi k/N})$ and $X[k] = X(e^{j2\pi k/N})$ yield the sequences $h[n]$ and $x[n]$ padded with $(N - M)$ and $(N - L)$ zeros, respectively (see Section 7.3.3). Therefore, (7.150) leads to the following DFT pair:

$$y_{zp}[n] = h_{zp}[n] \, \textcircled{N} \, x_{zp}[n] \xleftarrow[N]{\text{DFT}} Y[k] = H[k]X[k]. \tag{7.152}$$

Note that if $N \geq L + M - 1$, $y[n] = y_{zp}[n]$, $0 \leq n \leq L + M - 2$, that is, circular convolution is identical to linear convolution. If $N < L + M - 1$, due to time-domain aliasing, $y_{zp}[n]$ may not be equal to $y[n]$ for some or all values of n (see Tutorial Problem 15). The computation of linear convolution of two finite-length sequences using the DFT is illustrated in Figure 7.18.

To appreciate this result, we recall that the inverse DFT of $Y[k]$ yields not $y[n]$ but its periodic replication $\tilde{y}[n]$,

$$\tilde{y}[n] = \sum_{\ell=-\infty}^{\infty} y[n - \ell N] \xleftarrow[N]{\text{DFT}} Y[k] = H[k]X[k]. \tag{7.153}$$

As explained in Section 7.3.2, the sequence $y[n]$ can be fully recovered from $\tilde{y}[n]$ only if $N \geq L + M - 1$; otherwise, some or all samples may suffer aliasing distortion. A MATLAB function called `circonvtfft` that computes linear convolution using the DFT approach in (7.153) is given in Figure 7.19.

The length M of the impulse response at which the DFT based approach is more efficient than direct computation of convolution depends on the hardware and software available to implement the computations.

A matrix approach interpretation As we discussed in Section 2.7, linear convolution can be expressed in matrix form as a matrix by vector multiplication. For $M = 3$ and $L = 5$, the convolution sum (7.149) can be written as

```
function y=circonvfft(h,x)
% Computation of linear convolution using the FFT algorithm

N=length(h)+length(x)-1;
y=ifft(fft(h,N).*fft(x,N));
```

Figure 7.19 MATLAB function for computation of linear convolution using the DFT.

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \end{bmatrix} = \begin{bmatrix} x[0] & 0 & 0 \\ x[1] & x[0] & 0 \\ x[2] & x[1] & x[0] \\ x[3] & x[2] & x[1] \\ x[4] & x[3] & x[2] \\ 0 & x[4] & x[3] \\ 0 & 0 & x[4] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ h[2] \end{bmatrix}. \tag{7.154}$$

The output sequence $y[n]$ has length $N = M + L - 1 = 7$ samples. We note that the first column of the Toeplitz matrix in (7.154) is the L-point sequence $x[n]$ padded with $(M - 1)$ zeros. The second column is obtained by circular shifting of the first column "down;" the element leaving the bottom enters from the top. Similarly, circular shift of the second column yields the third column. If we repeat this process $(L - 1)$ times and we append the M-point sequence $h[n]$ with $(L - 1)$ zeros, we obtain the following matrix equation

$$\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \end{bmatrix} = \begin{bmatrix} x[0] & 0 & 0 & x[4] & x[3] & x[2] & x[1] \\ x[1] & x[0] & 0 & 0 & x[4] & x[3] & x[2] \\ x[2] & x[1] & x[0] & 0 & 0 & x[4] & x[3] \\ x[3] & x[2] & x[1] & x[0] & 0 & 0 & x[4] \\ x[4] & x[3] & x[2] & x[1] & x[0] & 0 & 0 \\ 0 & x[4] & x[3] & x[2] & x[1] & x[0] & 0 \\ 0 & 0 & x[4] & x[3] & x[2] & x[1] & x[0] \end{bmatrix} \begin{bmatrix} h[0] \\ h[1] \\ h[2] \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{7.155}$$

The $N \times N$ matrix in (7.155) is circulant, because the first row is the circular fold of the first column. Every other row is obtained by circularly shifting the previous row one position to the right. Thus, the matrix multiplication in (7.155) yields the circular convolution of the zero padded sequences defined by the first column of the matrix and the right-hand side vector. However, careful inspection of (7.155) reveals that the $(L - 1)$ blue (padded) zeros in $h[n]$ *cancel* the contribution of the last $(L - 1)$ blue columns of the circulant matrix. Therefore, the circular convolution (7.152) yields the same results with the linear convolution (7.149).

7.5.2 Implementation of FIR filters using the DFT

The implementation of an FIR filter using the DFT, as shown in Figure 7.18, may not be practically feasible for the following reasons:

- In many practical applications, like speech processing and communications, the input sequence may have *indefinite* length.
- The length of the input sequence may be *too large* for storage and computation for the DFT to be practical.
- The computation of the output sequence cannot be started until *all* input signal samples have been collected. This may cause unacceptable delays for many applications.

To overcome these problems we can segment the input sequence into blocks of length Q and use one of the methods of block convolution described below.

The overlap–add method The output of an FIR filter with impulse response $h[n]$ of length $M = 3$ to an input sequence $x[n]$ with length $L = 8$ is given by

$$
\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \\ y[8] \\ y[9] \end{bmatrix} = \begin{bmatrix} h[0] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h[1] & h[0] & 0 & 0 & 0 & 0 & 0 & 0 \\ h[2] & h[1] & h[0] & 0 & 0 & 0 & 0 & 0 \\ 0 & h[2] & h[1] & h[0] & 0 & 0 & 0 & 0 \\ 0 & 0 & h[2] & h[1] & h[0] & 0 & 0 & 0 \\ 0 & 0 & 0 & h[2] & h[1] & h[0] & 0 & 0 \\ 0 & 0 & 0 & 0 & h[2] & h[1] & h[0] & 0 \\ 0 & 0 & 0 & 0 & 0 & h[2] & h[1] & h[0] \\ 0 & 0 & 0 & 0 & 0 & 0 & h[2] & h[1] \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & h[2] \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}.
$$

(7.156)

Suppose now that we segment the input sequence $x[n]$ into two blocks (black and blue) of length $Q = 4$. The big matrix by vector multiplication in (7.156) can be split into two smaller ones: the black part provides the output samples $y[0], \ldots, y[5]$, and the blue part the samples $y[4], \ldots, y[9]$. To compute samples $y[4]$ and $y[5]$, we need to add the contributions from both the black and blue blocks. This overlap of successive output blocks, by $(M - 1)$ samples, led to the name "overlap and add" for this method of block convolution. Each block convolution can be implemented using the DFT approach in Figure 7.20.

The overlap–save method In the overlap–add method, we need the output of the next block to complete the computation of the current block. To avoid this drawback, we allow successive input segments to overlap by $(M - 1)$ samples. To explain this approach, we compute the convolution in (7.149) using input blocks of length $Q = 5$ that overlap by $M - 1 = 2$ samples. A graphical illustration is obtained by partitioning (7.156) as follows

```
function y=overlap_add(h,x,Q)
% Overlap-Add Method

L=length(x); M=length(h);
Nblocks=floor(L/Q);
ni=(1:Q)'; no=(1:M+Q-1)';
y=zeros(L+M-1,1);

y(no)=conv(h,x(ni));
for m=1:Nblocks-1
    z=conv(h,x(m*Q+ni));
    y(m*Q+no)= y(m*Q+no)+z;
end
```

Figure 7.20 MATLAB function for the overlap and add method.

$$
\begin{bmatrix} y[0] \\ y[1] \\ y[2] \\ y[3] \\ y[4] \\ y[5] \\ y[6] \\ y[7] \\ y[8] \\ y[9] \end{bmatrix}
=
\begin{bmatrix}
h[0] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
h[1] & h[0] & 0 & 0 & 0 & 0 & 0 & 0 \\
h[2] & h[1] & h[0] & 0 & 0 & 0 & 0 & 0 \\
0 & h[2] & h[1] & h[0] & 0 & 0 & 0 & 0 \\
0 & 0 & h[2] & h[1] & h[0] & 0 & 0 & 0 \\
0 & 0 & 0 & h[2] & h[1] & h[0] & 0 & 0 \\
0 & 0 & 0 & 0 & h[2] & h[1] & h[0] & 0 \\
0 & 0 & 0 & 0 & 0 & h[2] & h[1] & h[0] \\
0 & 0 & 0 & 0 & 0 & 0 & h[2] & h[1] \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & h[2]
\end{bmatrix}
\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}.
$$

$$(7.157)$$

To understand the basic idea, we note that the black matrix-by-vector multiplication provides the correct output values for $y[0], y[1], y[2], y[3], y[4]$. The blue matrix by vector multiplication provides wrong values for $y[3]$ and $y[4]$, and correct values for $y[5], y[6], y[7]$. However, there is no problem with that because $y[3]$ and $y[4]$ have been correctly computed in the previous block. The $Q \times Q$ impulse response matrix used in the block convolutions can be easily modified to become circulant. Indeed, if we change the last zeros in the first $M - 1 = 2$ rows of this matrix (see blue entries), we obtain the following circular convolution:

$$
\begin{bmatrix} y_{\text{cir}}[3] \\ y_{\text{cir}}[4] \\ y[5] \\ y[6] \\ y[7] \end{bmatrix}
=
\begin{bmatrix}
h[0] & 0 & 0 & h[2] & h[1] \\
h[1] & h[0] & 0 & 0 & h[2] \\
h[2] & h[1] & h[0] & 0 & 0 \\
0 & h[2] & h[1] & h[0] & 0 \\
0 & 0 & h[2] & h[1] & h[0]
\end{bmatrix}
\begin{bmatrix} x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}.
$$

$$(7.158)$$

The last $Q - (M - 1) = 3$ samples of the circular convolution are equal to the required samples of linear convolution. The first $M - 1 = 2$ samples have been calculated in the previous step. To obtain the values $y[0], y[1], \ldots, y[Q - M - 1]$, we start the process with an artificial input block obtained by replacing the last $Q + 1 - M = 3$ elements of the first input block $x[0], x[1], \ldots, x[Q - 1]$ by zeros. This approach, which is known as the *overlap–save method*, is implemented using the MATLAB function shown in Figure 7.21. The method is called overlap–save because the input segments overlap so that each $Q \times Q$ block circular convolution computes $(Q - M + 1)$ new output samples and uses $(M - 1)$ output samples from the previous block.

7.6   Fourier analysis of signals using the DFT

As we noted in Section 7.1, the DFT can be used to compute (either exactly or approximately) any Fourier transform or series. Thus, the DFT, through its efficient FFT-algorithm implementation, provides the fundamental computational tool for practical Fourier analysis. The application of DFT requires three steps: (a) sample the continuous-time signal, (b) select a finite number of samples for analysis, and (c) compute the spectrum at a finite number of frequencies.

```
function y=overlap_save(h,x,Q)
% Overlap-Save Method

L=length(x); M=length(h);
if Q >= M; else end;
P=Q-M+1; % Overlap
Nblocks=floor((L-Q)/P+1);
y=zeros(L+M-1,1);

y(1:Q)=cconv(h,x(1:Q-M),Q);

z=cconv(h,x(1:Q),Q);
y(P:Q)=z(P:Q);
for m=1:Nblocks-1
    z=cconv(h,x(m*P+(1:Q)),Q);
    y(Q+(m-1)*P+(1:M))= z(P:Q);
end
```

Figure 7.21 MATLAB function for the overlap and save method.

The effects of time-domain sampling have been extensively studied in Chapter 6. For the remainder of this section we assume that sampling has been done satisfactorily and that the effects of aliasing are negligible. The effects of the frequency-domain sampling inherent in the DFT have been discussed in Section 7.3. In this section, we concentrate specifically on the effects of the selection of a finite-length segment of the sampled signal, because it has the biggest impact on the accuracy of the estimated spectrum.

7.6.1   Effects of time-windowing on sinusoidal signals

The operation of selecting a finite number of samples is equivalent to multiplying the actual sequence $x[n]$, defined in the range $-\infty < n < \infty$, by a finite-length sequence $w[n]$ called a *data window* or simply a *window*. For example, if we define the rectangular window

$$w[n] \triangleq \begin{cases} 1, & 0 \leq n \leq L-1 \\ 0, & \text{otherwise} \end{cases} \tag{7.159}$$

the "windowing" operation yields a sequence

$$\hat{x}[n] = w[n]x[n], \tag{7.160}$$

which consists of L samples from $x[n]$. The term "windowing" is used to emphasize the fact that we can only "see" a finite part of the signal through a "window." Therefore, *the DFT provides samples of the DTFT of the windowed signal $\hat{x}[n] = w[n]x[n]$, not of the original signal $x[n]$*. This interpretation is illustrated in Figure 7.22(a). To understand how the
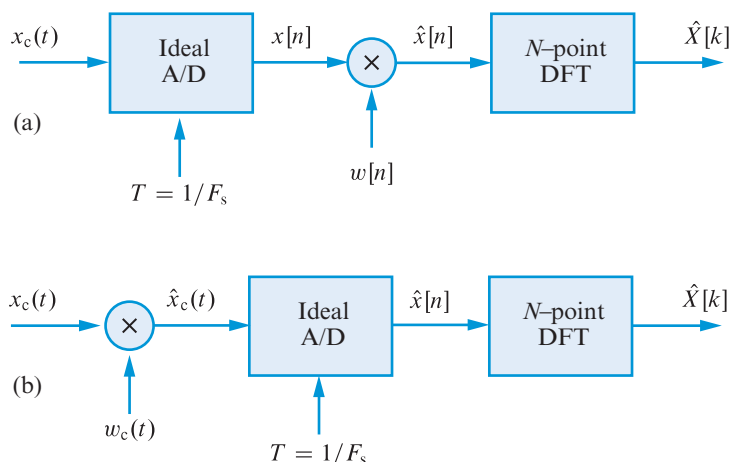
Figure 7.22 Equivalent systems for spectrum analysis using the DFT. The two outputs are identical if the windows satisfy the condition $w[n] = w_c(nT)$.

operation of time-windowing changes the spectrum of the original signal we interchange the order of sampling and windowing operations as shown in Figure 7.22(b). Clearly, the window $w_c(t)$ should be nonzero in the interval $0 \le t \le T_0$, where $(L-1)T < T_0 < LT$. The two systems are equivalent as long as the continuous-time window $w_c(t)$ satisfies the condition $w[n] = w_c(nT)$. Indeed, sampling the continuous-time windowed signal

$$\hat{x}_c(t) = w_c(t)x_c(t) \tag{7.161}$$

yields the windowed sequence

$$\hat{x}[n] = \hat{x}_c(nT) = w_c(nT)x_c(nT) = w[n]x[n], \tag{7.162}$$

which is identical to the one given by (7.160) (see Tutorial Problem 20 for another proof).

   The continuous-time windowing operation (7.161) can be used to understand the effects of a finite observation interval, in terms of the physical time and frequency variables, without interference from the subsequent time and frequency sampling operations. Furthermore, we avoid the complications of dealing with periodic spectra. We shall introduce the spectral effects of windowing progressively from simpler models to more complicated ones. We begin with a single frequency sinusoidal signal and obtain its windowed spectrum to study the effect of windowing. We then model a bandpass signal as a sum of isolated sinusoids with narrowly separated frequencies and study its windowed spectrum. In the limit this model then leads to the spectrum of a windowed aperiodic signal as a convolution between the given signal spectrum and the window spectrum.

   We can illustrate the effects of time-windowing using the sinusoidal signal

$$x_c(t) = A_1 \cos(\Omega_1 t + \phi_1). \quad -\infty < t < \infty \tag{7.163}$$

The spectrum of this signal, as we can see from the following expansion, consists of two spectral lines with complex amplitude $\frac{1}{2}A_1 e^{\pm j\phi_1}$ at frequencies $\Omega = \Omega_1$ and $\Omega = -\Omega_1$:

$$x_c(t) = \frac{1}{2}A_1 e^{-j\phi_1} e^{-j\Omega_1 t} + \frac{1}{2}A_1 e^{j\phi_1} e^{j\Omega_1 t}. \tag{7.164}$$

The time-windowed signal $\hat{x}_c(t)$ is given by

$$\hat{x}_c(t) = w_c(t)x_c(t) = \frac{1}{2}A_1 e^{-j\phi_1} w_c(t)e^{-j\Omega_1 t} + \frac{1}{2}A_1 e^{j\phi_1} w_c(t)e^{j\Omega_1 t}. \tag{7.165}$$

Taking the CTFT of (7.165) and using the frequency-shift property, we have

$$\hat{X}_c(j\Omega) = \frac{1}{2}A_1 e^{-j\phi_1} W_c(j(\Omega + \Omega_1)) + \frac{1}{2}A_1 e^{j\phi_1} W_c(j(\Omega - \Omega_1)), \tag{7.166}$$

where $W_c(j\Omega)$ is the CTFT of the window. As we can see from (7.166), the effect of time-windowing is to replace each line of the discrete spectrum with a scaled copy of $W_c(j\Omega)$ centered at the frequency of the line. The rectangular window

$$w_c(t) \triangleq w_R(t) = \begin{cases} 1, & 0 \le t \le T_0 \\ 0, & \text{otherwise} \end{cases} \tag{7.167}$$

has a CTFT given by

$$W_c(j\Omega) \triangleq W_R(j\Omega) = \frac{\sin(\Omega T_0/2)}{\Omega/2} e^{-j\Omega T_0/2}. \tag{7.168}$$

This definition of the rectangular window is useful for causal signals that start at $t = 0$ and it is compatible with the DFT range that starts at $n = 0$. However, we can use any shifted version of $w_c(t)$; the only difference is going to be a linear-phase shift in $W_c(j\Omega)$. To simplify the pictorial illustrations we shift the rectangular window in the interval $-T_0/2 \le t \le T_0/2$. The result is a symmetric window with real Fourier transform $W_R(j\Omega) = \sin(\Omega T_0/2)/(\Omega/2)$.

The effects of time-windowing on the spectrum of a sinusoidal signal are illustrated in Figure 7.23. The spectrum of the infinite duration sinusoidal signal consists of two lines at frequencies $\Omega = \pm\Omega_1$; hence, as shown in Figure 7.23(a), its power is perfectly "localized" to only two points of the frequency axis. Figure 7.23(b) shows a symmetric rectangular window and its Fourier transform. The shape of $W_R(j\Omega)$ is typical for most windows of practical interest. There is one large peak, or "mainlobe," at the origin together with a series of subsidiary or spurious peaks of decreasing magnitude, or "sidelobes," on both sides of the origin. The mainlobe has zero-crossings at multiples of $2\pi/T_0$ and an approximate 3-dB bandwidth of $2\pi/T_0$ rads/s. Each line of the original spectrum is replaced by a copy of $W_R(j\Omega)$ scaled by the amplitude of the corresponding complex exponential, as shown in Figure 7.23(c). The sum of the scaled and shifted copies yields the Fourier transform of the windowed sinusoidal signal, $\hat{X}_c(j\Omega)$, which is shown in Figure 7.23(d). We note that sidelobes adding in (out of) phase can increase (reduce) the heights of the peaks. For this reason, the left (right) pulse in Figure 7.23(d) is not symmetric about $\Omega = -\Omega_1$ ($\Omega = \Omega_1$).
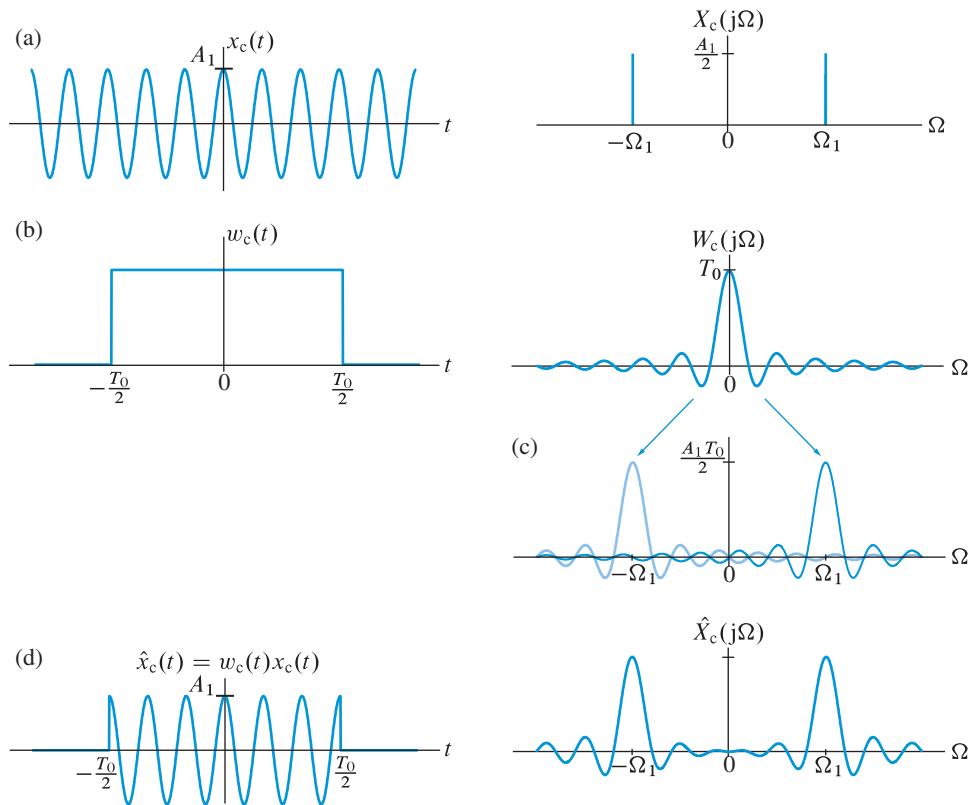
Figure 7.23 The effects of rectangular windowing (truncation) on the spectrum of a sinusoidal signal. In this case, windowing can be interpreted as modulation of a sinusoidal carrier by the window function.

Comparing the spectra of $x_c(t)$ and $\hat{x}_c(t) = w_c(t)x_c(t)$ reveals the effects of rectangular windowing (truncation). First, we note that the spectral lines of $X_c(j\Omega)$, which have zero width, have been "spread out" or "smeared" in $\hat{X}_c(j\Omega)$. The amount of spread is determined by the width of the mainlobe of the window spectrum. This effect is known as *spectral spreading* or *smearing*. Second, we see that although the spectrum of $x_c(t)$ is zero everywhere except at $\Omega = \pm\Omega_1$, the spectrum of $\hat{x}_c(t)$ is zero nowhere because of the sidelobes. This effect, which is called *leakage*, causes transfer of power from one band to another. Usually, the problem is spectral leakage from a "strong" band, where the spectrum is large, to a "weak" band, where the spectrum is small or zero. Leakage creates "false" peaks, that is, peaks at wrong frequencies, nonexisting peaks, or changes the amplitude of existing peaks.

The smearing or blurring introduced by the time-windowing operation reduces the ability to pick out peaks (*resolvability*) in the spectrum. As shown in Figure 7.24, if the spacing $\Delta\Omega = \Omega_2 - \Omega_1$ between two spectral lines is smaller than the bandwidth $2\pi/T_0$ of the window, the two shifted copies of $W_R(j\Omega)$ fuse into one. As a result, the sinusoidal component (spectral line) at Ω_1 is not resolved from the sinusoidal component (spectral line)
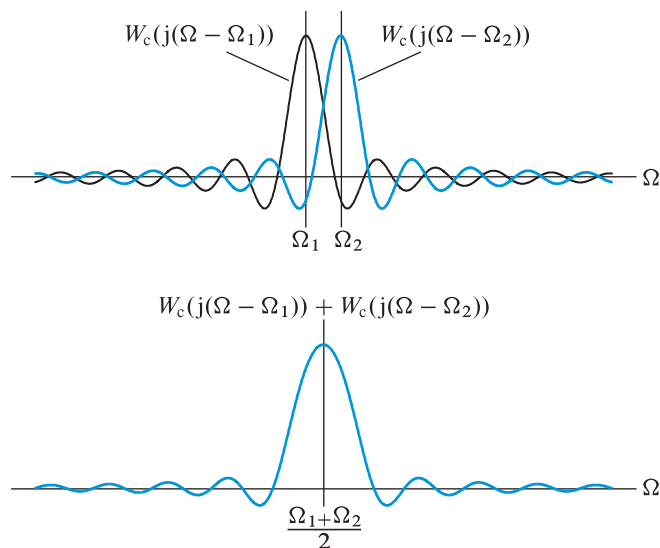
Figure 7.24 "Peak" merging (loss of spectral resolution) when two spectral lines are closer than the width of the mainlobe of the window.

at Ω_2. Therefore, to resolve two frequency components $\Delta\Omega$ rad/s apart, the length of the rectangular window should satisfy the condition

$$\Delta\Omega = \Omega_2 - \Omega_1 \geq \frac{2\pi}{T_0} \quad \text{or} \quad T_0 \geq \frac{1}{F_2 - F_1}. \tag{7.169}$$

To see another important implication of (7.169), we note that $F_1 = 0$ yields $T_0 \geq 1/F_2$. Thus, the length of the window determines the lowest frequency that may be "seen" in the spectrum of the windowed signal. In other words, the length of the rectangular window required to distinguish a sinusoid from "DC" should be larger than its fundamental period.

7.6.2 Effects of time-windowing on signals with continuous spectra

The ideas discussed in Section 7.6.1 can be easily extended to an arbitrary number of sinusoidal signals. This is illustrated in Figure 7.25 for a number of sinusoids with equally spaced frequencies. Clearly, as the number of sinusoids increases the spacing between the spectral lines decreases; in the limit, the result is an ideal continuous bandpass spectrum.

The spectrum of a windowed aperiodic signal is obtained by taking the CTFT of (7.161). The result is the following convolution integral (see Review Problem 57):

$$\hat{X}_c(j\Omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X_c(j\theta) W_c(j(\Omega - \theta))d\theta. \tag{7.170}$$

Thus, *the Fourier transform of the windowed signal is obtained by convolving the Fourier transform of the original signal with the Fourier transform of the window*. To understand
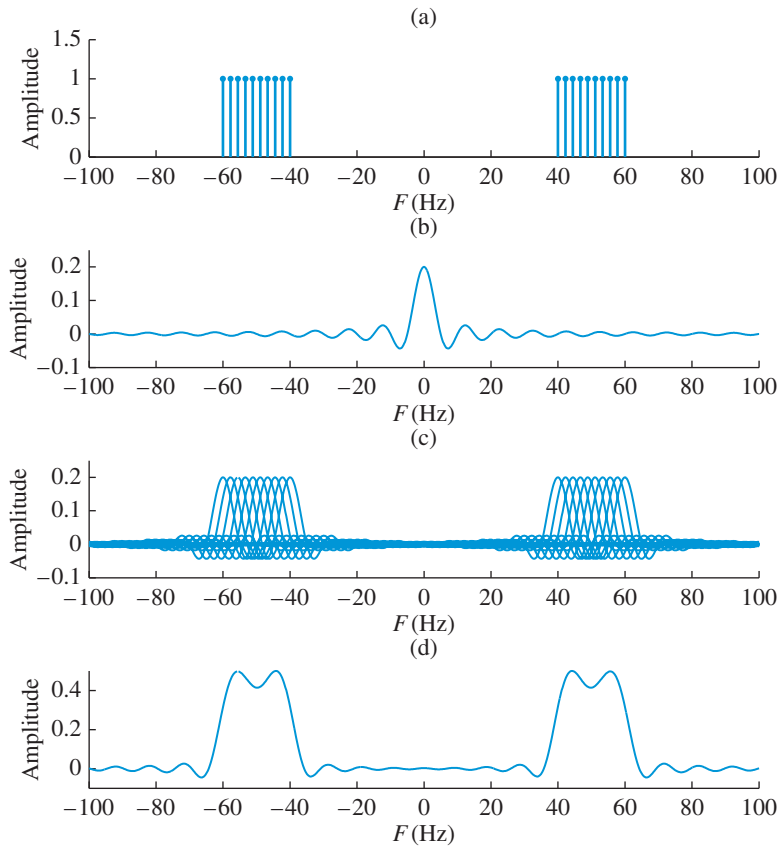
Figure 7.25 The effects of windowing on the spectrum of an ideal bandpass signal using a sum of equally spaced sinusoidal components: (a) spectrum of infinite duration signal, (b) spectrum of rectangular window, (c) shifted copies of window spectrum, and (d) spectrum of windowed signal.

how this operation changes the original spectrum, we approximate the integral (7.170) using trapezoidal integration as follows:

$$\hat{X}_c(j\Omega) \approx \frac{1}{2\pi} \sum_{k=-\infty}^{\infty} X_c(j\theta_k) W_c(j(\Omega - \theta_k)) \Delta\theta. \tag{7.171}$$

This relation shows that the effect of time-windowing is to create a "continuous" set of frequency-shifted copies of $W_c(j\Omega)$, each scaled by the corresponding amplitude of the original spectrum. The estimated CTFT, $\hat{X}_c(j\Omega)$, is the sum of all these copies; this process is illustrated in Figure 7.25.

Alternatively, $\hat{X}_c(j\Omega)$ may be expressed as a weighted integral of $X_c(j\Omega)$, where the weight function, $W_c(j\Omega)$, is the Fourier transform of the window. The convolution operation (7.170) produces a weighted average of the values of $X_c(j\theta)$ with the largest weights attached to the frequencies in the neighborhood of $\theta = \Omega$. In other words, $\hat{X}_c(j\Omega)$ corresponds to a "locally" weighted average of $X_c(j\theta)$ in the neighborhood of the frequency Ω.

Another way to understand the effects of time windowing on the original spectrum is to assume that the mainlobe acts as a smoothing (lowpass) operator and the sidelobes act as a peak-enhancement (highpass) operator.

7.6.3   "Good" windows and the uncertainty principle

In summary, time-windowing of a signal introduces two types of spectral distortion:

Smearing The predominant effect of the mainlobe is to smear or spread the original spectrum. The result is loss of resolution. An ideal spectral line in the original spectrum will have a width of about $2\pi/T_0$ after windowing. Two equal amplitude sinusoids with frequencies less than $2\pi/T_0$ apart will blend with each other and may appear as a single sinusoid.

Leakage The major effect of the sidelobes is to transfer power from frequency bands that contain large amounts of signal power into bands that contain little or no power. This transfer of power, which is called leakage, may create "false" peaks (that is, peaks at wrong frequencies), nonexisting peaks, or change the amplitude of existing peaks.

Smearing and leakage are especially critical for spectra with strong peaks and valleys while their effects on smooth and relatively flat spectra are negligible.

A "good" window should have a narrow mainlobe (to minimize spectral spreading) and "low" sidelobes (to minimize spectral leakage). Unfortunately, as we show below, it is impossible to satisfy both of these requirements simultaneously. This is a consequence of the uncertainty principle of Fourier transforms.

The rectangular window $w_R(t)$ has finite duration T_0 and a spectrum $W_R(j\Omega)$ of infinite extent in frequency. However, the majority of its energy is contained in the mainlobe of the sinc function, $|\Omega| < 2\pi/T_0$. As the duration T_0 increases, the width $4\pi/T_0$ of the mainlobe decreases, and vice versa (see Figure 4.14).

Time and frequency scaling property This result is a consequence of a more general property of the Fourier transform, known as scaling theorem, which states that if $x_c(t)$ has Fourier transform $X_c(j\Omega)$, then for any real constant a we have (see Tutorial Problem 22)

$$x_c(at) \xleftrightarrow{\text{CTFT}} \frac{1}{|a|}X_c\left(\frac{j\Omega}{a}\right). \tag{7.172}$$

The scaling property implies that if $x_c(t)$ becomes wider, its spectrum becomes narrower, and vice versa. Intuitively, compressing (expanding) the time axis by a factor $a > 1$ ($a < 1$) means that the signal is varying faster (slower) by factor a. This can be easily seen by scaling the signal $x_c(t) = \cos(\Omega_0 t)$. The result is

$$x_c(at) = \cos[\Omega_0(at)] = \cos[(a\Omega_0)t]. \tag{7.173}$$

For example, if $a > 1$, compressing the time axis by a factor a increases the frequency of the sinusoid by a factor of a.

Uncertainty principle The scaling property describes the general nature of the inverse relationship between the time and frequency "extent" of a continuous-time signal.

To obtain an analytical expression we need precise definitions of time duration and band-width (frequency extent). A convenient and widely used definition has originated from the statistical concept of variance (see Section 13.1.2). Without loss of generality, we consider a signal $x_c(t)$, which is normalized to have unit energy, that is

$$E_x = \int_{-\infty}^{\infty} |x_c(t)|^2 dt = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X_c(j\Omega)|^2 d\Omega = 1. \qquad (7.174)$$

To assure the existence of the various integrals, we impose the condition $\sqrt{|t|}x_c(t) \to 0$ as $|t| \to \infty$. Furthermore, we assume that $x_c(t)$ is centered about the origin so that the "mean value" m_t of $|x_c(t)|^2$ satisfies the condition

$$m_t \triangleq \int_{-\infty}^{\infty} t|x_c(t)|^2 dt = 0. \qquad (7.175)$$

The time duration σ_t of $x_c(t)$ is defined by the following formula:

$$\sigma_t^2 \triangleq \int_{-\infty}^{\infty} t^2 |x_c(t)|^2 dt. \qquad (7.176)$$

A signal $x_c(t)$ which is large for large values of t will have larger duration than a signal with large values for small values of t. Thus, σ_t measures the spread of the curve $|x_c(t)|^2$ about $t = 0$. Similarly, the bandwidth is defined by

$$\sigma_\Omega^2 \triangleq \frac{1}{2\pi} \int_{-\infty}^{\infty} \Omega^2 |X_c(j\Omega)|^2 d\Omega, \qquad (7.177)$$

where we again assume that the mean $m_\Omega \triangleq \int_{-\infty}^{\infty} \Omega |X_c(j\Omega)|^2 d\Omega = 0$.

The *uncertainty principle*, which was first introduced in quantum mechanics in a differ-ent interpretation and units, states that the time–bandwidth product for any signal is lower bounded according to the relationship

$$\sigma_t \sigma_\Omega \geq \frac{1}{2}. \qquad (7.178)$$

In other words, *the duration and bandwidth of any signal cannot be arbitrarily small simul-taneously*. Thus, signals of short duration must have large bandwidth and signals of narrow bandwidth must have long duration.

To prove (7.178) we shall use the famous Schwarz inequality

$$\left| \int_{-\infty}^{\infty} x_{c_1}(t) x_{c_2}(t) dt \right|^2 \leq \int_{-\infty}^{\infty} |x_{c_1}(t)|^2 dt \int_{-\infty}^{\infty} |x_{c_2}(t)|^2 dt, \qquad (7.179)$$

which, as shown in Tutorial Problem 23, is easy to derive. The equality holds when $x_{c_1}(t)$ is proportional to $x_{c_2}(t)$, that is, $x_{c_1}(t) = kx_{c_2}(t)$ for all t. If we now define the functions $x_{c_1}(t)$ and $x_{c_2}(t)$ as

$$x_{c_1}(t) = tx_c(t), \qquad (7.180)$$

$$x_{c_1}(t) = \frac{dx_c(t)}{dt}, \qquad (7.181)$$

then the left hand side of (7.179) yields

$$\int_{-\infty}^{\infty} tx_c(t)\frac{dx_c(t)}{dt}dt = t\frac{x_c^2(t)}{2}\Big|_{-\infty}^{\infty} - \frac{1}{2}\int_{-\infty}^{\infty} x_c^2(t)dt = -\frac{1}{2}. \tag{7.182}$$

Next, we recall that

$$x_c(t) \xleftarrow{\text{CTFT}} X_c(j\Omega), \tag{7.183}$$

$$\frac{dx_c(t)}{dt} \xleftarrow{\text{CTFT}} j\Omega X_c(j\Omega). \tag{7.184}$$

Hence (7.179) becomes

$$\frac{1}{4} \leq \int_{-\infty}^{\infty} t^2 |x_c(t)|^2 dt \; \frac{1}{2\pi}\int_{-\infty}^{\infty} \Omega^2 |X_c(j\Omega)|^2 d\Omega, \tag{7.185}$$

from which (7.178) follows immediately. The equality in (7.178) holds when $ktx_c(t) = x_c'(t)$ or $x_c'(t)/x_c(t) = kt$. Integrating, we have $\ln[x_c(t)] = kt^2/2 + c_1$ or

$$x_c(t) = c_2 e^{kt^2/2}. \tag{7.186}$$

For $k < 0$ this is a finite energy signal known as the *Gaussian pulse*. Gaussian pulses are the only signals that satisfy the uncertainty principle (7.178) with equality. Other definitions of time duration and bandwidth lead to inequalities with a different lower bound (see Problem 45). Analogous uncertainty principles can be derived for the other Fourier representations.

Window choices A good window should have a very narrow mainlobe and no sidelobes, that is, it should be a good approximation to a delta function. However, according to the uncertainty principle, a small bandwidth would require an extremely long window. Given a window of fixed length, we can only change its shape. Unfortunately, there are no systematic procedures to design windows with desired shapes. The rectangular window, which corresponds to signal truncation without reshaping, can be used as the starting point for design of some useful windows. To simplify the derivations we consider time-limited windows $w_c(t) = 0$ for $|t| > T_0/2$, which are centered at $t = 0$ and normalized so that

$$w_c(0) = \frac{1}{2\pi}\int_{-\infty}^{\infty} W_c(j\Omega)d\Omega = 1. \tag{7.187}$$

A simple way to reduce the level of sidelobes of the rectangular window by a factor of 2 (in dB) is based on squaring $W_R(j\Omega)$. This time-domain window is obtained by the convolution $w_R(t) * w_R(t)$, which yields a triangular pulse with duration $2T_0$. Adjusting the duration to T_0 and normalizing according to (7.187) yields

$$w_B(t) = \left(1 - \frac{2|t|}{T_0}\right)w_R(t) \xleftarrow{\text{CTFT}} W_B(j\Omega) = \frac{4\sin^2(\Omega T_0/4)}{\Omega^2 T_0/2}. \tag{7.188}$$

This window is known as a *triangular window* due to its shape or as a *Bartlett window* after its discoverer.

Another way to reduce the level of the sidelobes is to shift two properly weighted replicas of $W_R(j\Omega)$ at $\pm 2\pi/T_0$ rads/s to achieve partial cancellation by superposition. The Fourier transform of the new window is

$$W_c(j\Omega) = aW_R(j\Omega) + bW_R(j(\Omega - 2\pi/T_0)) + bW_R(j(\Omega + 2\pi/T_0)). \qquad (7.189)$$

The choice $a = 0.5$, $b = 0.25$, made by the Austrian scientist Julius von Hann, yields the *Hann window*; the choice $a = 0.54$, $b = 0.23$, made by Richard Hamming using trial and error to minimize the level of the highest sidelobe, results in the *Hamming window*. Taking the inverse Fourier transform of (7.189), we obtain (see Tutorial Problem 24)

$$w_{\text{Han}}(t) = \left[0.50 + 0.50\cos\left(\frac{2\pi t}{T_0}\right)\right] w_R(t), \qquad (7.190)$$

$$w_{\text{Ham}}(t) = \left[0.54 + 0.46\cos\left(\frac{2\pi t}{T_0}\right)\right] w_R(t). \qquad (7.191)$$

Figure 7.26 shows the four windows discussed and their spectra in a combination of linear and logarithmic scales. We first note that the increased concentration of the three nonrectangular windows increases the width of their mainlobe by a factor of two compared to the
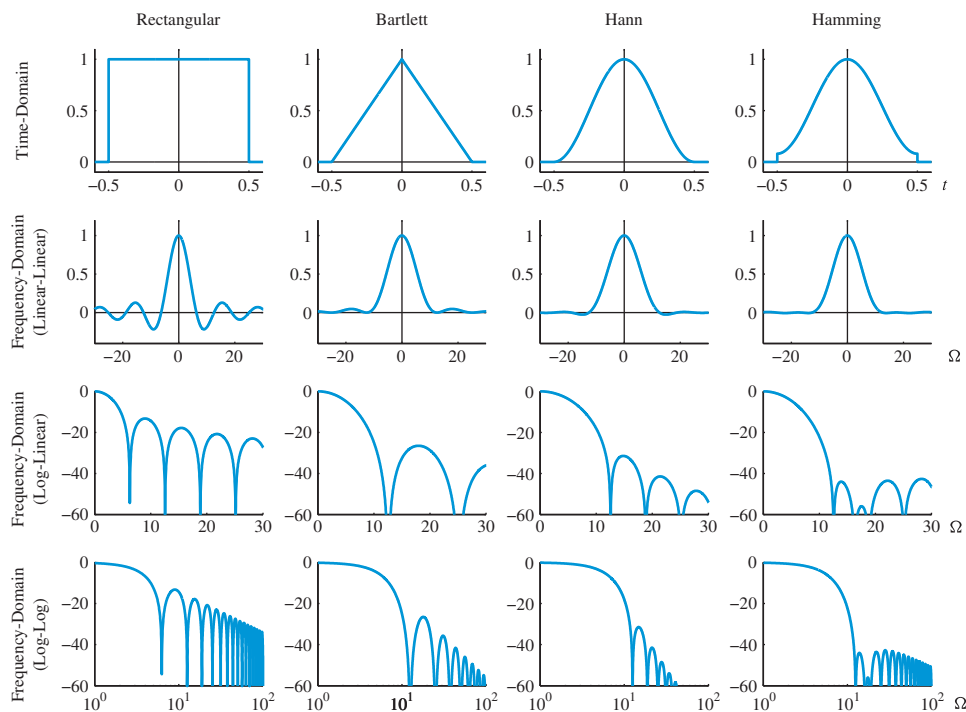
Figure 7.26 Time-domain and frequency-domain characteristics of continuous-time rectangular, Bartlett (triangular), Hann, and Hamming windows.

Table 7.5 Some continuous-time windows and their characteristics

Window	Mainlobe width	Rolloff rate (dB/octave)	Peak sidelobe level (dB)
Rectangular	$4\pi/T_0$	-6	-13.3
Bartlett	$8\pi/T_0$	-12	-26.5
Hann	$8\pi/T_0$	-18	-31.5
Hamming	$8\pi/T_0$	-6	-42.9

rectangular window. This is a consequence of the uncertainty principle. Thus, if we wish for the best spectral resolution we should choose the rectangular window. However, this gain in spectral resolution comes at the expense of spectral leakage. To reduce leakage, we need a window with lower level sidelobes with a faster rate of decay. The rectangular window has a jump discontinuity and its spectrum decays asymptotically as $1/\Omega$. The triangular window, which has a discontinuous first derivative, has a spectrum decaying as $1/\Omega^2$. In general, the smoothness of a signal is measured by the number of continuous derivatives it possesses. *The smoother the signal, the faster the decay of its spectrum.* Thus, we can improve leakage behavior by choosing a smooth (tapered) window. For a given duration, smoothing the window by tapering to reduce the level of sidelobes decreases the effective time-duration, and therefore increases the width of the mainlobe. Thus, *we cannot simultaneously increase spectral resolution and decrease leakage.*

The key characteristics of the four windows are summarized in Table 7.5. The rectangular window has the smallest mainlobe width ($4\pi/T_0$) and the highest sidelobe level. Among the three windows with mainlobe width $8\pi/T_0$, the Hamming window has the lowest sidelobe level, and the Hann window has the fastest sidelobe decay. Although many windows with optimum properties have been proposed in the literature, *the Hann window is sufficient for spectral analysis of real-world signals* (see Section 14.2 on PSD estimation). A more detailed treatment of windows, in the context of filter design, is provided in Section 10.3.

Example 7.7

In this example, we illustrate the effects of window shape on the spectrum of windowed sinusoids. Consider a continuous-time signal consisting of a sum of $(K + 1)$ sinusoidal components:

$$x_c(t) = \sum_{k=0}^{K} A_k \cos(\Omega_k t + \phi_k). \quad -\infty < t < \infty \quad (7.192)$$

The CTFT of the windowed signal $\hat{x}_c(t) = w_c(t)x_c(t)$ is given by [see (7.166)]

$$\hat{X}_c(j\Omega) = \frac{1}{2} \sum_{k=0}^{K} A_k e^{-j\phi_k} W_c(j(\Omega + \Omega_k)) + A_k e^{j\phi_k} W_c(j(\Omega - \Omega_k)), \quad (7.193)$$
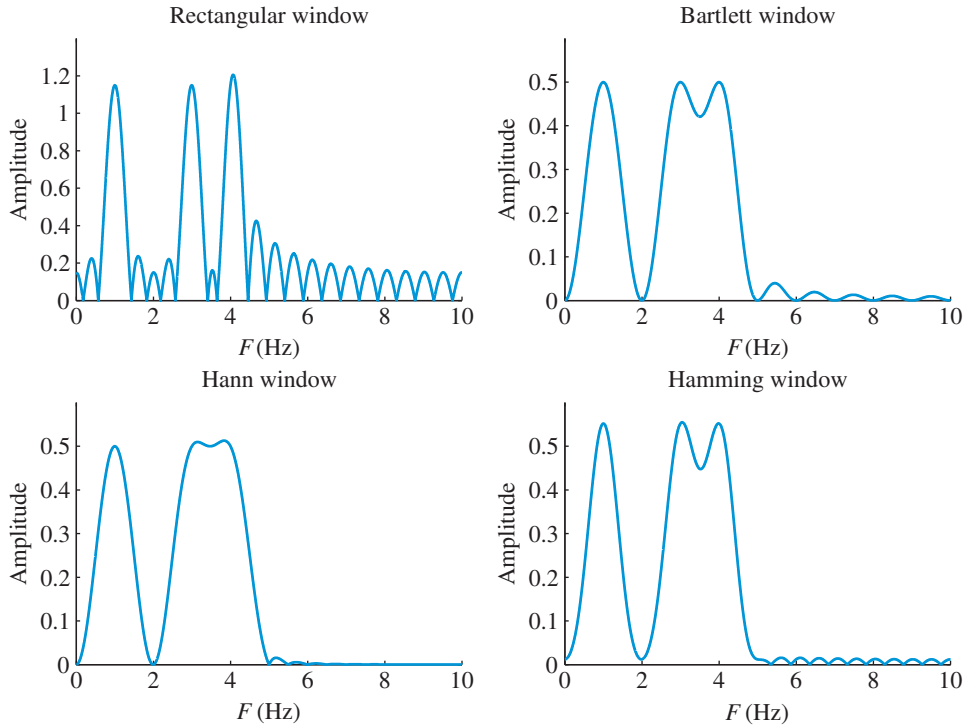
Figure 7.27 Effect of window shape on the spectrum of sinusoidal signals.

where $W_c(j\Omega)$ is the Fourier transform of the window. We choose three sinusoids with amplitude one, phase zero, and frequencies $F_0 = 1$ Hz, $F_1 = 3$ Hz, and $F_2 = 4$ Hz. The length $T_0 = 2$ s of the windows was chosen to satisfy the condition $T_0 > 1/(F_2 - F_1)$. Figure 7.27 shows that it is possible to distinguish the peaks at F_1 and F_2 with a rectangular window of length T_0, but not with the other windows. As expected, the Hann window has the smallest sidelobes at the expense of a slightly lower resolution. To distinguish two peaks at frequencies F_1 and F_2 with nonrectangular windows, their length should satisfy the condition $T_0 > 2/(F_2 - F_1)$. This is not surprising because the mainlobe of nonrectangular windows has double width compared to that of the rectangular window. ∎

7.6.4   Effects of frequency-domain sampling

Figure 7.28 shows the basic steps required to determine the frequency content of a continuous-time signal using the DFT. The first step is that of windowing, which as we have discussed has the biggest impact on the quality of the calculated spectrum. To avoid unnecessary complications from the periodicity of the DTFT, we have explained the effects of windowing in continuous-time using the CTFT. However, in practice the windowing operation takes place in discrete-time. Therefore, ultimately, what is available for digital computation is the windowed sequence
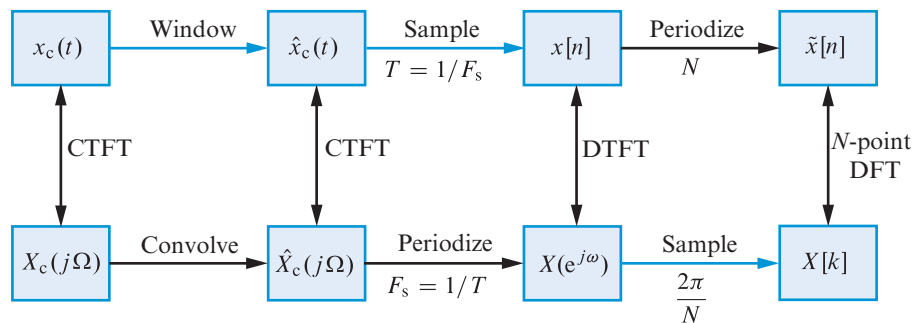
Figure 7.28 Steps (shown by blue arrows) for spectral analysis of a continuous-time signal using the DFT and their implications. Time windowing leads to spectral smearing and leakage. Sampling in one domain causes periodization (and maybe aliasing) in the other domain (see Figure 7.8 for illustrations).

$$\hat{x}[n] = w_c(nT)x_c(nT) = w[n]x[n]. \quad 0 \leq n \leq L - 1 \tag{7.194}$$

The DTFT of $\hat{x}[n]$ is given by the periodic convolution formula, that is

$$\hat{X}(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta})W(e^{j(\omega-\theta)})d\theta. \tag{7.195}$$

Our goal is to determine $X_c(j\Omega)$, the CTFT of $x_c(t)$; what we are able to compute is the DTFT $\hat{X}(e^{j\omega})$ of the sampled and windowed signal $\hat{x}[n]$.

Sampling $x_c(t)$ with period T is equivalent to periodization of $X_c(j\Omega)$ with period $2\pi/T$. Sampling, which may create aliasing distortion, limits the useful frequency range to $|\Omega| \leq \pi/T$. Discrete-time windowing smooths sharp peaks and discontinuities (mainlobe effects) and may create false peaks (sidelobe effects), like continuous-time windowing. The shape of $W(e^{j\omega})$, for $|\Omega| \leq \pi/T$, is very similar to the shape of $W_c(j\Omega)$ for typical values of L. For example, the DTFT of the discrete-time rectangular window is the Dirichlet function

$$W_R(e^{j\omega}) = \frac{\sin(\omega L/2)}{\sin(\omega/2)}e^{-j\omega(L-1)/2}. \tag{7.196}$$

For large L the Dirichlet function (7.196) and the sinc function (7.168) are essentially identical in the region around the mainlobe (see Tutorial Problem 25). The periodic convolution operation (7.195), which is used in filter design, is further discussed in Section 10.3.

The final step is to compute samples of the DTFT at frequencies $\omega_k = (2\pi k)/N$. This is equivalent to periodizing the sequence $\hat{x}[n]$ with period N. The actual computation is performed by computing the N-point DFT ($N \geq L$) of the windowed signal $\hat{x}[n]$, $0 \leq n \leq L - 1$ using a FFT algorithm. The correspondence between the index of DFT coefficients and continuous-time frequencies is

$$\Omega_k = 2\pi F_k = \frac{2\pi k}{NT}. \quad 0 \leq n \leq N - 1 \tag{7.197}$$

The quantity $2\pi/(NT)$ determines the spacing between the samples of DTFT evaluated by the DFT; we typically use zero-padding to obtain a faithful visual representation of the DTFT. The physical spectral resolution, which was explained in Section 7.6.3, depends upon the length L of the window. We stress that *the window should be applied to the signal segment (actual data) before zero-padding*. Thus, we should chose the length of the window, L, by avoiding any leading or trailing zeros in the data.

All fast implementations of DFT compute the DTFT at these frequencies. However, sometimes we need samples of the DTFT in the range $-\pi < \omega \le \pi$, instead of the range $0 \le \omega < 2\pi$. The conversion between the two ranges, which is based on the periodicity of the DTFT, is done using the MATLAB function

$$X = \texttt{fftshift(X)}. \tag{7.198}$$

This function rearranges the output of `fft` by moving the zero-frequency component to the center of the array. It is useful for visualizing a Fourier transform with the zero-frequency component in the middle of the spectrum. The effects of `fftshift` are undone using the function

$$X = \texttt{ifftshift(X)}. \tag{7.199}$$

We can avoid most pitfalls related to the application of DFT in spectral analysis, if we keep in mind the following observations:

1. The sampling frequency, $F_s = 1/T$, which is chosen to avoid aliasing distortion, determines the upper limit of the useful frequency range $0 \le F \le F_s/2$.
2. The length and shape of the window, $T_0 = LT$, determines the spectral resolution. To resolve two sinusoidal components at frequencies F_1 and F_2, we should chose $T_0 > 1/(F_2 - F_1)$, for a rectangular window, and $T_0 > 2/(F_2 - F_1)$, for a nonrectangular window.
3. The Hann window provides a good trade-off between spectral resolution and leakage for most practical applications. The discrete-time Hann window is

$$w[n] = \begin{cases} 0.5 - 0.5\cos(2\pi n/N), & 0 \le n \le N-1 \\ 0. & \text{otherwise} \end{cases} \tag{7.200}$$

The Hann window is implemented by the MATLAB function `w=hann(N)` and its time and frequency domain characteristics can be examined using the MATLAB window visualization tool function `wvtool(winname(N))`, which is shown in Figure 7.29.
4. The length N of the DFT should be much larger than $L = T_0/T$ to obtain a good visual representation of the DTFT. If we set N to a power of two, that is, $N = 2^Q$, the MATLAB `fft` function runs faster.

We next illustrate the effects of windowing and DFT spectral sampling by means of some examples. Without loss of generality we consider discrete-time signals.
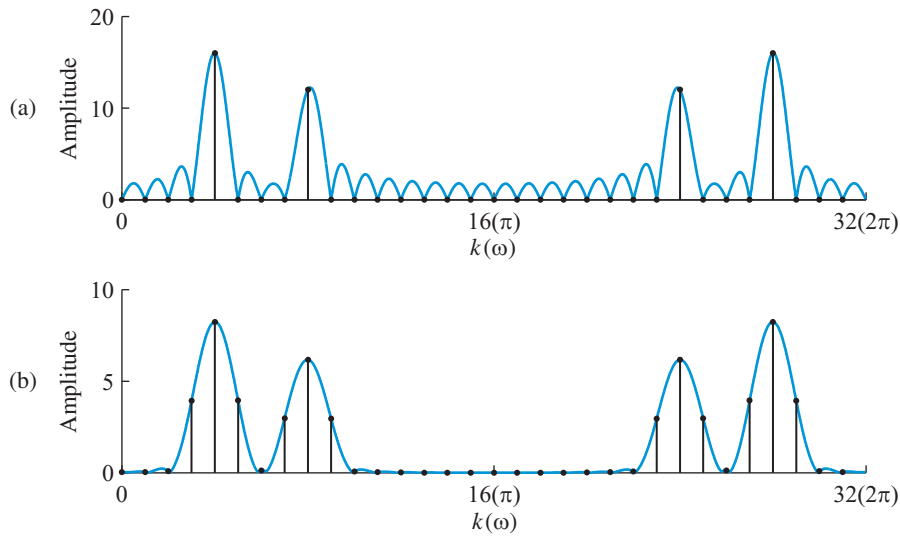
Figure 7.30 Spectrum analysis of the sum of two sinusoids whose frequencies coincide with the sampling points (bins) of the DFT. The plots show the magnitude of the DTFT as a solid line and the magnitude of the DTF as a stem plot when the signal is windowed with (a) a rectangular window, and (b) a Hann window.

Example 7.9　Sinusoids with frequencies *not* coinciding with DFT "bins"

Consider the sequence

$$x[n] = \begin{cases} \cos\left(\frac{2\pi}{9}n\right) + \frac{3}{4}\cos\left(\frac{4\pi}{7}n\right), & 0 \le n \le 31 \\ 0. & \text{otherwise} \end{cases} \tag{7.202}$$

In this case, the frequencies $\omega_1 = 2\pi/9$ and $\omega_2 = 4\pi/7$ fall between the bins ("cracks") of the DFT. Indeed, we can easily see that $3/32 < 3/27 < 4/32$ and $9/32 < 8/28 < 10/32$. As in Example 7.8, the spectrum of the windowed signals is given by the DTFT, which is shown in Figure 7.31 with a continuous blue line. Because of the mismatch between the zero crossings of the rectangular window and the DFT bins, all samples of the DFT are nonzero. In this sense, there is a significant visual difference between the DFTs for the rectangular windows in Figures 7.30 and 7.31. In contrast, the DFTs for the Hann windowed signals are remarkably similar. Thus, to avoid misleading conclusions we should always use a Hann window and a DFT with large zero-padding. ∎

In summary, it is important to keep in mind that in practical spectral analysis we deal with finite duration discrete-time signals, whose spectrum is given by the DTFT. The N-point DFT is merely used to evaluate samples of the DTFT at N equally spaced frequencies $\omega = 2\pi k/N$, $0 \le n \le N - 1$. For meaningful results we should use a good window (like the Hann window) and oversample the DTFT.
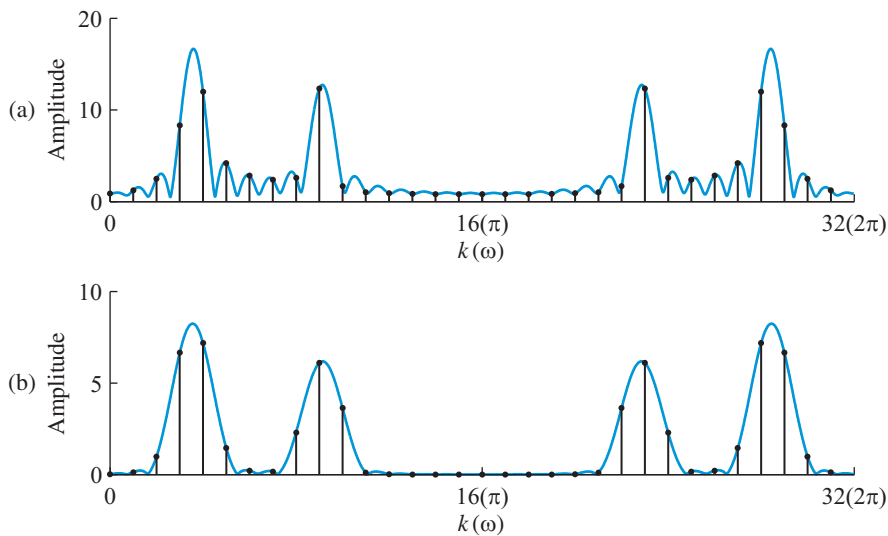
Figure 7.31 Spectrum analysis of the sum of two sinusoids whose frequencies do *not* coincide with the sampling points (bins) of the DFT. The plots show the magnitude of the DTFT as a solid line and the magnitude of the DTF as a stem plot when the signal is windowed with a rectangular window (a), and a Hann window (b).

7.6.5   The spectrogram

The validity of spectrum analysis using the DFT is based on an implicit fundamental assumption: the amplitudes, frequencies, and phases of the sinusoidal components of the analyzed signal do *not* change with time within the analysis window. Since increasing the length of the window results in better frequency resolution, we would like to use long windows. However, there are two major problems prohibiting the use of very long windows in practical applications. First, waiting to collect all necessary samples introduces long delays and requires the computation of huge DFTs. Second, the frequency content of speech, radar, sonar, and other practical signals changes with time. Thus, the length of the window should be sufficiently short to assure that the spectral content does not vary significantly within the window for practical purposes. If the spectral content changes significantly inside the analysis window, the DFT will provide erroneous frequency analysis results.

A reasonable practical solution to these problems is to break a long signal into small segments and analyze each one with the DFT. To formalize this approach we define the *time-dependent DFT* or *short-time DFT* of a signal $x[n]$ by

$$X[k, n] \triangleq \sum_{m=0}^{L-1} w[m] x[n + m] e^{-j(2\pi k/N)m}, \qquad (7.203)$$

where L is the length of the window $w[n]$ and $k = 0, 1, \ldots, N - 1$. This equation has a simple interpretation: the set of numbers $X[k, n]$, $0 \leq k \leq N - 1$ is the N-point DFT of a windowed segment of $x[m]$ starting at $m = n$ and ending at $m = n + L - 1$. The

window is fixed in the interval from $m = 0$ to $m = L - 1$. As the shift index n changes, the signal $x[n + m]$ slides and the window extracts a different segment of the signal for analysis. Essentially, for each value of n we extract a windowed segment of the signal $x[m]$ and we evaluate the "local" spectrum. This process is also known as *short-time Fourier analysis*. The two-dimensional sequence $X[k, n]$, which represents the contribution of frequency component $\omega_k = 2\pi k/N$ at the segment specified by the time index n, is called a *spectrogram*. We usually plot $|X[k, n]|$ or $\log |X[k, n]|$ as a grayscale or pseudocolor image where the horizontal axis represents time and the vertical axis frequency. The logarithmic scale helps us to see small amplitude components. To illustrate these ideas, consider the following example.

Example 7.10   Spectrogram of linear FM (chirp) signal

Consider the continuous-time linear FM (chirp) signal defined in Example 5.2 by

$$x_c(t) = \sin[\pi (F_1/\tau)t^2]. \quad 0 \le t \le \tau \tag{7.204}$$

The instantaneous frequency, that is, the time derivative of the angle is given by

$$F_i(t) = \frac{1}{2\pi} \frac{d}{dt} \left(\pi t^2 F_1/\tau \right) = F_1 \frac{t}{\tau}. \quad 0 \le t \le \tau \tag{7.205}$$

Thus, the frequency of $x_c(t)$ increases linearly from $F = 0$ to $F = F_1$. If we sample $x_c(t)$ with $F_s = 1/T$ and we choose $\tau = LT$, we obtain the sequence

$$x[n] \triangleq x_c(nT) = \sin(\pi f_1 n^2/L), \tag{7.206}$$

where $f_1 \triangleq F_1/F_s$. The instantaneous normalized frequency is given by

$$f_i(n) = f_1 \frac{n}{L}. \quad 0 \le n \le L - 1 \tag{7.207}$$

Choosing $F_1 = 500$ Hz, $\tau = 10$ s, and $F_s = 1000$ Hz, we generate $L = \tau F_s = 10\,000$ samples of the sequence $x[n]$. Figure 7.32(a) shows the first 2000 samples of $x[n]$ plotted as a continuous waveform (successive samples are connected by straight lines). We note that as time progresses, the peaks get closer, which indicates that the frequency increases with time. We next compute the DFT of this segment and we plot its magnitude after shifting the zero frequency to the middle with function `fftshift` (7.198). The result, which is shown in Figure 7.32(b), suggests that the signal contains all frequency components from about 0 Hz to 100 Hz in approximately the same amount. This conclusion, which follows from the fundamental assumption of Fourier analysis that the amplitude, frequency, and phase of each frequency component remain constant over time, is obviously incorrect. Indeed, from (7.204) we know that the signal is made from a single sinusoid whose frequency changes linearly with time.

If we look at any short segment of $x[n]$, it resembles a sinusoid with an almost constant frequency. Therefore, if we window a sufficiently short segment and compute its DFT, we could get a local representative spectrum of the signal. This idea is illustrated in Figure 7.33. Since the window is centered around $t = 0.8$ s, the instantaneous frequency is
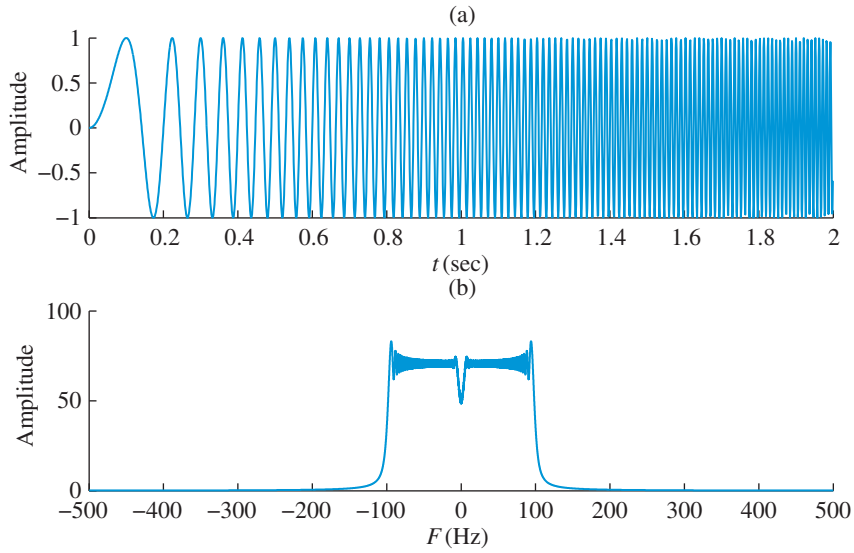
Figure 7.32 (a) Linear FM (chirp) signal, and (b) the magnitude of its DTFT.
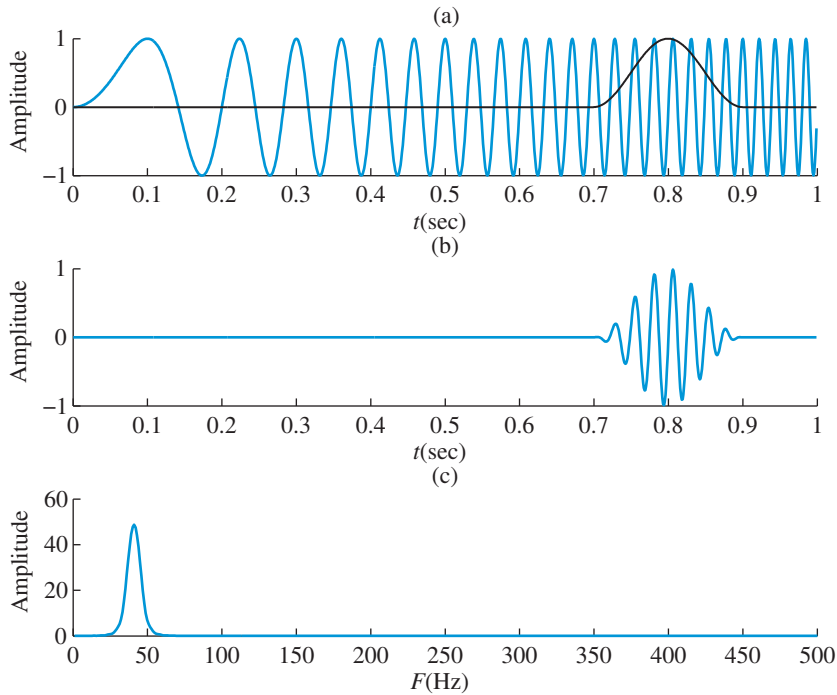
Figure 7.33 A time-dependent DFT: (a) linear FM signal and a time-shifted copy of the Hann window, (b) windowed signal segment, and (c) magnitude of DFT of the windowed signal segment shown in (b).

about $F_i(t) = 40$ Hz, and the spectrum is a narrow peak centered at $F = 40$ Hz. To obtain these results we have used a Hann window with length $L = 200$ and a DFT with $N = 256$ points.

```
function S=spectrogram0(x,L,NFFT,step,Fs)
% S=spectrogram0(x,L,NFFT,step,Fs)
N=length(x); K=fix((N-L+step)/step);
w=hanning(L); time=(1:L)';
N2=NFFT/2+1; S=zeros(K,N2);
for k=1:K
    xw=x(time).*w;
    X=fft(xw,NFFT);
    X1=X(1:N2)';
    S(k,1:N2)=X1.*conj(X1);
    time=time+step;
end
S=fliplr(S)'; S=S/max(max(S));
colormap(1-gray); % colormap(jet);
tk=(0:K-1)'*step/Fs; F=(0:NFFT/2)'*Fs/NFFT;
imagesc(tk,flipud(F),20*log10(S),[-100 0]); axis xy
```

Figure 7.34 MATLAB function for calculation and display of spectrograms.

The time-varying frequency content of the chirp signal is captured by the spectrograms shown in Figure 7.35. The spectrograms are obtained by the MATLAB function spectrogram0, which is shown in Figure 7.34. This function computes the N-point DFT of successive segments using a Hann window of length $L \leq N$, which is shifted every $M \leq L$ samples. The scales of time axis (s) and frequency axis (Hz) are determined by the sampling frequency F_s. The call statement is

$$S=\text{spectrogram0}(x,L,N,M,Fs). \tag{7.208}$$

MATLAB includes a more general function spectrogram in the Signal Processing toolbox; however, spectrogram0 illustrates more clearly how to compute and display the spectrogram. Figure 7.35(a) shows the spectrogram of a chirp signal with duration $\tau = 10$ s, $F_1 = 500$ Hz, and $F_s = 1000$ Hz, obtained using a Hann window with $L = 50$ and a DFT with $N = 256$ points. As expected, the instantaneous frequency grows linearly from $F = 0$ Hz to $F = F_s/2 = 500$ Hz. The value of $20 \log |X[k, n]|$ over a restricted range of 100 dB is represented by the grayness of the pixel at $[k, n]$. For displaying on a computer screen it is preferable to use a pseudocolor representation. The thickness of the line is related to the width of the mainlobe, which is approximately $4F_s/L$ Hz; the time resolution is proportional to the length of the window. Therefore, we cannot simultaneously improve time and frequency resolution. Figure 7.35(b) shows a spectrogram with improved frequency resolution obtained by increasing the length of the window to $L = 200$. If we set
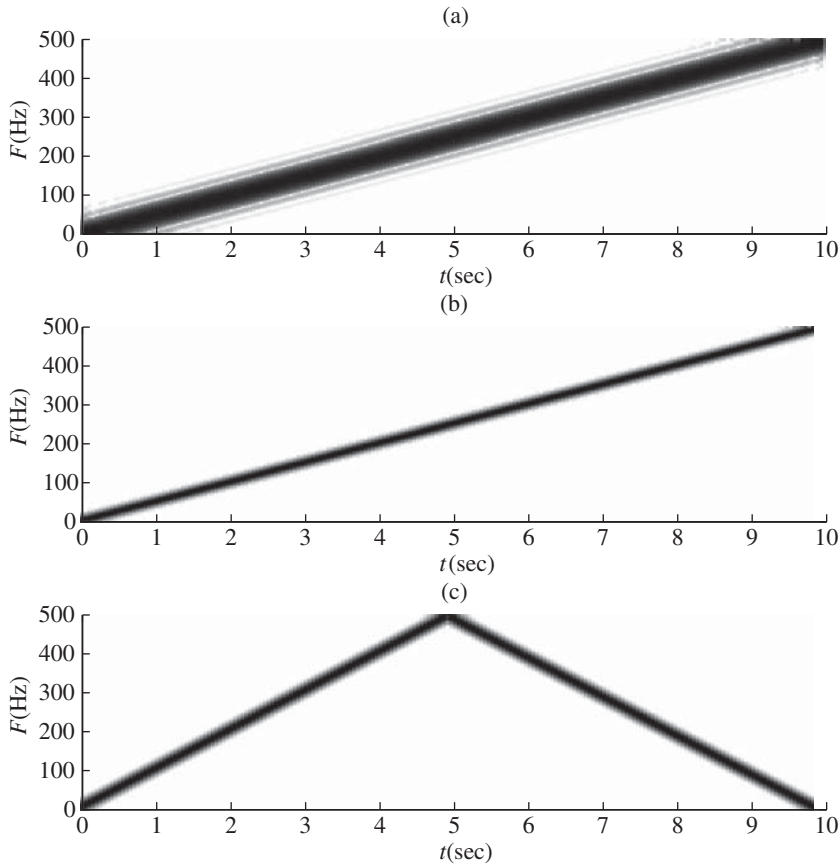
Figure 7.35 (a) Spectrogram of a linear FM signal with a Hann window of length $L = 50$, maximum frequency $F_1 = 500$ Hz and sampling frequency $F_s = 1000$ Hz. (b) Increasing the length to $L = 200$ improves the frequency resolution of the spectrogram. (c) Increasing the highest frequency to $F_1 = 1000$ Hz is not "seen" in the spectrogram because the upper frequency limit is $F_s/2 = 500$ Hz.

$F_1 = F_s = 1000$ Hz, we obtain the spectrogram in Figure 7.35(c). Since the highest frequency we can "see" in a discrete-time signal is $F_s/2$, we notice that the frequency grows linearly from 0 to 500 Hz and then decays linearly to 0 due to aliasing (see the discussion regarding Figure 6.15). ■

This example illustrates how to apply the principles of spectrum analysis to signals whose properties change with time. The spectrogram is the most practical tool for analysis of signals with time-varying spectra and is extensively used in speech processing applications. More details can be found in the references.

Learning summary

- The Discrete Fourier Transform (DFT) is a finite orthogonal transform which provides a unique representation of N consecutive samples $x[n]$, $0 \leq n \leq N - 1$ of a sequence through a set of N DFT coefficients $X[k]$, $0 \leq k \leq N - 1$

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \xleftarrow[\;N\;]{\text{DFT}} x[n] = \frac{1}{N} \sum_{k=0}^{N-1} x[n]e^{j2\pi kn/N}.$$

  The DFT does not provide any information about the unavailable samples of the sequence, that is, the samples *not* used in the computation. The interpretation or physical meaning of the DFT coefficients depends upon the assumptions we make about the unavailable samples of the sequence.

- If we create a periodic sequence $\tilde{x}[n]$ by repeating the N consecutive samples, the DFT coefficients and the DTFS coefficients $\tilde{c}_k$ of $\tilde{x}[n]$ are related by $X[k] = N\tilde{c}_k$. Thus, the DFT "treats" the finite segment as one period of a periodic sequence. If $x[n] = x[n+N_0]$ and $N \neq N_0$, the period "seen" by the DFT differs from the actual period of the analyzed sequence.

- If $X(e^{j\Omega})$ is the DTFT of the entire sequence and $X_N(e^{j\Omega})$ the DTFT of the finite segment $x[n]$, $0 \leq n \leq N - 1$, we have:
  - The DFT provides samples of $X_N(e^{j\Omega})$ at equally spaced points on the unit circle, that is, $X[k] = X_N(^{j2\pi k/N})$. If $x[n]$ has length $L \leq N$, then $X(e^{j\Omega}) = X_N(e^{j\Omega})$.
  - If $\tilde{X}[k] \triangleq X(^{j2\pi k/N})$, $0 \leq k \leq N-1$, the inverse DFT yields an aliased version of $x[n]$, that is, $\tilde{x}[n] = \sum_\ell x[n - \ell N]$. If $x[n]$ has length $L \leq N$, we have $X(e^{j\Omega}) = X_N(e^{j\Omega})$ and $x[n] = \tilde{x}[n]$.

- The multiplication of two N-point DFTs is equivalent to the circular convolution of the corresponding N-point sequences. Since circular convolution is related to linear convolution, we can use the DFT to compute the output of an FIR filter to an indefinitely long input sequence.

- The DFT is widely used in practical applications to determine the frequency content of continuous-time signals (spectral analysis). The basic steps are: (a) sampling the continuous-time signal, (b) multiplication with a finite-length window (Hann or Hamming) to reduce leakage, (c) computing the DFT of the windowed segment, with zero-padding, to obtain an oversampled estimate of the spectrum. The frequency resolution, which is about $8\pi/L$ rads, is determined by the length L of the window.

- The value of the DFT stems from its relation to the DTFT, its relation to convolution and correlation operations, and the existence of very efficient algorithms for its computation. These algorithms are collectively known as Fast Fourier Transform (FFT) algorithms. The FFT is not a new transform; it is simply an efficient algorithm for computing the DFT.

TERMS AND CONCEPTS

CTFS Expresses a continuous-time periodic signal $\tilde{x}_c(t)$ as a sum of scaled complex exponentials (or sinusoids) at harmonics kF_0 of the fundamental frequency F_0 of the signal. The scaling factors are called Fourier series coefficients c_k.

CTFT Expresses a continuous-time aperiodic signal $x(t)$ as an integral of scaled complex exponentials (or sinusoids) of all frequencies. The scaling factor is denoted by $X_c(j\Omega)$.

Circular addressing A "wrap around" operation on integers after they reach integer multiples of N. It produces integers between 0 and $N - 1$. Also called modulo-N operation. Denoted by $\langle n \rangle_N$.

Circular buffer Memory storage in which data are stored in a circular fashion and accessed by modulo-N addressing.

Circular convolution A convolution between two N-point sequences using circular shift resulting in another N-point sequence; it is denoted and given by

$$x_1[n] \,\textcircled{N}\, x_2[n] = \sum_{k=0}^{N-1} x_1[k] x_2[\langle n - k \rangle_N].$$

Circular folding A time- or frequency-reversal operation implemented according to the modulo-N circular addressing on a finite-length sequence. The sample at 0 remains at its position while the remaining samples are arranged in reverse order. Denoted by $\langle -n \rangle_N$.

Circular shift A shifting operation on a finite-length sequence implemented using the modulo-N circular addressing. Denoted by $\langle n - m \rangle_N$.

Circular-even symmetry A kind of even symmetry created when an even sequence is wrapped around a circle and then recovered by unwrapping and laying the axis flat.

Circular-odd symmetry A kind of odd symmetry created when an odd sequence is wrapped around a circle and then recovered by unwrapping and laying the axis flat.

DFS The periodic extension of the DFT $X[k]$ for all k and denoted by $\tilde{X}$.

DFT matrix An $N \times N$ matrix formed using Nth roots of unity and denoted by W_N.

DFT A transform-like operation on a finite-length N-point sequence $x[n]$ resulting in a finite-length N-point sequence $X[k]$ given by

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N}.$$

DTFS Expresses a discrete-time periodic signal $\tilde{x}[n]$ as a finite sum of scaled complex exponentials (or sinusoids) at harmonics k/N of the fundamental frequency $1/N$ of the signal. The scaling factors are called Fourier series coefficients $\tilde{c}_k$, and they themselves form a periodic sequence.

DTFT Expresses a discrete-time aperiodic signal $x(t)$ as an integral of scaled complex exponentials (or sinusoids) of all frequencies. The scaling factor is denoted by $\tilde{X}(\omega)$.

Data window A finite-length function (or sequence) used to truncate an infinite-length signal (or sequence) into a finite-length one by way of multiplication.

Gaussian pulse A Gaussian shaped signal that satisfies the uncertainty principle with equality, that is, the time-bandwidth product is the smallest for the Gaussian pulse.

IDFS An inverse of the DFS and a periodic extension of the IDFT $x[n]$ denoted by $\tilde{x}[n]$.

IDFT An inverse of DFT resulting in a finite-length N-point given by

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j2\pi nk/N}.$$

Inherent periodicity An intrinsic periodicity imposed by the DFT or IDFT operations on a finite-length sequence over the entire axis.

Modulo-N operation A "wrap around" operation on integers after they reach integer multiples of N. It produces integers between 0 and $N - 1$. Also called circular addressing operation. Denoted by $\langle n \rangle_N$.

Overlap–add method A block convolution approach for convolving a very long input

sequence with a finite-length M-point impulse response. The successive output blocks are overlapped by $(M - 1)$ samples and the contributions from overlapping blocks are added to assemble the correct result.

Overlap–save method A block convolution approach for convolving a very long input sequence with a finite-length M-point impulse response. The successive input blocks are overlapped by $(M - 1)$ samples and the last $(M - 1)$ samples saved from the previous output block are used to assemble the correct result.

Periodic extension or Periodization Creation of a periodic sequence from an N-point sequence as "seen" by the DFT. It is the result of the IDFT operation.

Resolvability Ability of the time-windowing operation to separate two closely spaced sharp peaks in the spectra. It is related to the length of the time-window.

Short-time DFT A DFT computed over short but overlapping data segments to illustrate time-dependent distribution of spectral power.

Spectral leakage A time-windowing effect that transfers (or leaks) power from one band to another caused by the nonzero sidelobes of the window spectra.

Spectral spreading or smearing A blurring, introduced in the spectral shape by the time-windowing operation, which affects the ability to resolve sharp peaks in the spectrum. This is due to the finite nonzero width of the mainlobe of the window spectra.

Spectrogram A time-frequency plot used to illustrate time-varying spectral distribution of power. It is computed using short-time DFT.

Time-domain aliasing When the DTFT is sampled at N equally-spaced frequencies followed by an N-point IDFT, the resulting periodic sequence is obtained by adding overlapping shifted replicas of the original sequence creating aliasing.

Uncertainty principle A principle similar to one in quantum mechanics. It states that the duration, σ_t, and bandwidth, σ_Ω, of any signal cannot be arbitrarily small simultaneously, that is, $\sigma_t \sigma_\Omega \geq 1/2$.

Zero-padding Appending zeros to a sequence prior to taking its DFT which results in a densely sampled DTFT spectrum and is a practical approach for DTFT reconstruction.

MATLAB functions and scripts

Name	Description	Page
bartlett	Computes the N-point Bartlett window coefficients	565
circfold	Circular time reversal of a sequence using modulo N	377
circonv	Circular convolution in the time-domain	386
circonvfft	Circular convolution using the FFT algorithm	393
cirshift0	Circular shift of a sequence using modulo N	384
fft	Fast algorithm for the computation of the 1D-DFT	458
fft2	Fast algorithm for the computation of the 2D-DFT	429
fftshift	Moves the zero-frequency component to the center	410
hann	Computes the N-point Hann window coefficients	410
hamming	Computes the N-point Hamming window coefficients	565
ifft	Fast algorithm for the computation of the inverse 1D-DFT	458
ifft2	Fast algorithm for the computation of the inverse 2D-DFT	429
ifftshift	Moves time-origin component to the center	410
mod	Performs the modulo-N operation on signal arguments	375
overlap_add	Overlap-and-add method of block convolution	395
overlap_save	Overlap-and-save method of block convolution	397
rectwin	Computes the N-point rectangular window coefficients	565
spectrogram0	Calculation and display of spectrograms	416

FURTHER READING

1. A detailed treatment of the DFT, at the same level as in this book, is given in Oppenheim and Schafer (2010), Proakis and Manolakis (2007), and Mitra (2006). One of the first comprehensive discussions of the DFT and its properties can be found in Cooley *et al.* (1969).

2. The two-dimensional DFT and its applications in image processing are discussed in Gonzalez and Woods (2008) and Pratt (2007). Applications of the three-dimensional DFT to video processing can be found in Woods (2006).

Review questions

1. Out of the four analysis techniques discussed in Chapter 4, which techniques can be numerically computed exactly and why?
2. Describe an approach to approximately computing CTFT using the DFT.
3. Describe an approach to approximately computing CTFS using the DFT.
4. Describe an approach to approximately computing DTFT using the DFT.
5. In numerical computation, which operation has the biggest impact on the accuracy of the DTFT spectra?
6. Provide the analysis/synthesis equation description of the DFT and explain why it is considered as a transform.
7. Explain the meaning of the N-point DFT samples $X[k]$ that are computed from N-points of a sequence $x[n]$.
8. Describe the matrix formulation of the DFT along with one important property of the DFT matrix W_N.
9. When an N-point DFT is performed on N samples of $x[n]$, which sequence is "seen" by the DFT?
10. Which concept is used to account for the inherent periodicity of the DFT and IDFT and why?
11. Describe the time-domain effect of sampling the DTFT at N equally-spaced frequencies.
12. Explain the time-domain effect of sampling the z-transform at N equally-spaced frequencies around the unit circle.
13. What is the aliasing effect in the time-domain? Under what conditions can it be eliminated?
14. Assuming no time-domain aliasing, describe the theoretical approach needed to reconstruct the DTFT from its equally-spaced samples.
15. Assuming no time-domain aliasing, describe the practical approach used in reconstruction of the DTFT from its equally-spaced samples. What is this technique known as?
16. It is said that the "zero-padding" operation in DFT provides a high-density (visually smooth) spectrum but not a high-resolution (quality) spectrum. Do you agree or disagree? Explain.
17. Assuming no time-domain aliasing, describe the theoretical approach needed to reconstruct the z-transform from its equally-spaced samples around the unit circle.
18. Describe the relationship between the CTFT, DTFT, and the DFT.
19. Under what condition on the length of a sequence does the linearity property of the DFT hold?
20. Describe the periodic, circular, and modulo-N operations.
21. Which two approaches are used to deal with the inherent periodicity of the DFT?

22. Explain circular-folding and circular-shifting operations over the primary $[0, N-1]$ interval.

23. Describe circular-even and circular-odd sequences over the primary $[0, N-1]$ interval.

24. Describe circular-conjugate-even and circular-conjugate-odd sequences over the primary $[0, N-1]$ interval.

25. What is circular convolution, why should we care about it, and how is it different from the usual (linear) convolution?

26. What is a circulant matrix and how is it related to the Toeplitz matrix?

27. We want to perform linear convolution between two finite-length sequences. Provide the necessary sequence of operations needed if we want to use the DFT for numerical computations.

28. Describe the overlap–add method of block convolution for long input sequences.

29. Describe the overlap–save method of block convolution for long input sequences.

30. Explain clearly the spectral effect of windowing a sinusoidal signal using a rectangular window.

31. What are the essential qualities of a "good" window? How are they related to the window shapes?

32. Explain the effects of window shape and length on the resolution of two sinusoids.

33. Describe the uncertainty principle in signal processing. Which signal satisfies it with equality?

34. What is a spectrogram and when is its use required in practice?

Problems

Tutorial problems

1. Let $x_c(t) = 5e^{-10t} \sin(20\pi t) u(t)$.
   - (a) Determine the CTFT $X_c(j2\pi F)$ of $x_c(t)$.
   - (b) Plot magnitude and phase of $X_c(j2\pi F)$ over $-50 \le F \le 50$ Hz.
   - (c) Use the `fft` function to approximate the CTFT computation. Choose a sampling rate to minimize aliasing and the number of samples to capture the signal waveform. Plot magnitude and phase of your approximation and compare it with the plot in (a) above.

2. A periodic signal $\tilde{x}_c(t)$ with fundamental period $T_0 = 5$ is given by $\tilde{x}_c(t) = e^{-t}$, $0 \le t \le 5$.
   - (a) Determine the CTFS c_k of $\tilde{x}_c(t)$.
   - (b) Choose sampling interval $T = 0.5$ s. Sample one period of $\tilde{x}_c(t)$ to obtain $N = 10$ samples and compute the approximate CTFS $\hat{c}_k$ using the `fft` function. Graph magnitude stem plots of c_k and $\hat{c}_k$ in one sub-plot over $-N/2 \le k \le N/2$. Similarly graph phase stem-plots in another sub-plot. Comment on your results.
   - (c) Repeat part (b) for $T = 0.25$ s and $N = 20$.
   - (d) Repeat part (b) for $T = 0.05$ s and $N = 100$.

3. Let $x[n] = n(0.9)^n u[n]$.
   (a) Determine the DTFT $\tilde{X}(e^{j\omega})$ of $x[n]$.
   (b) Choose first $N = 20$ samples of $x[n]$ and compute the approximate DTFT $\tilde{X}_N(e^{j\omega})$ using the `fft` function. Plot magnitudes of $\tilde{X}(e^{j\omega})$ and $\tilde{X}_N(e^{j\omega})$ in one plot and compare your results.
   (c) Repeat part (b) using $N = 50$.
   (d) Repeat part (b) using $N = 100$.

4. Let W_N be the $N \times N$ DFT matrix.
   (a) Determine W_N^2 and verify that it is equal to $N J_N$ where J_N is known as a *flip matrix*. Describe this matrix and its effect on Jx product.
   (b) Show that $W_N^4 = N^2 I_N$. Explain the implication of this result.
   (c) Using MATLAB determine eigenvalues of $W_N / \sqrt{N}$ for $4 \leq N \leq 10$. Some of the eigenvalues may be repeated. Can you guess a general rule for the multiplicity of eigenvalues as a function of N?

5. Determine the N-point DFTs of the following sequences defined over $0 \leq n < N$.
   (a) $x[n] = 4 - n$, $N = 8$.
   (b) $x[n] = 4\sin(0.2\pi n)$, $N = 10$.
   (c) $x[n] = 6\cos^2(0.2\pi n)$, $N = 10$.
   (d) $x[n] = 5(0.8)^n$, $N = 16$.
   (e) $x[n] = \begin{cases} 3, & n \text{ even} \\ -2. & n \text{ odd} \end{cases}$ $N = 20$

6. Show that the DFT coefficients $X[k]$ are the projections of the signal $x[n]$ on the DFT (basis) vectors $\{w_k\}$.

7. Determine DFS coefficients of the following periodic sequences:
   (a) $\tilde{x}[n] = 2\cos(\pi n/4)$.
   (b) $\tilde{x}[n] = 3\sin(0.25\pi n) + 4\cos(0.75\pi n)$.

8. Example 7.3 illustrates sampling and reconstruction of the DTFT of the signal $x[n] = a^n u[n]$.
   (a) Using MATLAB generate the plot given in Figure 7.5. Use the `fft` and `ifft` functions.
   (b) Repeat (a) for $a = 0.8$ and $N = 8$. Explain your results.
   (c) Repeat (a) for $a = 0.8$ and $N = 64$. Explain your results.

9. Example 7.3 illustrates sampling and reconstruction of the DTFT of the signal $x[n] = a^n u[n]$. Show that the aliasing error tends to zero, that is, $\tilde{x}[n]$ tends to $x[n]$ as $a \to 0$ or $N \to \infty$.

10. Starting with (7.72) and substituting $z = e^{j\omega}$, show that we can obtain (7.62).

11. Show that the N-point DFT of the circularly folded sequence $x[\langle -n \rangle_N]$ is given by $X[\langle -k \rangle_N]$.

12. Let $x[n] = x_1[n] + jx_2[n]$ where sequences $x_1[n]$ and $x_2[n]$ are real-valued.
   (a) Show that $X_1[k] = X^{cce}[k]$ and $jX_2[k] = X^{cco}[k]$.
   (b) Write a MATLAB function
   ```
   [X1,X2] = tworealDFTs(x1,x2)
   ```
   that implements the results in part (a).
   (c) Verify your function on the following two sequences: $x_1[n] = 0.9^n$, $x_2[n] = (1 - 0.8^n)$; $0 \leq n \leq 49$.

13. Let $x[n]$ be an N-point sequence with an N-point DFT $X[k]$.

(a) If N is even and if $x[n] = -x[\langle n + N/2 \rangle_N]$ for all n, then show that $X[k] = 0$ for even k.

(b) Show that if $N = 4m$ where m is an integer and if $x[n] = -x[\langle n + N/4 \rangle_N]$ for all n, then $X[k] = 0$ for $k = 4\ell$, $0 \le \ell \le \frac{N}{4} - 1$.

14. Let $x_1[n] = \{1, 2, 3, 4, 5\}$ be a 5-point sequence and let $x_2[n] = \{2, -1, 1, -1\}$ be a 4-point sequence.

(a) Determine $x_1[n] \, (5) \, x_2[n]$ using hand calculations.

(b) Verify your calculations in (a) using the `circconv` function.

(c) Verify your calculations in (a) by computing the DFTs and IDFT.

15. Let $x_1[n]$, $0 \le n \le N_1 - 1$, be an N_1-point sequence and let $x_2[n]$, $0 \le n \le N_2 - 1$, be an N_2-point sequence. Let $x_3[n] = x_1[n] * x_2[n]$ and let $x_4[n] = x_1[n] \, (N) \, x_2[n]$, $N \ge \max(N_1, N_2)$.

(a) Show that

$$x_4[n] = \sum_{\ell=-\infty}^{\infty} x_3[n + \ell N]. \tag{7.209}$$

(b) Let $e[n] = x_4[n] - x_3[n]$. Show that

$$e[n] = \begin{cases} x_3[n + N], & \max(N_1, N_2) \le N < L \\ 0, & N \ge L \end{cases}$$

where $L = N_1 + N_2 - 1$.

(c) Verify the results in (a) and (b) for $x_1[n] = \{1, 2, 3, 4\}$, $x_2[n] = \{4, 3, 2, 1\}$, and $N = 5$ and $N = 8$.

16. Circular correlation $r_{xy}[\ell]$ between two N-point sequences $x[n]$ and $y[n]$ is defined in (7.137). Show that the N-point DFT of $r_{xy}[\ell]$ is given by $X[k]Y^*[k]$.

17. Show that the DFT of the stretched sequence $x_{(M)}[n]$ is given by the periodic extension (7.140) and that the IDFT of the stretched sequence $X_{(M)}[k]$ is given by the periodic extension (7.141).

18. Show that the DFT of the sampled sequence $x^{(L)}[n]$ is given by the aliasing relation (7.143) and that the IDFT of the sampled sequence $X^{(L)}[k]$ is given by the aliasing relation (7.144).

19. The DFT of the product $w[n]x[n]$ of two sequences (windowing operation) is given by the circular convolution of their respective DFTs (7.148).

(a) Prove (7.148) by direct computation of the DFT of $w[n]x[n]$.

(b) Prove (7.148) by first starting with the circular convolution of $w[n]$ and $x[n]$ and then using duality between DFT and IDFT relations.

20. The continuous-time windowing operation is given in (7.161).

(a) Show that (7.161) when viewed as a system with $x_c(t)$ as input and $\hat{x}_c(t)$ as output is a linear but time-varying system.

(b) Let

$$\hat{w}_c(t) = \begin{cases} 1, & 0 \le t \le T_0 \\ 0, & \text{otherwise} \end{cases}$$

where $(L-1)T \leq T_0 \leq LT$ and T is the sampling interval. Then show that (7.160) is equal to (7.162).

21. If $\hat{x}_c(t) = w_c(t)x_c(t)$ then show that the CTFT of $\hat{x}_c(t)$ is given by the convolution integral (7.170).

22. Prove the scaling property (7.172) of the CTFT.

23. Derive the Schwarz inequality given in (7.179).

24. The CTFT $W(j\Omega)$ of a generic window function $w(t)$ is given in (7.189) in which a and b are design parameters and $W_R(j\Omega)$ is the CTFT of the rectangular window.
    (a) For the choice of $a = b = 0.5$ and using ICTFT show that the resulting Hann window function is given by (7.190).
    (b) For the choice of $a = 0.54$ and $b = 0.23$ and using ICTFT show that the resulting Hamming window function is given by (7.191).

25. Show that the mainlobes of the sinc function (7.168) and the Dirichlet function (7.196) are almost identical in the region around the mainlobe for large L.

26. The 2D-DFT $X[k, \ell]$ of size $M \times N$ image $x[m, n]$, $0 \leq m < M$, $0 \leq n < N$ is defined as

$$X[k, \ell] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m, n]W_M^{mk}W_N^{n\ell},\qquad(7.210)$$

while the 2D-IDFT is defined as

$$x[m, n] = \frac{1}{MN} \sum_{k=0}^{M-1} \sum_{\ell=0}^{N-1} x[m, n]W_M^{-mk}W_N^{-n\ell}.$$

Similarly to 1D-DFT, these 2D signals are rectangularly periodic outside the primary region. The 2D-DFT is used extensively in image processing for frequency-domain filtering operations.
    (a) Show that (7.210) can be computed using nested row/column 1D DFTs. Thus the 1D-FFT algorithm can be used to compute 2D-DFT.
    (b) Let $x[m, n] = (0.9)^{m+n}u[m, n]$. Using the `fft` function compute the 100×100 size 2D-DFT of $x[m, n]$ and display its magnitude as an image.
    (c) If $x[m, n]$ can be expressed as $x[m, n] = x_1[m]x_2[n]$, then show that $X[k, \ell]$ can also be expressed as $X[k, \ell] = X_1[k]X_2[\ell]$ where $X_1[k]$ and $X_2[\ell]$ are M- and N-point 1D-DFTs, respectively.
    (d) Using the result in (c) above, determine the 2D-DFT of the signal in (b) and compare your results.

Basic problems

27. Let $x_c(t) = 10te^{-20t}\cos(20\pi t)u(t)$.
    (a) Determine the CTFT $X_c(j2\pi F)$ of $x_c(t)$.
    (b) Plot magnitude and phase of $X_c(j2\pi F)$ over $-75 \leq F \leq 75$ Hz.
    (c) Use the `fft` function to approximate the CTFT computation. Choose sampling rate to minimize aliasing and the number of samples to capture the signal waveform. Plot magnitude and phase of your approximation and compare it with the plot in (a) above.

28. A periodic signal $\tilde{x}_c(t)$ with fundamental period $T_0 = 5$ is given by $\tilde{x}_c(t) = te^{-0.5t}$, $0 \le t \le 5$.
    (a) Determine the CTFS c_k of $\tilde{x}_c(t)$.
    (b) Choose sampling interval $T = 0.1$ s. Sample one period of $\tilde{x}_c(t)$ to obtain $N = 50$ samples and compute the approximate CTFS $\hat{c}_k$ using the `fft` function. Graph magnitude stem plots of c_k and $\hat{c}_k$ in one sub-plot over $-N/2 \le k \le N/2$. Similarly graph phase stem-plots in another sub-plot. Comment on your results.
    (c) Repeat part (b) for $T = 0.05$ s and $N = 100$.
    (d) Repeat part (b) for $T = 0.01$ s and $N = 500$.

29. Let $x[n] = 10(0.5)^n \sin(0.1\pi n)u[n]$.
    (a) Determine the DTFT $\tilde{X}(e^{j\omega})$ of $x[n]$.
    (b) Choose first $N = 10$ samples of $x[n]$ and compute the approximate DTFT $\tilde{X}_N(e^{j\omega})$ using the `fft` function. Plot magnitudes of $\tilde{X}(e^{j\omega})$ and $\tilde{X}_N(e^{j\omega})$ in one plot and compare your results.
    (c) Repeat part (b) using $N = 50$.
    (d) Repeat part (b) using $N = 100$.

30. Consider the complex vector w_k which is the kth column of the DFT matrix W_N given in (7.33).
    (a) For $N = 6$, sketch the components of w_k as vectors from the origin to the unit circle for $k = 0, 1, 2, 3, 4, 5$.
    (b) Using these sketches, determine the products $w_1^H w_2$, $w_1^H w_1$, and $w_2^H w_2$.
    (c) Verify that $w_k^H w_m = N\delta[k - m]$, $0 \le k, m \le N - 1$.

31. Compute and plot the N-point DFT and IDFT of the following sequences in the range $-(N - 1) \le n \le (2N - 1)$:
    (a) $x[n] = \delta[n]$, $N = 8$.
    (b) $x[n] = n$, $N = 10$.
    (c) $x[n] = \cos(6\pi n/15)$, $N = 30$.
    (d) $x[n] = \cos(0.1\pi n)$, $N = 30$.

32. Let $\tilde{x}[n]$ be a periodic sequence with fundamental period N and let $\tilde{X}[k]$ be its DFS. Let $\tilde{x}_3[n]$ be periodic with period $3N$ consisting of three periods of $\tilde{x}[n]$ and let $\tilde{X}_3[k]$ be its DFS.
    (a) Determine $\tilde{X}_3[k]$ in terms of $\tilde{X}[k]$.
    (b) Let $\tilde{x}[n] = \{\ldots, 1, 3, \underset{\uparrow}{1}, 3, 1, 3, 1, 3, \ldots\}$. Verify your results in part (a) by explicitly computing $\tilde{X}[k]$ and $\tilde{X}_3[k]$.

33. Explain the result of `plot(dftmtx(16))`.

34. Let $x[n] = (0.8)^{|n|}$. A periodic sequence $\tilde{x}[n]$ is obtained using the aliasing

$$\tilde{x}[n] = \sum_{\ell=-\infty}^{\infty} x[n - 8\ell].$$

    (a) Determine and plot the DTFT $\tilde{X}(e^{j\omega})$ of $x[n]$.
    (b) Determine and stem-plot the DFS $\tilde{X}[k]$ of $\tilde{x}[n]$. How is it related to the DTFT $\tilde{X}(e^{j\omega})$?
    (c) Repeat (a) and (b) for $x[n] = (0.4)^{|n|}$.

35. Let $x[n] = 0.8^n u[n] + (-0.8)^n u[n]$.

(a) Determine the DTFT $\tilde{X}(e^{j\omega})$.

(b) Let $G[k] = \tilde{X}(e^{j\frac{2\pi k}{10}})$. Determine $g[n]$ without computing the IDFT.

36. The following 8-point sequences are defined over $0 \leq n \leq 7$. Without computing their DFTs, determine which have real-valued 8-point DFTs and which have 8-point imaginary-valued DFTs. If DFTs are complex valued, explain why.

(a) $x_1[n] = \{0, -3, 1, -2, 0, 2, -1, 3\}$.

(b) $x_2[n] = \{5, 2, -9, 4, 7, 4, -9, 2\}$.

(c) $x_3[n] = \{8, -3, 1, -2, 6, 2, -1, 3\}$.

(d) $x_4[n] = \{0, 1, 3, -2, 5, 2, -3, 1\}$.

(e) $x_5[n] = \{10, 5, -7, -4, 5, -4, -7, 5\}$.

37. Consider the real-valued sequence

$$x[n] = \begin{cases} \cos(0.25\pi n + \pi/6), & 0 \leq n \leq 99 \\ 0. & \text{otherwise} \end{cases}$$

(a) Determine and plot the DTFT $\tilde{X}(e^{j\omega})$ of $x[n]$.

(b) Determine the 100-point DFT $X[k]$ of $x[n]$ and superimpose its samples on the DTFT plot in part (a).

(c) Repeat part (b) using $N = 200$.

(d) Discuss your results in parts (a) through (c).

38. The first five values of the 9-point DFT of a real-valued sequence $x[n]$ are given by

$$\{4, 2 - j3, 3 + j2, -4 + j6, 8 - j7\}.$$

Without computing IDFT and then DFT but using DFT properties only, determine the DFT of each of the following sequences:

(a) $x_1[n] = x[\langle n + 2 \rangle_9]$, (b) $x_2[n] = 2x[\langle 2 - n \rangle_9]$, (c) $x_3[n] = x[n] \textcircled{9} x[\langle -n \rangle_9]$, (d) $x_4[n] = x^2[n]$, (e) $x_5[n] = x[n]e^{-j4\pi n/9}$.

39. Let $x[n]$ be a real-valued N-point sequence with N-point DFT $X[k]$.

(a) Show that $X[0]$ is real-valued.

(b) Show that $X[\langle N - k \rangle_N] = X^*[k]$.

(c) Show that $X[N/2]$ is real-valued if N is even.

40. Determine the relationship between 8-point DFTs of the following 8-point sequences:

$$x_1[n] = \{\underset{\uparrow}{a}, 0, b, c, 0, d, 0, 0\}, \quad x_2[n] = \{\underset{\uparrow}{d}, 0, c, b, 0, a, 0, 0\}.$$

Verify your result by choosing $a = 1$, $b = 2$, $c = 3$, and $d = 4$.

41. Let $x_1[n] = \{-2, 1, \underset{\uparrow}{-3}, -5, 6, 8\}$ be a 6-point sequence and let $x_2[n] = \{\underset{\uparrow}{1}, 2, 3, 4\}$ be a 4-point sequence.

(a) Determine $x_1[n] \textcircled{7} x_2[n]$ using hand calculations.

(b) Verify your calculations in (a) using the `circonv` function.

(c) Verify your calculations in (a) by computing the DFTs and IDFT.

42. Let $x_1[n] = 0.9^n$, $0 \leq n \leq 9$ and let $x_2[n] = n(0.6)^n$, $0 \leq n \leq 9$. Let $x_3[n] = x_1[n] * x_2[n]$.

(a) Determine $x_3[n]$ using MATLAB.

(b) Determine $x_4[n] = x_1[n] \textcircled{15} x_2[n]$.

(c) Let $e[n] = x_4[n] - x_3[n]$. Determine $e[n]$ and verify that it is equal to $x_3[n + 15]$.

43. Let $x_1[n]$ be an N_1-point and $x_2[n]$ be an N_2-point sequence. Let $N \geq \max(N_1, N_2)$. Their N-point circular convolution is shown to be equal to the aliased version of their linear convolution in (7.209) in Problem 15. This result can be used to compute the circular convolution via the linear convolution.

(a) Develop a MATLAB function
```
y = lin2circonv(x,h)
```
that implements this approach.

(b) For $x[n] = \{1, 2, 3, 4\}$ and $h[n] = \{1, -1, 1, -1\}$, determine their 4-point circular convolution using the `lin2circonv` function and verify using the `circonv` function.

44. Consider the two finite-length sequences:

$$x_1[n] = \{\underset{\uparrow}{1}, -2, 1, -3\}, \quad x_2[n] = \{\underset{\uparrow}{0}, 2, -1, 0, 0, 4\}.$$

(a) Determine the linear convolution $x_1[n] * x_2[n]$.

(b) Determine the circular convolution $x_1[n] \textcircled{6} x_2[n]$.

(c) What should be the smallest value of N so that N-point circular convolution is equal to the linear convolution?

45. Let $x_c(t)$ be a continuous-time signal with CTFT $X_c(j\Omega)$.

(a) Define time-duration ΔT_1 and bandwidth ΔF_1 as

$$\Delta T_1 \triangleq \frac{\int_{-\infty}^{\infty} x_c(t) \, dt}{x_c(0)}, \quad \Delta F_1 \triangleq \frac{\int_{-\infty}^{\infty} X_c(j2\pi F) \, dF}{X_c(0)}.$$

Show that $\Delta T_1 \Delta F_1 = 1$.

(b) Let $x_{c_1}(t) = u(t + 1) - u(t - 1)$ and $x_{c_2}(t) = \cos(\pi t)[u(t + 1) - u(t - 1)]$. Evaluate ΔT_1, ΔF_1, and their product for these two waveforms and explain for which waveform the definition of time-duration and bandwidth in (a) is reasonable.

(c) To avoid problems with waveforms that can become negative, let the time-duration ΔT_2 and bandwidth ΔF_2 be defined as

$$\Delta T_2 \triangleq \frac{\int_{-\infty}^{\infty} |x_c(t)| \, dt}{|x_c(0)|}, \quad \Delta F_2 \triangleq \frac{\int_{-\infty}^{\infty} |X_c(j2\pi F)| \, dF}{|X_c(0)|}.$$

Show that $\Delta T_2 \Delta F_2 \geq 1$.

(d) Repeat part (b) for the time-duration and bandwidth defined in (c).

46. The 2D-DFT introduced in Problem 26 can be efficiently computed using the `X=fft2(x,M,N)` function in MATLAB which computes $M \times N$ 2D-DFT of an image array `x` in array `X`. Similarly, `x=ifft2(X)` computes the inverse 2D-DFT.

(a) Consider the 256×256 "Lena" image available at the book website. Compute its 2D-DFT and display the log-magnitude and phase as images. Use `fftshift` so that the zero frequency is at the center of the display.

(b) Set the phase of the above 2D-DFT equal to zero and then compute the 2D-IDFT using the magnitude array. Display the resulting image and comment on your observations.

(c) Set the magnitude array to value 128 for all pixels. Use it along with the phase array of part (a) to compute the 2D-IDFT. Display the resulting image and comment on the importance of Fourier transform phase in image processing.

Assessment problems

47. The DFT (7.21) and the IDFT (7.22) share the same complex-exponential term W_N^{nk} but with different signs.
    (a) Show that it is possible to express (7.22) as a DFT operation with additional processing steps. This approach can be used to obtain both transforms using one software function.
    (b) Write a MATLAB function x=myifft(X) that implements your procedure in part (a) above. Use fft to implement the DFT computations.
    (c) Let $x[n] = \sin(0.1\pi n)$, $0 \le n \le 9$. Determine its DFT using the fft and then its IDFT using the myifft function to verify its accuracy.

48. Consider the following three periodic sequences (note that the time origin is not given):

$$\tilde{x}_1[n] = \{\ldots, 0, -1, 1, 1, 0, -1, 1, 1, 0, -1, \ldots\},$$

$$\tilde{x}_2[n] = \{\ldots, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, \ldots\},$$

$$\tilde{x}_3[n] = \{\ldots, -1, -1, 1, 1, -1, -1, 1, 1, \ldots\}.$$

   (a) For which sequence is it possible to have all DFS values real-valued? If it is possible, explain how.
   (b) For which sequence is it possible to have all (except for $k = \ell N$) DFS values imaginary-valued? If it is possible explain how.
   (c) For which sequence are the DFS coefficients zero for $k = \pm 2, \pm 4$?

49. Let $x[n] = 0.8^n \{u[n] - u[n - 10]\}$.
   (a) Determine and plot the magnitude of the DTFT $\tilde{X}(e^{j\omega})$.
   (b) Let $X[k]$ be the 10-point DFT of $x[n]$. Determine $X[k]$ and indicate its magnitude on the plot of $\tilde{X}(e^{j\omega})$ in part (a).
   (c) Let $y[n] = x[n]e^{-j\frac{\pi n}{N}}$. Determine its $Y[k]$ and indicate its magnitude (using a different color) on the plot of $\tilde{X}(e^{j\omega})$ in part (a). Comment on your observation.
   (d) Based on your observation in (c), how would you recover the signal $x[n]$ given the DFT samples $Y[k]$?

50. Let the DTFT $\tilde{X}(e^{j\omega})$ of a sequence $x[n]$ be given by

$$\tilde{X}(e^{j\omega}) = \frac{3}{5 - 4\cos(\omega)}.$$

It is sampled at N equally-spaced frequencies to obtain $X[k] = X(e^{j\frac{2\pi k}{N}})$ for $0 \le k \le N - 1$.
   (a) For $N = 16$ determine and stem-plot the sequence $x_1[n]$ from $-8 \le n \le 8$ by taking the IDFT of $X[k]$.

(b) For $N = 32$ determine and stem-plot the sequence $x_2[n]$ from $-16 \leq n \leq 16$ by taking the IDFT of $X[k]$.

(c) From your observations of the plots in (a) and (b) above, what do you think the original sequence $x[n]$ is? Verify your answer by computing its DTFT and comparing it with $\tilde{X}(e^{j\omega})$.

51. Consider the real-valued sequence

$$
x[n] = \begin{cases} \sin(0.6\pi n + \pi/3), & 0 \leq n \leq 99 \\ 0. & \text{otherwise} \end{cases}
$$

(a) Determine and plot the DTFT $\tilde{X}(e^{j\omega})$ of $x[n]$.
(b) Determine the 100-point DFT $X[k]$ of $x[n]$ and superimpose its samples on the DTFT plot in part (a).
(c) Repeat part (b) using $N = 200$.
(d) Discuss your results in parts (a) through (c).

52. Consider the following two sequences

$$
x_1[n] = \{\underset{\uparrow}{2}, -1, 0, 1, 3, 0, 4\}, \quad x_2[n] = \{\underset{\uparrow}{1}, 3, 0, 4, 2, -1, 0\}.
$$

If their N-point DFTs are related by $X_1[k] = X_2[k]e^{-j2\pi k\ell/7}$, determine the smallest positive ℓ.

53. Let $X[k]$ be the N-point DFT of an N-point sequence $x[n]$.
(a) If $x[n]$ satisfies the condition $x[n] = x[\langle N - 1 - n \rangle_N]$, show that $X[N/2] = 0$ for N even.
(b) If $x[n]$ satisfies the condition $x[n] = -x[\langle N - 1 - n \rangle_N]$, show that $X[0] = 0$ for N even.
(c) If $x[n]$ satisfies the condition $x[n] = x[\langle n + M \rangle_N]$ where $N = 2M$, show that $X[2\ell + 1] = 0$ for $\ell = 0, 1, \ldots, M - 1$.

54. Show Parseval's theorem for the DFT given in Table 7.4 and use it to prove Parseval's relation given in the same Table.

55. The 2D-DFT introduced in Problem 26 possesses properties that are similar to those of 1D-DFT.
(a) Show that the image $x[\langle m - m_0 \rangle_M, \langle n - n_0 \rangle_N]$ has the 2D-DFT given by $W_M^{km_0} W_N^{\ell n_0} X[k, \ell]$.
(b) Show that the image $x[\langle -m \rangle_M, \langle -n \rangle_N]$ has the 2D-DFT given by $X^*[k, \ell]$.
(c) The 2D $M \times N$ circular convolution between two images $x[m, n]$ and $h[m, n]$ is defined as

$$
y[m, n] \triangleq x[m, n] \,\textcircled{M}\textcircled{N}\, h[m, n]
$$

$$
= \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} x[i, j] \times h[\langle m - i \rangle_M, \langle n - j \rangle_N]. \tag{7.211}
$$

Show that 2D-DFT is given by $Y[k, l] = X[k, l]H[k, l]$.

Review problems

56. Consider an analog integrator filter with impulse response and frequency response functions given by

$$h_c(t) = Ae^{-At}u(t),$$

$$H_c(j2\pi F) = \mathcal{F}[h_c(t)] = \frac{1}{1 + j(F/F_c)},$$

where $A \triangleq 1/(RC)$ and $F_c \triangleq 1/(2\pi RC)$ is the 3-dB cutoff frequency. We wish to simulate this analog filter with a discrete-time filter.

The first approach to simulate an analog filter, known as *impulse-invariance* (see Section 11.3.1), is to obtain a discrete-time filter by sampling the impulse response $h_c(t)$. The result is a discrete-time filter with impulse response

$$h[n] = A(e^{-AT})^n u[n].$$

(a) Show that the resulting digital filter is given by the difference equation

$$y[n] = e^{-AT}y[n-1] + Ax[n].$$

(b) Graph impulse responses $h[n]$ and $h_c(t)$ on the same plot for $F_c = 2$ Hz and $F_s = 20$ Hz.
(c) Graph magnitude responses of the analog integrator and its digital simulator for $F_c = 2$ Hz and $F_s = 20$ Hz on the same plot and comment on your observations.

The second approach to simulate an analog filter, known as *frequency sampling* (see Section 10.4), is to uniformly sample the segment $H_0(j2\pi F)$ of $H_c(j2\pi F)$ from $-F_s/2$ to $F_s/2$, where F_s is the sampling frequency to be used at the simulation.

(d) For $F_s = 20$ Hz and $N = 20$, obtain (complex-valued) samples $H[k] = H_0(j2\pi F_s k/N)$ for $-10 \leq k \leq 9$. Determine the impulse response $h[n]$ using IDFT and the `fftshift` operations. Graph the impulse responses $h[n]$ and $h_c(t)$ on the same plot and comment on the result.
(e) Using zero-padding compute the smooth frequency response of the digital simulator and graph its magnitude and phase along with that of the analog integrator in the same plot over $-10 \leq F \leq 10$ Hz. Comment on your observations.

57. To illustrate and analyze the effects of time-domain windowing, consider the signal

$$x_c(t) = \sum_{k=1}^{K} \cos(2\pi F_k t),$$

where the set of frequencies $\{F_k\}_1^K$ are equally spaced between $F_L = F_1$ to $F_H = F_K$ given by $F_k = F_L + (k-1)(F_H - F_L)/(K-1)$.

(a) For $F_L = 40$ Hz, $F_H = 60$ Hz, and $K = 10$, determine and plot $X_c(j2\pi F)$ from -100 to 100 Hz.

Let $\hat{x}_c(t) = w_c(t)x_c(t)$ be the windowed signal given by

$$\hat{x}_c(t) = \sum_{k=1}^{K} w_c(t) \cos(2\pi F_k t). \tag{7.212}$$

(b) After taking the CTFT of (7.212) or using the frequency-shift property, show that the CTFT of $\hat{x}_c(t)$ is given by

$$\hat{X}_c(j2\pi F) = \frac{1}{2} \sum_{k=1}^{K} \{W_c[j2\pi(F - F_k)] + W_c[j2\pi(F + F_k)]\}, \qquad (7.213)$$

where $W_c(j\Omega)$ is the CTFT of the window.

(c) Using a rectangular window $w_R(t) = 1$, $-0.1 \le t \le 0.1$ s and the set of frequencies given in (a) above, determine and plot the CTFT of the windowed signal $\hat{x}_c(t)$. On a separate graph, also plot the CTFT of the individual windowed sinusoidal components (see Figure 7.25(c)). Comment on your observation.

(d) If the number K of sinusoids in the fixed range (F_L, F_H) increases, a larger and more closely spaced set of shifted copies of $W_c(j2\pi F)$ is used to form $\hat{X}_c(j2\pi F)$. Explain the CTFT spectra $X_c(j2\pi F)$ that we will obtain in the limit as $K \to \infty$. Furthermore, explain the CTFT $\hat{X}_c(j2\pi F)$ in (7.213) in light of the limit $X_c(j2\pi F)$ and the window CTFT $W_c(j2\pi F)$.

58. Let a 2D filter impulse response $h[m, n]$ be given by

$$h[m, n] = \begin{cases} \frac{1}{2\pi\sigma^2} e^{-\frac{m^2+n^2}{2\sigma^2}}, & \begin{aligned} -128 &\le m \le 127 \\ -128 &\le n \le 127 \end{aligned} \\ 0, & \text{otherwise} \end{cases}$$

where σ is a parameter. For this problem use the "Lena" image.

(a) For $\sigma = 4$, determine $h[m, n]$ and compute its 2D-DFT $H[k, \ell]$ via the `fft2` function taking care of shifting the origin of the array from the middle to the beginning (using the `ifftshift` function). Show the log-magnitude of $H[k, \ell]$ as an image.

(b) Process the "Lena" image in the frequency domain using the above $H[k, \ell]$. This will involve taking 2D-DFT of the image, multiplying the two DFTs and then taking the inverse of the product. Comment on the visual quality of the resulting filtered image.

(c) Repeat (a) and (b) for $\sigma = 32$ and comment on the resulting filtered image as well as the difference between the two filtered images.

(d) The filtered image in part (c) also suffers from an additional distortion due to a spatial-domain aliasing effect in the circular convolution. To eliminate this artifact, consider both the image and the filter $h[m, n]$ as 512×512 size images using zero-padding in each dimension. Now perform the frequency-domain filtering and comment on the resulting filtered image.

(e) Repeat part (b) for $\sigma = 4$ but now using the frequency response $1 - H[k, \ell]$ for the filtering. Compare the resulting filtered image with that in (b).

8 Computation of the Discrete Fourier Transform

This chapter is primarily concerned with algorithms for efficient computation of the Discrete Fourier Transform (DFT). This is an important topic because the DFT plays an important role in the analysis, design, and implementation of many digital signal processing systems. Direct computation of the N-point DFT requires computational cost proportional to N^2. The most important class of efficient DFT algorithms, known collectively as *Fast Fourier Transform (FFT) algorithms*, compute *all* DFT coefficients as a "block" with computational cost proportional to $N\log_2 N$. However, when we only need a few DFT coefficients, a few samples of DTFT, or a few values of z-transform, it may be more efficient to use algorithms based on linear filtering operations, like the Goertzel algorithm or the chirp z-transform algorithm.

Although many computational environments provide FFT algorithms as built-in functions, the user should understand the fundamental principles of FFT algorithms to make effective use of these functions. The details of FFT algorithms are important to designers of real-time DSP systems in either software or hardware.

Study objectives

After studying this chapter you should be able to:

- Understand the derivation, operation, programming, and use of decimation-in-time and decimation-in-frequency radix-2 FFT algorithms.
- Understand the general principles underlying the development of FFT algorithms and use them to make effective use of existing functions, evaluate competing algorithms, or guide the selection of algorithms for a particular application or computer architecture.

8.1   Direct computation of the Discrete Fourier Transform

The DFT of a finite-length sequence of length N is defined by (see Chapter 7)

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{kn}, \quad k = 0, 1, \ldots, N-1 \tag{8.1}$$

where $W_N = \mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$, the root-of-unity, is also known as the *twiddle factor*. The inverse DFT is given by

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]W_N^{-kn}. \quad n = 0, 1, \ldots, N-1 \tag{8.2}$$

The two equations differ only in the sign of the exponent of W_N and the scaling factor $1/N$. Therefore, algorithms developed for computation of the DFT can easily be modified for computation of the inverse DFT. For example, the inverse DFT (8.2) can be written as

$$x[n] = \frac{1}{N} \left[\sum_{k=0}^{N-1} X^*[k]W_N^{kn} \right]^*. \quad n = 0, 1, \ldots, N-1 \tag{8.3}$$

Therefore, the inverse DFT of $X[k]$ can be evaluated by scaling by $1/N$ the complex conjugate of the DFT of $X^*[k]$; two additional approaches are discussed in Problems 16 and 33.

   If we use a computer language which supports complex arithmetic we can directly evaluate formula (8.1) using a double loop. If only real arithmetic is supported by the compiler, we use the formulas

$$X_{\mathrm{R}}[k] = \sum_{n=0}^{N-1} \left[x_{\mathrm{R}}[n]\cos\left(\frac{2\pi}{N}kn\right) + x_{\mathrm{I}}[n]\sin\left(\frac{2\pi}{N}kn\right) \right], \tag{8.4a}$$

$$X_{\mathrm{I}}[k] = -\sum_{n=0}^{N-1} \left[x_{\mathrm{R}}[n]\sin\left(\frac{2\pi}{N}kn\right) - x_{\mathrm{I}}[n]\cos\left(\frac{2\pi}{N}kn\right) \right]. \tag{8.4b}$$

The DFT coefficients $X[k]$ in (8.1), like any matrix by vector product, can be evaluated using a double loop as shown by the MATLAB function `dftdirect` in Figure 8.1. The total cost of computing *all* $X[k]$ coefficients is approximately N^2 operations when we define an operation to be the work required to execute the statement `S=S+W(k,n)*x(n)`. More specifically, we need to fetch `W(k,n)`, `x(n)`, and the "old" value of `S` from memory; compute the product `W(k,n)*x(n)` and the sum `S+W(k,n)*x(n)`; and store the result as the "new" value of `S`. The cost for initialization and other overhead operations is negligible for large values of N and hence will be ignored. We will say that the computational complexity of the direct DFT algorithm is "of the order of N^2" or $\mathrm{O}(N^2)$ operations for short. For a DFT of fixed size N the values of coefficients W_N^{kn} are usually computed and stored

```
function Xdft=dftdirect(x)
% Direct computation of the DFT
N=length(x); Q=2*pi/N;
for k=1:N
      S=0;
      for n=1:N
            W(k,n)=exp(-j*Q*(k-1)*(n-1));
            S=S+W(k,n)*x(n);
      end
      Xdft(k)=S;
end
```

Figure 8.1 MATLAB function for direct computation of DFT.

before the execution of the algorithm; therefore, they are not included when we evaluate the computational complexity of a DFT algorithm.

Since the computation time for $O(N^2)$ algorithms becomes very large for large values of N, we are interested in computational algorithms that reduce the number of operations. In the past, computational complexity was primarily dominated by the number of multiplications followed by the number of additions. Thus, we traditionally used the number of multiplications and additions as a measure of computational complexity. However, with current computer technology, computer architecture, compiler design, and other similar factors are also considered in implementing (8.1) for reducing the overall computational complexity. In this chapter we emphasize the complexity as explained above.

We next show how to exploit the periodicity and complex conjugate symmetry properties of W_N^{kn} to develop a family of $O(N\log_2 N)$ algorithms, known collectively as Fast Fourier Transform (FFT) algorithms. These algorithms use a "divide-and-conquer" approach, that is, they decompose the DFT of a sequence of length N into smaller-length DFTs that are "merged" to form the N-point DFT. This procedure may be applied again to the smaller DFTs. Algorithms that decompose the sequence $x[n]$ into smaller sequences are known as *decimation-in-time* FFTs; algorithms that decompose the DFT $X[k]$ into smaller sequences are known as *decimation-in-frequency* FFTs. The next section, where we introduce these ideas using a matrix framework, can be skipped without loss of continuity.

8.2   The FFT idea using a matrix approach

Most algorithms for efficient computation of the DFT exploit the periodicity and symmetry properties of the twiddle factor W_N^{kn}. These properties are

$$W_N^{kn} = W_N^{k(n+N)} = W_N^{(k+N)n}, \qquad \text{(periodicity in } k \text{ and } n) \qquad (8.5)$$

$$W_N^{k(N-n)} = W_N^{-kn} = \left(W_N^{kn}\right)^*. \qquad \text{(complex conjugate symmetry)} \qquad (8.6)$$

The periodicity and symmetry properties of the DFT matrix can be visualized by representing each twiddle factor W_N^{kn} as a phasor in the complex plane. For the W_8 matrix this pictorial representation takes the following form in which the phasor angle of 0 is shown vertically up and the phasor rotates in the clockwise direction:

$$
W_8 = \begin{bmatrix}
\uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow & \uparrow \\
\uparrow & \nearrow & \rightarrow & \searrow & \downarrow & \swarrow & \leftarrow & \nwarrow \\
\uparrow & \rightarrow & \downarrow & \leftarrow & \uparrow & \rightarrow & \downarrow & \leftarrow \\
\uparrow & \searrow & \leftarrow & \nearrow & \downarrow & \nwarrow & \rightarrow & \swarrow \\
\uparrow & \downarrow & \uparrow & \downarrow & \uparrow & \downarrow & \uparrow & \downarrow \\
\uparrow & \swarrow & \rightarrow & \nwarrow & \downarrow & \nearrow & \leftarrow & \searrow \\
\uparrow & \leftarrow & \downarrow & \rightarrow & \uparrow & \leftarrow & \downarrow & \rightarrow \\
\uparrow & \nwarrow & \leftarrow & \swarrow & \downarrow & \searrow & \rightarrow & \nearrow
\end{bmatrix}. \tag{8.7}
$$

We next show how to exploit the structure of the matrix W_N to develop efficient algorithms for computation of the DFT. We start by expressing the N-point DFT, for $N = 8$, in matrix form. The results is

$$
\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & W_8 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\
1 & W_8^2 & W_8^4 & W_8^6 & 1 & W_8^2 & W_8^4 & W_8^6 \\
1 & W_8^3 & W_8^6 & W_8 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\
1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 \\
1 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8 & W_8^6 & W_8^3 \\
1 & W_8^6 & W_8^4 & W_8^2 & 1 & W_8^6 & W_8^4 & W_8^2 \\
1 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8
\end{bmatrix}
\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}. \tag{8.8}
$$

Note that changing the order of the matrix columns and the elements of the right hand side vector in the same way does not alter the final result. Thus, we can put the columns for the samples $x[0], x[2], x[4], x[6]$ (even) first, and then the columns for $x[1], x[3], x[5], x[7]$ (odd). If we use the identity $W_8^4 = -1$ and rearrange the matrix equation (8.8) by grouping even and odd terms, we obtain

$$
\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix} =
\left[\begin{array}{cccc|cccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & W_8^2 & W_8^4 & W_8^6 & W_8 & W_8^3 & W_8^5 & W_8^7 \\
1 & W_8^4 & 1 & W_8^4 & W_8^2 & W_8^6 & W_8^2 & W_8^6 \\
1 & W_8^6 & W_8^4 & W_8^2 & W_8^3 & W_8 & W_8^7 & W_8^5 \\
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & W_8^2 & W_8^4 & W_8^6 & -W_8 & -W_8^3 & -W_8^5 & -W_8^7 \\
1 & W_8^4 & 1 & W_8^4 & -W_8^2 & -W_8^6 & -W_8^2 & -W_8^6 \\
1 & W_8^6 & W_8^4 & W_8^2 & -W_8^3 & -W_8 & -W_8^7 & -W_8^5
\end{array}\right]
\begin{bmatrix} x[0] \\ x[2] \\ x[4] \\ x[6] \\ x[1] \\ x[3] \\ x[5] \\ x[7] \end{bmatrix}, \tag{8.9}
$$

where we have partitioned the 8×8 DFT matrix as a 2×2 matrix with 4×4 blocks. If we use the identity $W_8^2 = W_4$, we can express the previous equation as follows:

$$\begin{bmatrix} X_T \\ X_B \end{bmatrix} = \begin{bmatrix} W_4 & D_8 W_4 \\ W_4 & -D_8 W_4 \end{bmatrix} \begin{bmatrix} x_E \\ x_O \end{bmatrix}, \tag{8.10}$$

where W_4, the 4-point DFT matrix, and the diagonal matrix D_8 are given by

$$W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4 & W_4^2 & W_4^3 \\ 1 & W_4^2 & 1 & W_4^2 \\ 1 & W_4^3 & W_4^2 & W_4 \end{bmatrix} \quad \text{and} \quad D_8 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & W_8 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{bmatrix}. \tag{8.11}$$

We use the subscripts T, B, E, and O to denote the top, bottom, even, and odd entries of a vector, respectively. If we define the $\frac{N}{2}$-point DFTs ($N = 8$)

$$X_E = W_{\frac{N}{2}} x_E, \tag{8.12a}$$

$$X_O = W_{\frac{N}{2}} x_O, \tag{8.12b}$$

the N-point DFT in (8.10) can be expressed as

$$X_T = X_E + D_N X_O, \tag{8.13a}$$

$$X_B = X_E - D_N X_O. \tag{8.13b}$$

We now have all the ingredients we need for a divide-and-conquer algorithm: we have expressed the N-point DFT in terms of two $\frac{N}{2}$-point DFTs. This is illustrated diagrammatically in Figure 8.2. Merging the two DFTs requires $\frac{N}{2}$ multiplications and N additions.
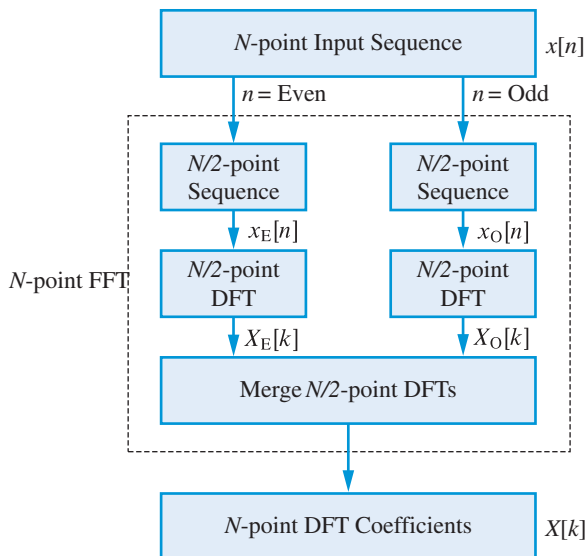
Figure 8.2 A divide-and-conquer approach for recursive computation of the DFT.

```
function Xdft = fftrecur(x)
% Recursive computation of the DFT using divide & conquer
% N should be a power of 2
N = length(x);
if N ==1
  Xdft = x;
else
    m = N/2;
    XE = fftrecur(x(1:2:N));
    XO = fftrecur(x(2:2:N));
    W = exp(-2*pi*sqrt(-1)/N).^(0:m-1)';
    temp = W.*XO;
    Xdft = [ XE+temp ; XO-temp ];
end
```

Figure 8.3 MATLAB function for recursive computation of the DFT.

If $\frac{N}{2}$ is even, we can express each $\frac{N}{2}$-point DFT in terms of two $\frac{N}{4}$-point DFTs. Merging of four $\frac{N}{4}$-point DFTs requires $\frac{N}{2}$ multiplications and N additions. If $N = 2^v$ we can repeat this process until $N = 1$; the one-point DFT is trivial: $X[0] = x[0]$, which can be checked easily in (8.1) for $N = 1$. Thus, we have replaced the computation of the N-point DFT with $v = \log_2 N$ merging operations. The result is an algorithm with computational complexity proportional to $N\log_2 N$ or $O(N\log_2 N)$. A MATLAB function, called fftrecur, for recursive computation of the DFT using this divide-and-conquer approach, based on Van Loan (2000), is given in Figure 8.3. Clearly, the FFT algorithm obtained belongs to the decimation-in-time class.

A *dual* divide-and-conquer algorithm can be obtained by reordering the equations for the DFT so that we compute first the even DFT coefficients and then the odd DFT coefficients. This yields the following matrix equation:

$$\begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \\ X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_8^2 & W_8^4 & W_8^6 & 1 & W_8^2 & W_8^4 & W_8^6 \\ 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 \\ 1 & W_8^6 & W_8^4 & W_8^2 & 1 & W_8^6 & W_8^4 & W_8^2 \\ 1 & W_8 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ 1 & W_8^3 & W_8^6 & W_8 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\ 1 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8 & W_8^6 & W_8^3 \\ 1 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}. \tag{8.14}$$

Since the matrix in (8.14) is the transpose of the matrix in (8.9), we have

$$\begin{bmatrix} X_E \\ X_O \end{bmatrix} = \begin{bmatrix} W_4 & W_4 \\ W_4 D_8 & -W_4 D_8 \end{bmatrix} \begin{bmatrix} x_T \\ x_B \end{bmatrix}. \tag{8.15}$$

In general, for N even, this matrix equation can be written as

$$X_E = W_{\frac{N}{2}} \, v, \tag{8.16a}$$

$$X_O = W_{\frac{N}{2}} \, z, \tag{8.16b}$$

where v and z are $\frac{N}{2}$-point vectors defined by

$$v \triangleq x_T + x_B, \tag{8.17a}$$

$$z \triangleq D(x_T - x_B). \tag{8.17b}$$

These preprocessing operations require $\frac{N}{2}$ multiplications and N additions. If $N = 2^\nu$, we can obtain a recursive algorithm with computational complexity $O(N \log_2 N)$ operations. This divide-and-conquer FFT algorithm belongs to the decimation-in-frequency class and more details are provided in Tutorial Problem 3. It has been shown by Van Loan (1992) that most known FFT algorithms can be expressed as matrix factorizations of the DFT matrix.

8.3   Decimation-in-time FFT algorithms

In this section we provide an algebraic derivation of the decimation-in-time FFT algorithm. This approach, which is very easy to follow, is particularly useful for programming purposes. A similar algebraic derivation for the decimation-in-frequency FFT algorithm is given in Section 8.4.

8.3.1   Algebraic derivation

To develop the decimation-in-time FFT algorithm we split the N-point DFT summation, assuming that N is an even integer, into two $\frac{N}{2}$-point summations: one sum over the even-indexed points of $x[n]$ and another sum over the odd-indexed points of $x[n]$. Thus, after some simple algebraic manipulations, we obtain

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, 1, \dots, N-1 \\
&= \sum_{m=0}^{\frac{N}{2}-1} x[2m] \, W_N^{k(2m)} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] \, W_N^{k(2m+1)} \\
&= \sum_{m=0}^{\frac{N}{2}-1} x[2m] \, W_N^{k(2m)} + W_N^k \sum_{m=0}^{\frac{N}{2}-1} x[2m+1] \, W_N^{k(2m)}.
\end{aligned} \tag{8.18}
$$

For convenience, we define the following $\frac{N}{2}$-point sequences:

$$a[n] \triangleq x[2n], \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.19a}$$

$$b[n] \triangleq x[2n+1]. \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.19b}$$

Because the shorter sequences $a[n]$ and $b[n]$ are obtained by sampling (or "decimating") the sequence $x[n]$, this process is called *decimation-in-time* decomposition. Substituting these definitions into (8.18) and using the identity $W_N^2 = W_{\frac{N}{2}}$ yields

$$X[k] = A[k] + W_N^k B[k], \quad k = 0, 1, \ldots, N - 1 \tag{8.20}$$

where $A[k]$ and $B[k]$ are the following $\frac{N}{2}$-point DFTs:

$$A[k] \triangleq \sum_{m=0}^{\frac{N}{2}-1} a[m] \, W_{\frac{N}{2}}^{km}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.21a}$$

$$B[k] \triangleq \sum_{m=0}^{\frac{N}{2}-1} b[m] \, W_{\frac{N}{2}}^{km}. \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.21b}$$

An interesting physical interpretation of this decomposition might go as follows. The even-indexed sub-sequence $a[n]$ is obtained by sampling the original sequence $x[n]$. The odd-indexed sub-sequence $b[n]$ is obtained by sampling $x[n]$ after circular shifting by one sample. The DFT of $x[n]$ is just an average of the DFTs of $a[n]$ and $b[n]$, where the second DFT is multiplied by W_N^k to account for the circular shifting of $b[n]$ with respect to $a[n]$ (see Tutorial Problem 4).

A careful inspection of (8.21) reveals that we can compute $X[k]$ via (8.20) from $A[k]$ and $B[k]$ for $k = 0, 1, \ldots, \frac{N}{2} - 1$. However, to calculate $X[k]$ for $k \geq \frac{N}{2}$ we need values of the $\frac{N}{2}$-point DFTs $A[k]$ and $B[k]$ outside their range. We can obtain these values using the periodicity property of DFT. Indeed, since the $\frac{N}{2}$-point transforms $A[k]$ and $B[k]$ are implicitly periodic with period $\frac{N}{2}$, we have

$$X\left[k + \frac{N}{2}\right] = A\left[k + \frac{N}{2}\right] + W_N^{k+\frac{N}{2}} B\left[k + \frac{N}{2}\right]$$

$$= A[k] - W_N^k B[k], \tag{8.22}$$

since $W_N^{(k+\frac{N}{2})} = -W_N^k$. Thus, we can compute the N-point DFT $X[k]$ from the $\frac{N}{2}$-point DFTs $A[k]$ and $B[k]$ using the following merging formulas

$$X[k] = A[k] + W_N^k B[k], \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.23a}$$

$$X\left[k + \frac{N}{2}\right] = A[k] - W_N^k B[k]. \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.23b}$$

We emphasize that merging formulas (8.23) can be applied to any FFT of even length. Therefore, if $N = 2^v$ the "even-odd" decomposition of the input sequence can be applied recursively until we reach the point where the DFT lengths are equal to two. The computation of the 2-point DFT is trivial; it is the sum and difference of the two input samples. Indeed, using (8.23) for $N = 2$ yields

$$X[0] = x[0] + W_2^0 x[1] = x[0] + x[1], \tag{8.24a}$$

$$X[1] = x[0] + W_2^1 x[1] = x[0] - x[1]. \tag{8.24b}$$

Since the one-point DFT is the input point itself, we can think of (8.24) as merging two 1-point DFTs using (8.23). The decimation-in-time FFT was introduced by Cooley and Tukey (1965).

To illustrate the fundamental aspects of the decimation-in-time FFT algorithm we will develop in detail all steps for $N = 8$.

Example 8.1   Decimation-in-time FFT for $N = 8$

Suppose that we wish to compute the following 8-point DFT coefficients:

$$X[k] = \text{DFT}_8\{x[0], x[1], x[2], x[3], x[4], x[5], x[6], x[7]\}, \ 0 \le k \le 7 \tag{8.25}$$

using the divide-and-conquer approach described by (8.23). This requires the 4-point DFTs of the even-indexed and odd-indexed decimated-in-time sequences

$$A[k] = \text{DFT}_4\{x[0], x[2], x[4], x[6]\}, \quad 0 \le k \le 3 \tag{8.26a}$$

$$B[k] = \text{DFT}_4\{x[1], x[3], x[5], x[7]\}. \quad 0 \le k \le 3 \tag{8.26b}$$

To calculate $A[k]$ and $B[k]$ we need the following two-point transforms

$$C[k] = \text{DFT}_2\{x[0], x[4]\}, \quad k = 0, 1 \tag{8.27a}$$

$$D[k] = \text{DFT}_2\{x[2], x[6]\}, \quad k = 0, 1 \tag{8.27b}$$

$$E[k] = \text{DFT}_2\{x[1], x[5]\}, \quad k = 0, 1 \tag{8.27c}$$

$$F[k] = \text{DFT}_2\{x[3], x[7]\}. \quad k = 0, 1 \tag{8.27d}$$

The decomposition stops here because the two-point DFTs are easily computed by merging one-point DFTs as in (8.24). Therefore, the major computational effort is to merge $C[k]$ with $D[k]$, $E[k]$ with $F[k]$, and $A[k]$ with $B[k]$ using (8.23) for $N = 4$ and $N = 8$. We assume that the required twiddle factors have already been computed and stored. The formulas for merging four two-point DFTs are

$$A[k] = C[k] + W_8^{2k} D[k], \quad k = 0, 1 \tag{8.28a}$$

$$A[k + 2] = C[k] - W_8^{2k} D[k], \quad k = 0, 1 \tag{8.28b}$$
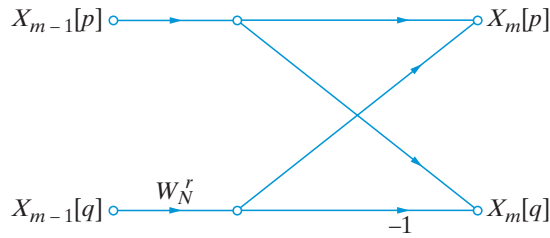
Figure 8.4 Flow graph of the butterfly operation for computation of the decimation-in-time FFT algorithm.

and

$$B[k] = E[k] + W_8^{2k}F[k], \qquad k = 0, 1 \tag{8.29a}$$

$$B[k + 2] = E[k] - W_8^{2k}F[k], \qquad k = 0, 1 \tag{8.29b}$$

respectively. Finally, we merge the four-point DFTs using

$$X[k] = A[k] + W_8^k B[k], \qquad k = 0, 1, \dots, 3 \tag{8.30a}$$

$$X[k + 4] = A[k] - W_8^k B[k]. \qquad k = 0, 1, \dots, 3 \tag{8.30b}$$

Note that we have used the identity $W_4^k = W_8^{2k}$ so that the twiddle factors for all merging equations correspond to $N = 8$.

A careful inspection of the merging formulas (8.28)–(8.30) shows that they all share the same form

$$X_m[p] = X_{m-1}[p] + W_N^r X_{m-1}[q], \tag{8.31a}$$

$$X_m[q] = X_{m-1}[p] - W_N^r X_{m-1}[q]. \tag{8.31b}$$

Figure 8.4 shows a flow graph of this operation, which is known as a *butterfly* computation because of its shape. The twiddle factors provide the adjustment required to double the size of the input DFT. The butterfly computation requires one complex addition, one complex subtraction, and one complex multiplication. ■

We note that the original problem of computing an N-point DFT has been replaced by the problem of computing N "trivial" one-point DFTs and then combining the results. The method takes place in two phases:

Shuffling The input sequence is successively decomposed into even and odd parts until we end-up with sub-sequences of length two. This reordering phase, known as *shuffling*, is shown in the left part of Figure 8.5.
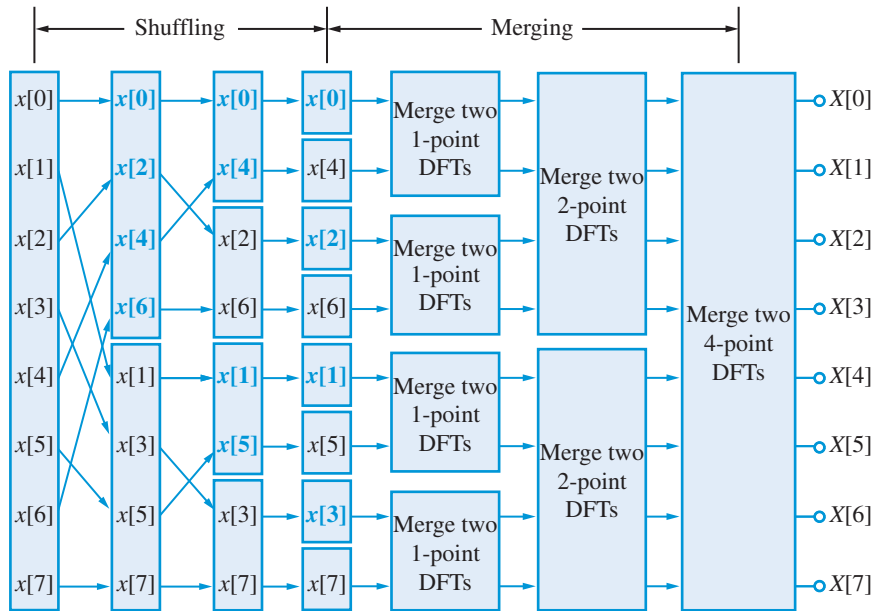
Figure 8.5 The shuffling and merging operations required for recursive computation of the 8-point DFT using the decimation-in-time FFT algorithm.

Merging The butterfly operations in (8.23) are used to combine DFTs of length 1 into DFTs of length 2, DFTs of length 2 into DFTs of length 4, and so on, until the final N-point DFT $X[k]$ is formed from two $\frac{N}{2}$-point DFTs. This combining phase, called *merging*, is shown in the right part of Figure 8.5.

Figure 8.6 shows a flow graph of the 8-point decimation-in-time FFT algorithm (merging phase only) using the butterfly operation in Figure 8.4.

8.3.2   Practical programming considerations

We noted that the decimation-in-time FFT algorithm has two sections: shuffling and merging. We next show that a systematic analysis of the diagrams in Figures 8.5 and 8.6 reveals some details and leads to some generalizations that are important for implementation of FFT algorithms in software or hardware.

Bit-reversed ordering The decimation-in-time FFT requires the input data to be re-ordered in a specific nonsequential order (see Figure 8.5). The shuffling process can be easily understood if we express the index of the original and shuffled sequences in binary form as illustrated in Figure 8.7(a). We note that the sample with binary index $b_2 b_1 b_0$ is moved to the memory location with binary index $b_0 b_1 b_2$. For example the sample $x[1] = x[(001)_2]$ is moved to memory location $(100)_2 = 4$. Figure 8.7(a) illustrates
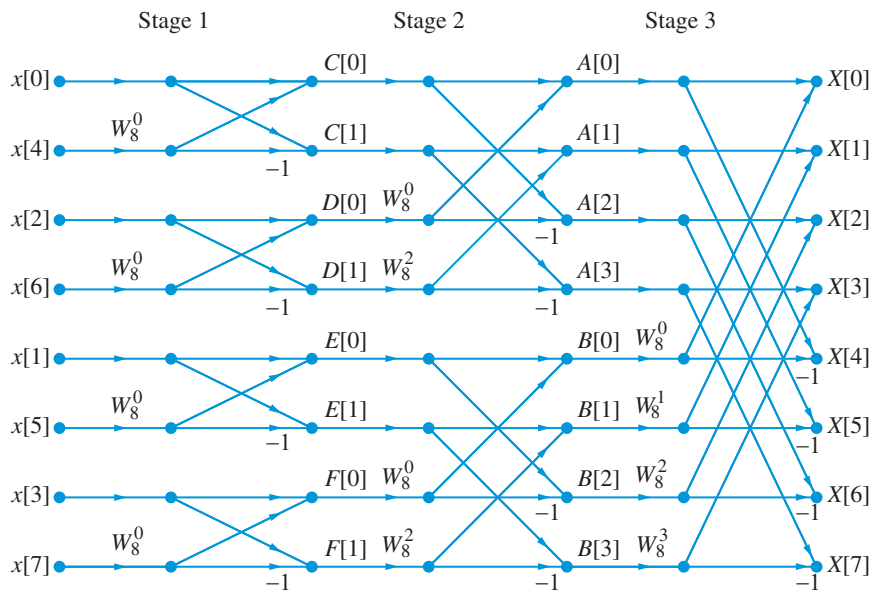
Figure 8.6 Flow graph of 8-point decimation-in-time FFT algorithm using the butterfly computation shown in Figure 8.4. The trivial twiddle factor $W_8^0 = 1$ is shown for the sake of generality.
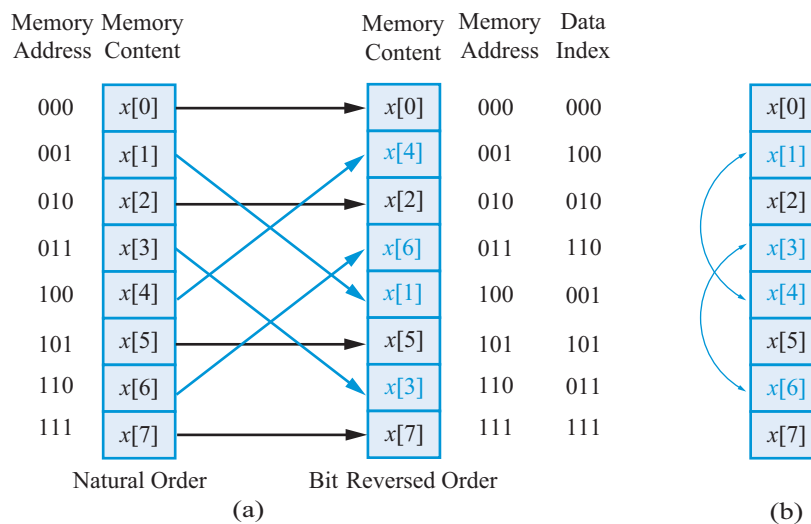
Figure 8.7 Shuffling a sequence with $N = 8$ samples by bit-reversal indexing: (a) shuffling using two arrays, and (b) in-place shuffling.

bit-reversal shuffling using two memory arrays. The first array contains the input sequence in *natural order* and the second in *bit-reversed order*. However, because this process involves only swapping pairs of elements (see Figure 8.7(b)) we do not need a second

array. Using an extra memory location, the MATLAB statements (using MATLAB indices) for swapping two samples are

```
temp = x(p);
x(p) = x(q);
x(q) = temp;
```

where `temp` is a temporary storage variable. Thus, we can reorder the input samples in the same array they originally occupied. An algorithm that uses the same memory locations to store both the input and output sequences is called an *in-place* algorithm. Note that samples with symmetric binary indices remain intact.

The primary goal of a bit-reversal algorithm is to obtain bit-reversed index `r` from the normal index `n` in a logical yet efficient manner. One logical approach, called a *naive* algorithm, computes indices sequentially starting with the index `n=0`. It is a nonrecursive procedure in that it does not take advantage of the previously computed index `r`. A more efficient procedure is a recursive one in which the successive `r` indices are computed from the previous one in a process called *reverse carry* algorithm. Figure 8.8 shows a MATLAB function, called `bitrev`, for bit-reversed reordering of a sequence $x[n]$ with length $N = 2^\nu$ samples that incorporates the reverse carry algorithm. It was first introduced in Gold and Rader (1969) and is explained in Kammler (2000).

The reverse carry is easy to understand and implement if we carefully study the mirror-image symmetry between the normal and reversed binary index columns in Figure 8.7(a). In the normal ordering we increment previous normal index `n` by one. Therefore in the reversed ordering, the mirror-image symmetry requires that we increment the reversed

```
function x=bitrev(x)
% Bit reversal algorithm based on Gold and Rader (1969)
N=length(x); r=0;
for n=0:N-2;
    if n<r % swap samples only for the first half of array
        temp=x(n+1);
        x(n+1)=x(r+1);
        x(r+1)=temp;
    end
    k=N/2; % even n: adds to the previous r;
           %  odd n: subtract from the previous r
    while k <= r
        r=r-k; % keep subtracting reverse carry
        k=k/2; % generate next reverse carry
    end
    r=r+k; % even n: add N/2; odd n: add the last carry.
end
```

Figure 8.8 Function for bit-reversed shuffling of an $N = 2^\nu$- point sequence.

index r by one from the *left hand side*, that is, propagate a reverse-carry. There are two possibilities in this recursive operation:

- If the index n is *even* then the last bit in n is 0 or the first bit in r is 0 and thus incrementing the reverse index by one amounts to adding $N/2$ to r (since no carry is generated) to obtain the next reverse index, that is $r=r+N/2$, as can be seen from Figure 8.7(a).
- If the index n is *odd* then adding one to the left-most bit of r generates a carry that needs to be propagated to the right. This amounts to successively subtracting $\frac{N}{2}$, $\frac{N}{4}$, ... from r until no carry is generated. This can be implemented as $r=r-k$ where $k=k/2$ starting with $k=N/2$ until $r<k$. Finally the last carry is added, $r=r+k$, to generate the next reverse index r.

The naive algorithm requires $O(3N\log_2 N)$ integer operations while the reverse carry algorithm requires $O(5N)$ integer operations (Kammler (2000)).

Successive DFT merging The flow graph in Figure 8.6 provides a clear description of the decimation-in-time FFT algorithm. Indeed, careful inspection of the flow graph shows the following important elements:

Stages The flow graph consists of $v = \log_2 N = \log_2 8 = 3$ stages. Each stage takes a set of N complex numbers $X_{m-1}[k]$ and transforms them to another set of N complex numbers $X_m[k]$ using butterfly operations of the form

$$X_m[p]=X_{m-1}[p] + W_N^r X_{m-1}[q], \tag{8.32a}$$

$$X_m[q]=X_{m-1}[p] - W_N^r X_{m-1}[q]. \tag{8.32b}$$

The array $X_0[k]$ contains the input samples in bit-reversed order and the array $X_v[k]$ contains the desired DFT coefficients $X[k]$. The stages are computed sequentially from left to right using a loop specified by the index m, which takes the values $m = 1, 2, \ldots, v$. We note that once a butterfly is evaluated its input values are no longer needed. Thus, we can conserve memory by saving the results of the butterfly computation in the same memory locations where the input data were stored (in-place algorithm). The MATLAB statements (using MATLAB indices) for this butterfly computation are

```
temp = W(r)*X(ib);
X(it) = X(it) + temp;
X(ib) = X(it) - temp;
```

where `temp` is a temporary storage variable. We use the index `it` for the top input of a butterfly (index top) and the index `ib` for the bottom input of a butterfly (index bottom). Thus, we can compute an entire stage in-place using only one additional memory location.

Butterflies and twiddle factors Each stage requires $\frac{N}{2} = 2^{v-1} = 4$ butterflies (for $N = 8$) and each butterfly requires the value of a twiddle factor W_N^r. The algorithm requires the twiddle factors, W_N^r, $r = 0, 1, \ldots, \frac{N}{2} - 1$. These coefficients are usually stored in a look-up table; otherwise, we must compute the values as needed (see Tutorial Problem 5).

We note that, at each stage, we can compute the butterflies working from top to bottom or vice versa. Computation of each butterfly requires accessing the proper pair of points from

memory and evaluating the proper twiddle factor. A disadvantage of this approach is that the twiddle factors may have to be computed and recomputed several times at each stage.

A more efficient approach is to compute a particular value of W_N^r at a given stage and then evaluate all butterflies that use this twiddle factor. For example, in Figure 8.6 at the second stage we should first evaluate the two butterflies that require W_8^0 and then evaluate the two butterflies that require W_8^2. This requires two additional loops within the loop for the stages.

We first note that, although each stage has $\frac{N}{2}$ twiddle factors, only 2^{m-1} of them have *different* values. Careful analysis of the pattern in Figure 8.6 shows that the required twiddle factors are given by

$$W_N^{(N/2^m)r} = \exp\left(-j2\pi r/2^m\right). \quad r = 0, 1, \ldots, 2^{m-1} - 1 \tag{8.33}$$

We use a loop with MATLAB index `ir` to evaluate the twiddle factors in (8.33). We need a second loop, within the twiddle factor loop, to compute the butterflies for each twiddle factor. This requires us (a) to specify each butterfly of the stage, and (b) to determine the "right" input values. A simple inspection of Figure 8.6 shows that the inputs to any butterfly in the second stage are separated by two memory locations, the two butterflies using W_8^0 are separated by four memory locations, and the two butterflies using W_8^2 are separated by four memory locations. Since, in general, butterflies using the same twiddle factor are separated by 2^m memory locations, we use an index `it`, with step 2^m, to access the top input of each butterfly. Similarly, since the inputs to each butterfly of the mth stage are separated by 2^{m-1} memory locations, we determine the index `ib` of the bottom input by increasing the index `it` of the top input by 2^{m-1}.

A simple FFT function   The result of this algorithmic approach and the associated indexing schemes is the three-loop MATLAB function, `fftditr2`, shown in Figure 8.9. The first loop iterates through the stages, the second loop scans the different twiddle factors of each stage, and the third loop computes the butterflies sharing the same twiddle factor. This MATLAB function is based on an elegant FORTRAN subroutine originally written by Cooley *et al.* (1969). The MATLAB function in Figure 8.9 is provided to illustrate the implementation of FFT algorithms; however, a C or FORTRAN version provides good performance for practical use. For maximum efficiency we can avoid the computation of twiddle factors using a look-up table (see Problem 45).

8.3.3   Alternative forms

Careful inspection of the flow graph in Figure 8.6 shows that we can rearrange the order of the nodes, without changing the resulting DFT coefficients, as long as we do not break them or change their coefficients. For example, as shown in Figure 8.10, we can rearrange the nodes in the flow graph of Figure 8.6 so that both the input and output are in natural order. Although this rearrangement changes the order in which the data are stored and retrieved from memory, the results of computation remain the same.

As we have already discussed, if the input and output nodes of each butterfly are horizontally adjacent, the computation can be done in-place using only one array of N complex memory locations. Since this requirement is violated in Figure 8.10 the resulting algorithm

```
function x=fftditr2(x)
% DIT Radix-2 FFT Algorithm
N=length(x); nu=log2(N);
x = bitrevorder(x);
for m=1:nu;
    L=2^m;
    L2=L/2;
    for ir=1:L2;
        W=exp(-1i*2*pi*(ir-1)/L);
        for it=ir:L:N;
            ib=it+L2;
            temp=x(ib)*W;
            x(it)=x(it)+temp;
            x(ib)=x(it)-temp;
        end
    end
end
```

Figure 8.9 MATLAB function for the decimation-in-time FFT algorithm.
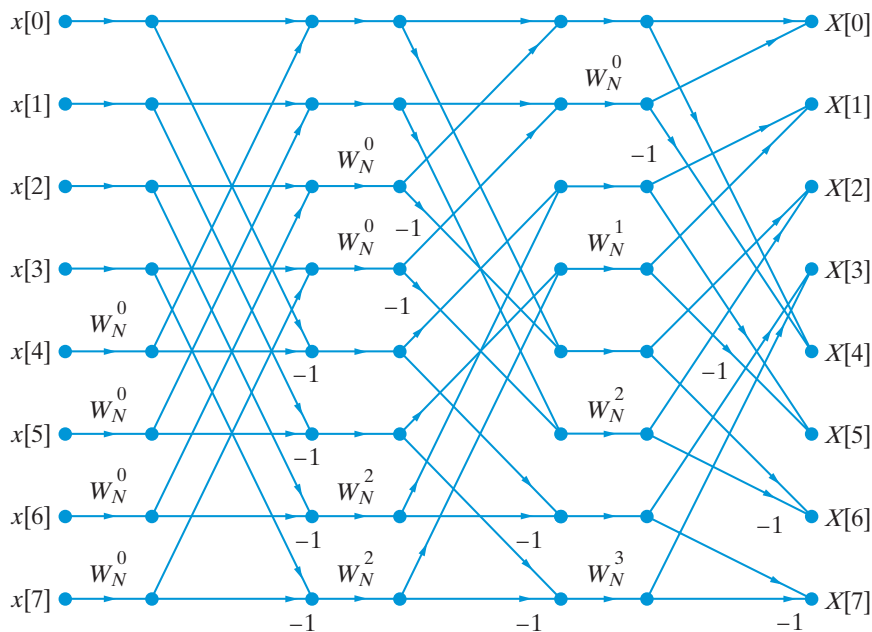
Figure 8.10 Decimation-in-time FFT algorithm with both input and output in natural order.
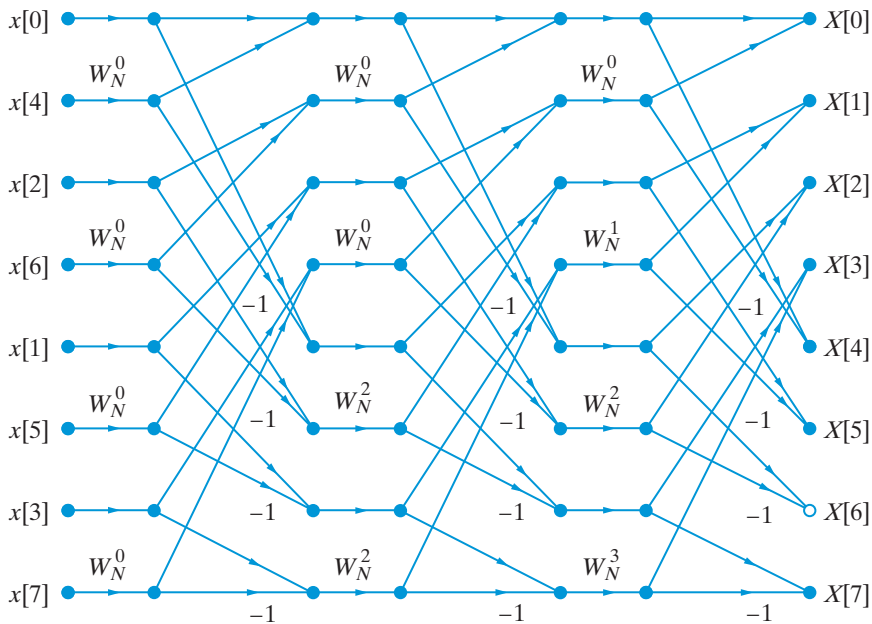
Figure 8.11 Decimation-in-time FFT algorithm where each stage has identical geometry.

cannot be computed in-place (see Problem 26). Furthermore, the indexing operations are so complicated that the use of these algorithms does not offer any practical advantage. We emphasize that bit-reversed shuffling of the input or output data is necessary for in-place computation.

Figure 8.11 shows the flow graph of a decimation-in-time FFT algorithm, where each stage has *identical* geometry. This algorithm, which was proposed by Singleton (1969), is attractive for hardware implementation. However, the software implementation is inconvenient because the algorithm is not in-place and requires bit-reversal.

8.4   Decimation-in-frequency FFT algorithms

The decimation-in-time FFT algorithm is based on recursive decomposition of the input sequence into even-indexed and odd-indexed subsequences. Another algorithm, with the same computational complexity, can be derived by applying the same divide-and-conquer decomposition on the DFT coefficients.

As before the starting point is with the definition of the N-point DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, 1, \ldots, N-1 \tag{8.34}$$

where we assume that $N = 2^\nu$. The even-indexed DFT coefficients are given by

$$X[2k] = \sum_{n=0}^{N-1} x[n]W_N^{(2k)n} = \sum_{n=0}^{N-1} x[n]W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \qquad (8.35)$$

where we have used the identity $W_N^2 = W_{\frac{N}{2}}$. Even if this looks like an $\frac{N}{2}$-point DFT, it really is not because the range of summation is from $n = 0$ to $n = N - 1$. To bypass this problem we split the summation into two parts as follows:

$$X[2k] = \sum_{n=0}^{\frac{N}{2}-1} x[n]W_{\frac{N}{2}}^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left[n + \frac{N}{2}\right] W_{\frac{N}{2}}^{k\left(n+\frac{N}{2}\right)}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]W_{\frac{N}{2}}^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left[n + \frac{N}{2}\right] W_{\frac{N}{2}}^{kn}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + x\left[n + \frac{N}{2}\right]\right) W_{\frac{N}{2}}^{kn}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \qquad (8.36)$$

Following a similar procedure for the odd-indexed DFT coefficients, we obtain

$$X[2k + 1] = \sum_{n=0}^{\frac{N}{2}-1} \left[\left(x[n] - x\left[n + \frac{N}{2}\right]\right) W_N^n\right] W_{\frac{N}{2}}^{kn}, \qquad (8.37)$$

for $k = 0, 1, \ldots, \frac{N}{2} - 1$. Equations (8.36) and (8.37) express the original N-point DFT in terms of two $\frac{N}{2}$-point DFTs: FFT algorithms based on the decomposition of the DFT sequence $X[k]$ into even-indexed and odd-indexed subsequences are called *decimation-in-frequency* algorithms.

To develop a divide-and-conquer algorithm we define the $\frac{N}{2}$-point sequences

$$a[n] \triangleq x[n] + x\left[n + \frac{N}{2}\right], \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \qquad (8.38a)$$

$$b[n] \triangleq \left(x[n] - x\left[n + \frac{N}{2}\right]\right) W_N^n, \qquad n = 0, 1, \ldots, \frac{N}{2} - 1 \qquad (8.38b)$$

and compute their $\frac{N}{2}$-point DFTs

$$A[k] = \sum_{n=0}^{\frac{N}{2}-1} a[n]W_{\frac{N}{2}}^{kn}, \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \qquad (8.39a)$$

$$B[k] = \sum_{n=0}^{\frac{N}{2}-1} b[k]W_{\frac{N}{2}}^{kn}. \qquad k = 0, 1, \ldots, \frac{N}{2} - 1 \qquad (8.39b)$$

To find the N-point DFT $X[k]$ of the original sequence $x[n]$, we interleave the values of the two $\frac{N}{2}$-point DFTs as follows:

$$X[2k] = A[k], \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.40a}$$

$$X[2k + 1] = B[k]. \quad k = 0, 1, \ldots, \frac{N}{2} - 1 \tag{8.40b}$$

Thus, the task of computing the N-point DFT has been replaced by the task of computing two $\frac{N}{2}$-point DFTs. As was the case with the decimation-in-time, we do not stop here. The same strategy is applied to compute the DFTs of $a[n]$ and $b[n]$, then again to compute the resulting $\frac{N}{4}$-point DFTs, until we eventually reach one-point DFTs, which are trivial to compute. Therefore, the only work involves preparing the input sequences for each successive DFT by applying relations (8.38) with the appropriate value of N. The decimation-in-frequency FFT was found independently by Sande and by Cooley and Stockham; see Cochran *et al.* (1967).

To understand the decimation-in-frequency FFT algorithm and compare it with the decimation-in-time approach, we provide a detailed derivation for $N = 8$.

Example 8.2   Decimation-in-frequency FFT for $N = 8$

Suppose that we wish to compute the 8-point DFT $X[k]$ of the sequence $x[0], x[1], \ldots, x[7]$ using the decimation-in-frequency approach. The first step is to compute the two 4-point sequences $a[n]$ and $b[n]$ for $n = 0, 1, 2, 3$

$$a[n] = x[n] + x[n + 4], \tag{8.41a}$$

$$b[n] = (x[n] - x[n + 4]) W_8^n. \tag{8.41b}$$

We denote by $A[k]$ and $B[k]$ the DFTs of $a[n]$ and $b[n]$, respectively. The 4-point DFTs $A[k]$ and $B[k]$ are equal to the even-indexed and odd-indexed samples of the original 8-point DFT $X[k]$. Thus, we have

$$
\begin{aligned}
A[0] &= X[0] & B[0] &= X[1] \\
A[1] &= X[2] & B[1] &= X[3] \\
A[2] &= X[4] & B[2] &= X[5] \\
A[3] &= X[6] & B[3] &= X[7].
\end{aligned} \tag{8.42}
$$

To compute the two 4-point DFTs, $A[k]$ and $B[k]$, we repeat (8.41) with $N = 4$. We start by forming the following two-point sequences for $n = 0, 1$

$$c[n] = a[n] + a[n + 2], \tag{8.43a}$$

$$d[n] = (a[n] - a[n + 2]) W_8^{2n}, \tag{8.43b}$$

$$e[n] = b[n] + b[n + 2], \tag{8.43c}$$

$$f[n] = (b[n] - b[n + 2]) W_8^{2n}. \tag{8.43d}$$
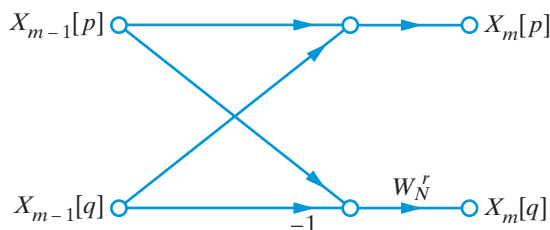
Figure 8.12 Flow graph for the decimation-in-frequency butterfly.

Note that we have used the identity $W_4^n = W_8^{2n}$ so that all twiddle factors correspond to $N = 8$. The DFTs $C[k], D[k], E[k]$, and $F[k]$ of the two-point sequences in (8.43) are easily determined using (8.39). Thus, using (8.39) and (8.40) we have

$$C[0] = c[0] + c[1] = A[0] = X[0]$$
$$C[1] = c[0] - c[1] = A[2] = X[4]$$
$$D[0] = d[0] + d[1] = A[1] = X[2]$$
$$D[1] = d[0] - d[1] = A[3] = X[6]$$
$$E[0] = e[0] + e[1] = B[0] = X[1]$$
$$E[1] = e[0] - e[1] = B[2] = X[5]$$
$$F[0] = f[0] + f[1] = B[1] = X[3]$$
$$F[1] = f[0] - f[1] = B[3] = X[7]. \tag{8.44}$$

We note that the computation of the 8-point DFT $X[k]$ has been reduced to computation of the sub-sequences $a[n], b[n], \ldots, f[n]$, and the two-point DFTs in (8.44). If we recall that $W_8^0 = 1$, all these computations can be done using the following butterfly operation (see flow graph in Figure 8.12):

$$X_m[p] = X_{m-1}[p] + X_{m-1}[q], \tag{8.45a}$$
$$X_m[q] = (X_{m-1}[p] - X_{m-1}[q])W_N^r. \tag{8.45b}$$

The procedure described by (8.41)–(8.44) can be represented by the flow graph shown in Figure 8.13. ∎

A careful examination of the flow graph in Figure 8.13 gives us enough information to implement the decimation-in-frequency FFT algorithm and to determine its computational cost. The decimation-in-frequency flow graph has $v = \log_2 N = 3$ stages, each stage has $\frac{N}{2} = 4$ butterflies, and each butterfly requires one complex multiplication and two complex additions. Thus, the algorithm requires $\frac{N}{2}\log_2 N$ complex multiplications and $N\log_2 N$ complex additions, similar to the decimation-in-time FFT. The decimation-in-frequency algorithm can also be done in place; however, unlike the decimation-in-time FFT algorithm, the merge part occurs first and the reordering part occurs last. The development of a MATLAB function for the decimation-in-frequency FFT algorithm is discussed in Tutorial Problem 8.
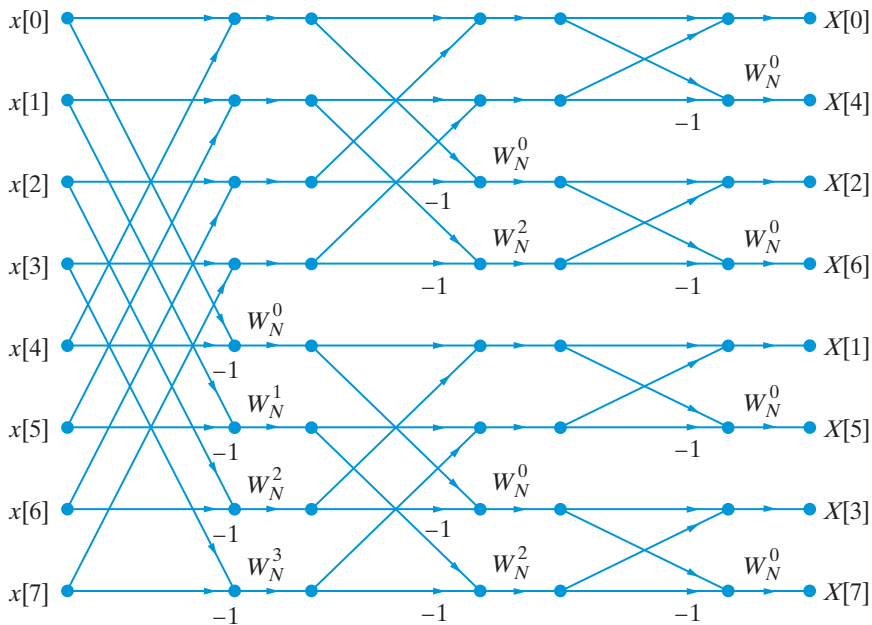
Figure 8.13 Flow graph for the decimation-in-frequency 8-point FFT algorithm. The input sequence is in natural order and the output sequence in bit-reversed order.

Alternative forms of the decimation-in-frequency FFT algorithm can be obtained by rearranging the flow graph in Figure 8.13 as we discussed in Section 8.3.3. However, the use of these algorithms does not offer any particular advantage. From a theoretical perspective it is worthwhile to note that the decimation-in-frequency flow graph in Figure 8.13 can be obtained from the decimation-in-time flow graph in Figure 8.6 by interchanging the input and output and reversing the direction of the arrows (see Problem 39). This is a special case of the more general transposition theorem introduced by Claasen and Mecklenbräuker (1978).

8.5   Generalizations and additional FFT algorithms

We start this section by introducing a general approach that can be used to derive both the decimation-in-time and decimation-in-frequency FFTs as special cases. We note that, if the length N of the sequence $x[n]$ is a composite integer, say $N = N_1 N_2$, we can divide the N-point sequence $x[n]$ and its DFT $X[k]$ into N_1 sub-sequences of length N_2 (or N_2 sub-sequences of length N_1) by expressing the index n as follows:

$$n = N_2 n_1 + n_2, \quad n_1 = 0, 1, \ldots, N_1 - 1, \quad n_2 = 0, 1, \ldots, N_2 - 1 \qquad (8.46a)$$

$$k = k_1 + N_1 k_2. \quad k_1 = 0, 1, \ldots, N_1 - 1, \quad k_2 = 0, 1, \ldots, N_2 - 1 \qquad (8.46b)$$

The product nk in the exponent of W_N can be written as follows

$$nk = (N_2 n_1 + n_2)(k_1 + N_1 k_2)$$
$$= N n_1 k_2 + N_1 n_2 k_2 + N_2 n_1 k_1 + n_2 k_1. \tag{8.47}$$

Using (7.24) and its properties, we have

$$W_N^N = 1, \quad W_N^{N_1} = W_{N_2}, \quad W_N^{N_2} = W_{N_1}. \tag{8.48}$$

Substituting the representations of n and k (8.46) into the definition (8.1) yields

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_N^{(N_2 n_1 + n_2)(k_1 + N_1 k_2)}. \tag{8.49}$$

Substituting (8.47) and (8.48) into (8.49), we obtain

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} \left[\left(\sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{k_1 n_1} \right) W_N^{k_1 n_2} \right] W_{N_2}^{k_2 n_2}. \tag{8.50}$$

We next define N_2 sequences of length N_1 as follows:

$$x_{n_2}[n_1] \triangleq x[N_2 n_1 + n_2]. \tag{8.51}$$

We note that each of these sequences is obtained by sampling (decimating) a shifted copy of the original sequence every N_2 samples. The DFT of $x_{n_2}[n_1]$ is

$$X_{n_2}[k_1] = \sum_{n_1=0}^{N_1-1} x_{n_2}[n_1] W_{N_1}^{k_1 n_1}. \tag{8.52}$$

Substituting (8.52) into (8.50) we have

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} W_N^{k_1 n_2} X_{n_2}[k_1] W_{N_2}^{k_2 n_2}. \tag{8.53}$$

Finally, if we define N_1 sequences of length N_2 by

$$x_{k_1}[n_2] \triangleq W_N^{k_1 n_2} X_{n_2}[k_1], \tag{8.54}$$

we have

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} x_{k_1}[n_2] W_{N_2}^{k_2 n_2}. \tag{8.55}$$

This is the basic decomposition step of the original FFT algorithm developed by Cooley and Tukey (1965). This procedure consists of the following steps:

1. Form the N_2 decimated sequences, defined by (8.51), and compute the N_1-point DFT of each sequence.
2. Multiply each N_1-point DFT by the twiddle factor $W_N^{k_1 n_2}$, as shown in (8.54).
3. Compute the N_2-point DFTs of the N_1 sequences determined from step 2.

If the smaller size DFTs are evaluated with the direct method, we can see that this decomposition requires $N(N_1 + N_2)$ complex operations compared to N^2. If $N_1 + N_2 \ll N$, we have a significant reduction in the number of operations. The process can be continued if N_1 and N_2 are composite numbers. An example of the Cooley–Tukey decomposition for $N = 15$ is given in Tutorial Problem 12. If $N_1 = \frac{N}{2}$ and $N_2 = 2$, we have the first stage of the decimation-in-time FFT. The choice $N_1 = 2$ and $N_2 = \frac{N}{2}$ yields the first stage of the decimation-in-frequency FFT. These ideas also provide the basis for the computation of two-dimensional DFTs or parallel FFT algorithms, see Problem 41.

The basic Cooley–Tukey algorithm assumes that N is a power of 2 and hence is called a *radix-2* FFT algorithm. However, the same approach can be applied when N is a power of R. The resulting algorithms are called radix-R FFTs. In general, if N can be factored as $N = R_1 R_2 \dots R_M$, we have a *mixed radix* FFT. The radix-4 FFT has a butterfly without multiplications and half as many stages as the radix-2 FFT; as a result, the number of multiplications is reduced by half (see Problem 13). The improvement diminishes as we increase the radix from 4 to 8 or from 8 to 16.

Another algorithm called the *split-radix FFT*, which was introduced by Yavne (1968) and Duhamel (1986), also assumes that $N = 2^\nu$, but it uses a radix-2 decimation-in-frequency algorithm to compute the even-indexed samples and a radix-4 decimation-in-frequency algorithm to compute the odd-indexed samples of the DFT (see Tutorial Problem 47). The resulting algorithm reduces the number of floating-point operations to about $4N\log_2 N$ compared to $5N\log_2 N$ for the radix-2 FFT. The modified split-radix FFT, introduced by Johnson and Frigo (2007), lowers the floating point operations by five percent to about $(34/9)N\log_2 N$. A computer program for the split-radix FFT algorithm is provided in Sorensen *et al.* (1986).

The twiddle factors in (8.54) can be eliminated by choosing appropriate index mapping. This is possible if the factors in $N = R_1 R_2 \dots R_M$ are relatively prime. The result is a family of more efficient algorithms known as *Prime-Factor Algorithms (PFAs)*. These algorithms are discussed by Burrus and Parks (1985) and Blahut (2010). The *Winograd Fourier Transform Algorithm (WFTA)*, see Winograd (1978), minimizes the number of multiplications at the expense of an increased number of additions. The WFTA achieves its efficiency by expressing the DFT in terms of convolution, or equivalently, polynomial multiplication. However, the significant increase in the number of additions and the complexity of indexing operations limit its practical value.

8.6   Practical considerations

With the availability of a large number of different FFT algorithms, the inevitable question is: which is the best FFT algorithm? The answer to this question, which should not come

as a surprise, is "it depends." To put this answer into perspective, we briefly discuss the main factors that determine the performance of FFT algorithms.

Twiddle factors There are three approaches to obtain the values of W_N^r required by a FFT algorithm. The slowest approach is to compute each value when it is needed. Another way is, at each stage, to compute one value and use a recursion to determine the remaining values. The fastest approach is to pre-calculate and store the twiddle factors in a look-up table, from where they can be fetched as needed. This approach is almost always used in practice.

Butterfly computations The additions and multiplications required by a butterfly used to be the slowest computer operations. However, current computer systems have arithmetic speeds that make indexing and data transfer operations equally important factors. As a result, code optimization should not exclusively focus on minimizing the number of numerical operations. Since many computers support single cycle multiply–accumulate operations, we should take this into account when coding FFT algorithms.

Indexing Determining indices for data access and loop control may consume a significant amount of computing cycles. The indexing complexity of some algorithms may outweigh the savings obtained from a reduction in the number of butterfly computations.

Memory management The efficiency of fetching and storing data during the butterfly computations depends on the memory hierarchy structure. Current general-purpose computers use a hierarchy of storage devices with increasing size and decreasing speed (central processing unit registers, cache memory, main memory, and external storage). Inefficient use of this memory structure may result in a significant slow down of a FFT algorithm.

Bit-reversed shuffling Bit-reversed reordering of the input or output data consume several percent of the total run-time of an FFT program. Several fast bit-reversal algorithms are described in Karp (1996) and Evans (1987). However, the simple algorithm in Figure 8.8 will continue to be used because it is simple and can be done in place; see the discussion in Rodriguez (1989).

The main conclusion from this cursory discussion is that when we look for the best FFT or we wish to optimize an FFT for speed, we should keep in perspective the factors that determine performance. First, the critical speed-up from $O(N^2)$ to $O(N\log_2 N)$ can be achieved by using any of the existing FFT algorithms. Additional significant savings result from the use of good programming techniques. These include the use of good compilers, assembly language programming, and hand-coding of critical parts of the algorithm. A well-written radix-2 FFT program, which may run faster than an inefficiently coded split-radix FFT, will be sufficient for many typical signal processing applications. For more demanding applications it is recommended that a highly optimized professional code be used; see for example, Frigo and Johnson (2005) and Press *et al.* (2007).

The speed of FFT algorithms on current computer hardware is determined by many factors besides the number of arithmetic operations. When we program a FFT algorithm in a high-level programming language, we use a compiler to translate the code into machine-level instructions which are executed on the target processor. In general, this approach will result in implementations whose performance depends on the structure of FFT algorithm, the target computer architecture, and the efficiency of the compiler. Efficient implementation of FFT algorithms on current computer architectures requires consideration of factors like parallelization, vectorization, and memory hierarchy architecture. Best performance

is obtained when the structure of the FFT algorithm matches the architecture of the target machine.

A review of optimizations necessary for good performance on multicore computers is given in Franchetti *et al.* (2009). This approach uses a matrix framework to manipulate the structure of a FFT algorithm to obtain variants that can be efficiently mapped to multicore systems.

Another approach is used by the FFTW ("Fastest Fourier Transform in the West") free-software library developed by Frigo and Johnson (1998, 2005). The FFTW searches over the possible factorizations of length *N* and empirically determines the one with the best performance for the target architecture. Using the FFTW library involves two stages: planning and execution. The user first invokes the FFTW planner to find the best decomposition into smaller transforms. Then, the user executes the code obtained as many times as desired for DFTs of the same length *N*. The FFTW library is used in the implementation of FFT in the latest releases of MATLAB.

Finally, we note that DSP processors have architectures and instruction sets optimized for digital filtering and FFT applications. For example, most DSP processors provide built-in hardware instructions for modulo and bit-reversed addressing. Thus, when we develop code for DSP processors, we should exploit these instructions to accelerate the execution of FFT algorithms. A discussion of these issues by Meyer and Schwarz (1990) is still relevant for current processors.

Computation using MATLAB's native functions The DFT and IDFT are efficiently computed in MATLAB using functions `fft` and `ifft`, respectively. The `X = fft(x)` function computes the `length(x)` DFT of vector `x` in `X`. Similarly, `x = ifft(x)` computes the `length(X)` IDFT of vector `X` in `x`. To compute the DFT of a specific length *N*, the `X = fft(x,N)` invocation is used in which the vector `x` is padded with zeros to length `N` if it is larger than the `length(x)`, otherwise `x` is truncated to the first `N` samples. The `x = ifft(X,N)` is also used in a similar manner.

MATLAB also provides two additional functions, `fftshift` and `ifftshift`, which are useful in plotting FFT/IFFT results. The `fftshift` function shifts the zero-frequency (or DC) component to the center of the spectrum while the `ifftshift` function undoes the results of `fftshift`. If `X` contains an odd number of elements, `ifftshift(fftshift(X))` must be used to obtain the original `X`.

The basic `fft` and `ifft` functions are based on the FFTW library as described above. When *N* is a composite number, the FFTW library planner determines the best decomposition of the DFT using a combination of the Cooley–Tukey, a prime factor algorithm, and/or a split-radix algorithm through one of several processor-generated fixed-size smaller transform code-fragments or "codelets." When *N* is a prime number, the FFTW library planner first decomposes an *N*-point DFT into three $(N-1)$-point DFTs using the algorithm in Rader (1968). It then uses the above decomposition to compute the $(N-1)$-point DFTs. Therefore MATLAB's FFT implementation is very efficient.

The execution time for `fft` depends on the length *N* of the transform. It is fastest for lengths equal to powers-of-two and almost as fast for lengths that have only small prime factors. However, it is several times slower for lengths that are prime or which have large prime factors. In the recent versions of MATLAB (version 6 and later), it is not easy to determine execution times in a meaningful fashion due to discontinued use of the `flops`
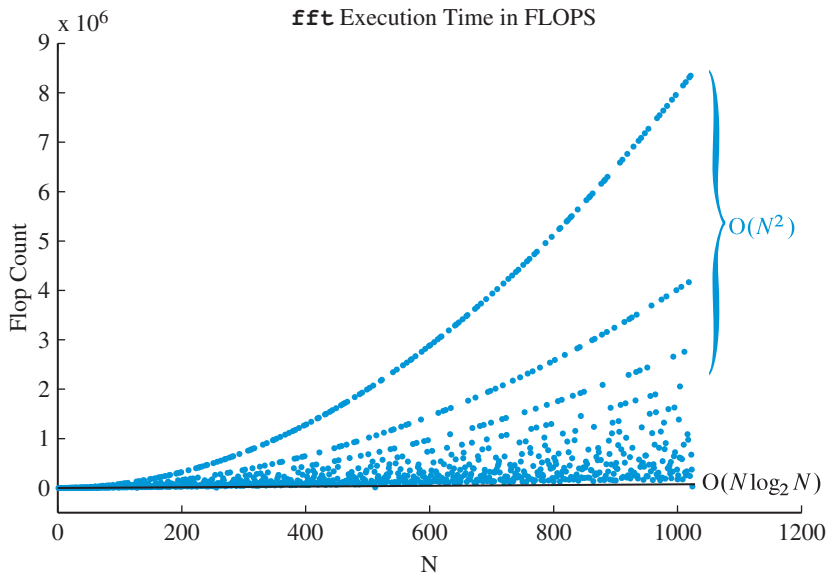
Figure 8.14 The floating-point operations (flops) count as a function of N in the **fft** function using MATLAB Version-5

function that computed number of floating-point operations in an algorithm execution and due to the use of the FFTW library. Figure 8.14 shows the plot of execution time in terms of floating-point operations as a function of N from 1 through 1024. It was generated using MATLAB version-5 which does not use the FFTW library. It shows several different trends in the number of flops. When N is a prime number or contains large prime factors, the flops count follows several of the $O(N^2)$ trends. When N is a power-of-two length or contains small prime factors, the flops counts are $O(N\log_2 N)$ which is almost a linear trend. For lengths that fall in between these two extremes, the flops counts show mixed trends.

8.7   Computation of DFT for special applications

The FFT algorithms presented in the previous sections compute *all* DFT values in an efficient manner given that *all* signal samples are available. There are applications in which only a few DFT values may be needed or where signal samples may be streaming in and not available at the same time. One may also wish to compute *z*-transform values somewhere in the *z*-plane that are not necessarily DFT values, or we may want to zoom-in on to a small segment of the unit circle. Algorithms presented in this section are designed for such applications by computing the DFT as a linear convolution or filtering operation. We also discuss the quick Fourier transform that only exploits symmetries in the definition and can be used for any length sequences. Finally, a time-recursive approach for DFT calculation, called sliding DFT, is presented.

Goertzel's algorithm

This approach for computation of the DFT uses the periodicity property of the W_N^{-kn} to reduce computations. The result is a recursive algorithm that is more efficient for computing some DFT values compared with FFT algorithms. Using the result

$$W_N^{-kN} = 1, \tag{8.56}$$

we can express (8.1) as

$$X[k] = W_N^{-kN} \sum_{n=0}^{N-1} x[n] W_N^{kn} = \sum_{n=0}^{N-1} x[n] W_N^{-k(N-n)}, \tag{8.57}$$

which appears to be a linear convolution sum. However (8.57) is also a polynomial in W_N^{-k} which can be efficiently computed as nested evaluations using the Horner's rule (Van Loan (2000)). To understand this computation let $N = 4$. Then (8.57) becomes

$$X[k] = \sum_{n=0}^{3} x[n] W_4^{-k(4-n)} = x[3] W_4^{-k} + x[2] W_4^{-2k} + x[1] W_4^{-3k} + x[0] W_4^{-4k}$$

$$= W_4^{-k} \left\{ x[3] + W_4^{-k} \left\{ x[2] + W_4^{-k} \left\{ x[1] + W_4^{-k} x[0] \right\} \right\} \right\}. \tag{8.58}$$

If we make the assignments

$$y_k[-1] = 0, \qquad\qquad y_k[0] = x[0] + W_4^{-k} y_k[-1],$$
$$y_k[1] = x[1] + W_4^{-k} y_k[0], \qquad y_k[2] = x[2] + W_4^{-k} y_k[1],$$
$$y_k[3] = x[3] + W_4^{-k} y_k[2], \qquad y_k[4] = x[4] + W_4^{-k} y_k[3] = W_4^{-k} y_k[3],$$

then these assignments suggest a recursive approach for computation of $X[k]$:

$$y_k[n] = W_N^{-k} y_k[n-1] + x[n], \quad 0 \le n \le N \tag{8.59a}$$

$$X[k] = y_k[N], \tag{8.59b}$$

with initial condition $y_k[-1] = 0$ and input $x[n] = 0$ for $n < 0$ and $n \ge N$. Thus to compute one DFT value we will need O(N) complex operations and to compute all DFT values we will need O(N^2) complex operations which is the same as that for the DFT. However, there are some advantages: (a) if only few $M < \log N$ DFT values are needed then we need only O(MN) < O($N\log N$) operations; (b) we do not have to compute or store $\{W_N^{-kn}\}$ complex values since these are computed recursively in the algorithm; and (c) the computations can start as soon as the first input sample is available, that is, we do not have to wait until all samples are available.

The computational efficiency of the algorithm in (8.59) can be improved further by converting it into a second-order recursive algorithm with real coefficients. Note that the system function of the recursive filter in (8.59a) can be written as

```
function X = gafft(x,N,k)
% Goertzel's algorithm
% X = gafft(x,N,k)
% Computes k-th sample of an N-point DFT X[k] of x[n]
% using Goertzel Algorithm
L = length(x); x = [reshape(x,1,L),zeros(1,N-L+1)];
K = length(k); X = zeros(1,K);
for i = 1:K
    v = filter(1,[1,-2*cos(2*pi*k(i)/N),1],x);
    X(i) = v(N+1)-exp(-1j*2*pi*k(i)/N)*v(N);
end
```

Figure 8.15 MATLAB function for Goertzel's algorithm.

$$
\begin{aligned}
H_k(z) &= \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^k z^{-1}}{(1 - W_N^{-k} z^{-1})(1 - W_N^k z^{-1})} \\
&= \frac{1 - W_N^k z^{-1}}{1 - 2\cos(2\pi k/N) z^{-1} + z^{-2}} \\
&= \left[\frac{1}{1 - 2\cos(2\pi k/N) z^{-1} + z^{-2}} \right] \left[1 - W_N^k z^{-1} \right],
\end{aligned}
\tag{8.60}
$$

which suggests a two-step approach. The first term on the right is a recursive second-order filter with real coefficients that can be operated on the input $x[n]$ to obtain an intermediate signal, say $v[n]$. The second term is an FIR filter with a complex coefficient that processes $v[n]$ but it needs to be operated only at $n = N$ to determine $X[k]$. Thus the *modified* Goertzel's algorithm (Goertzel (1958)) is given by

$$
v_k[n] = 2\cos(2\pi k/N) v_k[n-1] - v_k[n-2] + x[n], \quad 0 \le n \le N
\tag{8.61a}
$$
$$
X[k] = y_k[N] = v_k[N] - W_N^k v_k[N-1],
\tag{8.61b}
$$

with initial conditions $v_k[-1] = v_k[-2] = 0$. This algorithm requires $O(N^2)$ real and $O(N)$ complex operations to compute all DFT values and hence is more efficient in computation than the one in (8.59). The MATLAB function gafft, given in Figure 8.15, implements Goertzel's algorithm of (8.61) for the computation of one sample of $x[k]$. A similar function called goertzel is available in the SP toolbox.

As an additional bonus, computations for $X[k]$ can also be used for its symmetric frequency component $X[n - k]$ even when $x[n]$ is complex-valued. This is because for frequencies $2\pi k/N$ and $2\pi(N - k)/N$ we have $\cos(2\pi k/N) = \cos[2\pi(N - k)/N]$. Hence the recursive part (8.61a) is the same leading to $v_{N-k}[n] = v_k[n]$. The only change is in the FIR part in which the multiplier for $X[N - k]$ computation is $W_N^{N-k} = W_N^{-k}$, which is the complex-conjugate of W_N^k. Therefore, the overall computations are of $O(\frac{1}{2}N^2)$ for about 50% savings. An application of Goertzel's algorithm in detecting single-tone sinusoidal signals is explored in Problem 30 and in detecting dual-tone multifrequency (DTMF) signals in Problem 48.

Chirp transform algorithm (CTA)

This approach for computation of the DFT also uses a convolution operation but is very different from Goertzel's algorithm. It is also versatile in the sense that in addition to computing the DFT it can also compute, with a small modification, DTFT values at any set of equally spaced samples on the unit circle. We again begin with (8.1) and, in order to present the computation as a convolution operation, we interchange variables n and k to express

$$X[n] = \sum_{k=0}^{N-1} x[k] W_N^{kn}. \tag{8.62}$$

Rearranging the indices term kn using the identity (Bluestein (1970)):

$$kn = \tfrac{1}{2}\left[k^2 - (n-k)^2 + n^2 \right], \tag{8.63}$$

we obtain

$$X[n] = \sum_{k=0}^{N-1} x[k] W_N^{\frac{k^2}{2}} W_N^{-\frac{(n-k)^2}{2}} W_N^{\frac{n^2}{2}} = \left\{ \sum_{k=0}^{N-1} \left(x[k] W_N^{\frac{k^2}{2}} \right) W_N^{-\frac{(n-k)^2}{2}} \right\} W_N^{\frac{n^2}{2}}$$

$$= \left\{ \left(x[n] W_N^{n^2/2} \right) * W_N^{-n^2/2} \right\} W_N^{n^2/2}. \quad 0 \le n \le N-1 \tag{8.64}$$

If we define an impulse response

$$h[n] \triangleq W_N^{-n^2/2} = e^{j(\pi n/N)n}, \tag{8.65}$$

then (8.64) suggests a three step procedure: (i) modify $x[n]$ to obtain a new sequence $x[n]/h[n]$; (ii) convolve the modified sequence with the impulse response $h[n]$; and finally (iii) divide the result by $h[n]$ to obtain $X[n]$.

It is interesting to note that the impulse response $h[n]$ in (8.65) can be thought of as a complex exponential signal with *linearly increasing* frequency $n\omega_0$ where $\omega_0 = \pi/N$. Since in Chapter 5 such signals were termed *chirp signals*, the computations in (8.64) can be considered as a transform involving chirp signals.

The algorithm in (8.64) can be modified to compute M DTFT values over the frequency range $\Delta_\omega \triangleq \omega_H - \omega_L$ on the unit circle where ω_L and ω_H are the lower- and upper-end of the range, respectively. The set of frequencies $\{\omega_n\}$ over this range is given by

$$\omega_n \triangleq \omega_L + \frac{\Delta_\omega}{M-1} n \triangleq \omega_L + \delta_\omega n, \quad 0 \le n \le M \tag{8.66a}$$

$$\delta_\omega = \frac{\Delta_\omega}{M-1}. \tag{8.66b}$$

Figure 8.16 shows the plot of the z-plane indicating the DTFT frequencies in (8.66a). The DTFT values are now given by
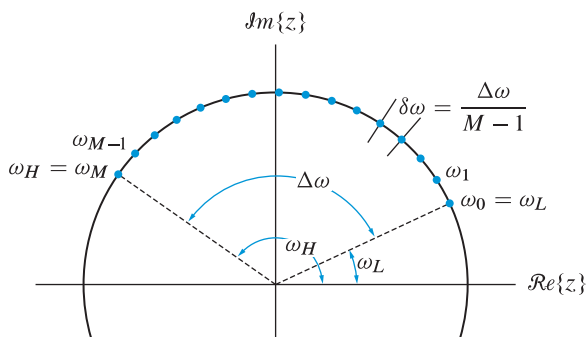
Figure 8.16 Frequency range and the set of frequencies in (8.66a) for the CTA.
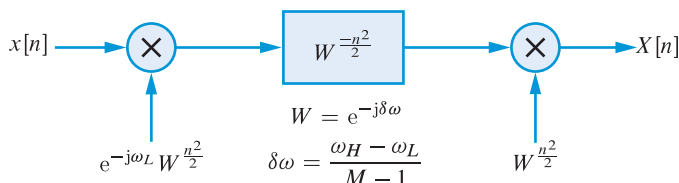
Figure 8.17 Block diagram implementation of the CTA.

$$X[n] = X(e^{j\omega_n}) = \sum_{k=0}^{N-1} x[k] \left\{ e^{-j(\omega_L + \delta_\omega n)} \right\}^k = \sum_{k=0}^{N-1} \left\{ x[k] e^{-j\omega_L k} \right\} \left\{ e^{-j\delta_\omega} \right\}^{nk}$$

$$\triangleq \sum_{k=0}^{N-1} g[k] W^{nk}, \quad n = 0, 1, \ldots, M \tag{8.67}$$

where we have defined

$$g[n] \triangleq x[n] e^{-j\omega_L n} \quad \text{and} \quad W \triangleq e^{-j\delta_\omega}. \tag{8.68}$$

Now (8.67) is similar to (8.62) and hence can be computed using steps similar to those in (8.64), that is

$$X[n] = \left\{ \left(g[n] W^{n^2/2} \right) * W^{-n^2/2} \right\} W^{n^2/2}. \quad 0 \le n \le M \tag{8.69}$$

This algorithm is known as the chirp transform algorithm (CTA) and its block-diagram implementation is shown in Figure 8.17, in which the filtering can be performed using high-speed convolution.

There are several advantages of the CTA over the FFT algorithms of previous sections which include: (a) the number of samples N in the input sequence need not be same as the number of frequency samples M on the unit circle; (b) both N and M need not be composite numbers and, in fact, can be prime numbers; (c) the frequency resolution δ_ω can be arbitrary; and (d) the starting frequency ω_L can be arbitrary, which along with the arbitrary frequency resolution can be used to perform a high-density narrowband spectral analysis.

```
function [X,w] = cta(x,M,wL,wH)
% Chirp Transform Algorithm (CTA)
%   Given x[n] CTA computes M equispaced DTFT values X[k]
%   on the unit circle over wL <= w <= wH
% [X,w] = cta(x,M,wL,wH)
Dw = wH-wL; dw = Dw/(M-1); W = exp(-1j*dw);
N = length(x); nx = 0:N-1;
K = max(M,N); n = 0:K; Wn2 = W.^(n.*n/2);
g = x.*exp(-1j*wL*nx).*Wn2(1:N);
nh = -(N-1):M-1; h = W.^(-nh.*nh/2);
y = conv(g,h);
X = y(N:N+M-1).*Wn2(1:M); w = wL:dw:wH;
```

Figure 8.18 MATLAB function for the chirp transform algorithm.

Computational aspects of CTA The CTA of (8.69) requires sequences of various lengths and hence care must be taken in evaluating the convolution contained in (8.69). The sequence $x[n]$ and hence $g[n]$ is of length N over $0 \le n \le N - 1$. The desired convolution result $X[n]$ is an M-point sequence over $0 \le n \le M - 1$. Therefore although the impulse response $h[n] = W^{-n^2/2}$ is of infinite length, we will need its segment only from $-(N - 1)$ to $M - 1$. In MATLAB, since indexing begins at 1, we need to extract convolution results over the $N \le n \le (N + M - 1)$ range. The MATLAB function `cta` given in Figure 8.18 contains these details.

For large values of M and N, instead of using convolution, as done in `cta`, one could use `fft` for fast convolution as depicted in Figure 8.17. In this implementation care must be taken to select the size of the FFT, which should be $L \ge (M + N)$, and to make sure that the impulse response values over the range $-(N - 1) \le n < -1$ are properly mapped to the positive side using modulo-L operation. These details and the implementation are explored in Problem 31.

Chirp z-transform (CZT) A further modification of the CTA in (8.69) can lead to computation of the z-transform over a spiral contour in the z-plane (Rabiner et al. (1969)). If we make the following changes to the definitions in (8.68)

$$e^{j\omega_L} \rightarrow Re^{j\omega_L} \qquad \Rightarrow \qquad g[n] = x[n]\frac{1}{R}e^{-j\omega_L}, \tag{8.70a}$$

$$e^{j\delta_\omega} \rightarrow re^{j\delta_\omega} \qquad \Rightarrow \qquad W = \frac{1}{r}e^{-j\delta_\omega}, \tag{8.70b}$$

then it can be shown that (8.67) computes the values of the z-transform on a spiraling contour given by (see Tutorial Problem 14)

$$z_n = \left(Re^{j\omega_L}\right)\left(re^{j\delta_\omega}\right)^n, \quad 0 \le n \le M \tag{8.71}$$

which begins at a complex location $Re^{j\omega_L}$ and spirals in (out) if $r < 1$ ($r > 1$). The resulting algorithm, similar to (8.69), to compute $X(z_n)$ is given by

$$X(z_n) = \left\{ \left(g[n]W^{n^2/2} \right) * W^{-n^2/2} \right\} W^{n^2/2}, \quad 0 \le n \le M \tag{8.72}$$

and is known as the chirp z-transform (CZT) algorithm. Clearly, CTA is a special case of the CZT. See Tutorial Problem 14 for more detail and the implementation of the CZT. MATLAB also provides the function czt in its SP Toolbox.

8.7.3 The zoom-FFT

This algorithm also efficiently computes the DFT of a sequence over a narrow band of frequencies along the unit circle. It is an alternative to the CTA but does not use the convolutional approach, instead its method is similar to the first stage of the DIT decomposition. Let us assume that we want to "zoom" the frequency range $\Delta_\omega = \omega_H - \omega_L$ in M samples using a higher frequency resolution $\delta_\omega \triangleq 2\pi/N$ than can be provided by the given signal length (assumed to be less than N) but without computing all N DFT values $X[k]$ of a sequence $x[n]$. Let the corresponding frequency index range be

$$k_L \le k \le k_L + M - 1 = k_H, \tag{8.73}$$

where $\omega_L = k_L\delta_\omega$ and $\omega_H = k_H\delta_\omega$. These quantities are illustrated in Figure 8.19. Let us further assume that $N = ML$ for some integer L which can always be found by zero-padding $x[n]$, if necessary.

To develop the zoom-FFT algorithm, we now decimate the sequence $x[n]$ into L sub-sequences of length M by expressing the index n as

$$n = \ell + mL, \quad 0 \le \ell \le L - 1, \ 0 \le m \le M - 1 \tag{8.74}$$

and let

$$x_\ell[m] \triangleq x[\ell + mL], \quad 0 \le m \le M - 1, \ 0 \le \ell \le L - 1 \tag{8.75}$$

be L sub-sequences, each of length M. From (8.73) we want to compute DFT $X[k]$ over $k_L \le k \le k_H$ only. Hence from (8.1) we have

$$X[k] = \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{\ell=0}^{L-1}\sum_{m=0}^{M-1} x[\ell + mL]W_N^{(\ell+mL)k}$$

$$= \sum_{\ell=0}^{L-1}\left[\sum_{m=0}^{M-1} x_\ell[m]W_M^{mk}\right] W_N^{\ell k}. \quad k_L \le k \le k_H \tag{8.76}$$
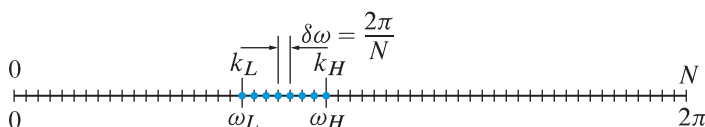
Figure 8.19 Frequency range and the index set for the zoom-FFT.

The expression inside the square bracket in (8.76) is an M-point DFT of $x_\ell[m]$ evaluated at the index range $k_L \leq k \leq k_H$. Let

$$X_\ell[r] \triangleq \sum_{m=0}^{M-1} x_\ell[m]W_M^{mr}, \quad 0 \leq r \leq M-1 \tag{8.77}$$

be the M-point DFT which is periodic in r with period M. Clearly, the computation in (8.76) represents the first stage decomposition in the DIT-FFT algorithm. Due to this periodicity, the expression inside the square bracket in (8.76) is given by

$$\sum_{m=0}^{M-1} x_\ell[m]W_M^{mk} = X_\ell[\langle k \rangle_M]. \quad k_L \leq k \leq k_H \tag{8.78}$$

Substituting (8.78) in (8.76) the zoom-FFT algorithm can be expressed as

$$X[k] = \sum_\ell^{L-1} X_\ell[\langle k \rangle_M]W_N^{\ell k}, \quad k_L \leq k \leq k_H \tag{8.79}$$

which suggests a three-step implementation: (i) the given sequence $x[n]$ is zero padded, if necessary, to length N and is decimated into L sub-sequences of length M using (8.75); (ii) an M FFT is taken of each sub-sequence $x_\ell[m]$; and (iii) for each k in $[k_L, k_H]$, a weighted combination of DFT values is performed according to (8.79) *after* reindexing DFTs using the modulo-M operation. The MATLAB function `zfa` given in Figure 8.20 incorporates these steps, in which M is assumed to be a power of 2 for efficient implementation.

```
function [X,w] = zfa(x,M,wL,wH)
% Zoom-FFT Algorithm (ZFA):
%    Given x[n] ZFA computes M equispaced DTFT values X[k]
%    on the unit circle over wL <= w <= wH
% [X,w] = zfa(x,M,wL,wH)
% Cautions:
%      1. M must be a power of 2
%      2. (2*pi)/(wH-wL)*M must be larger than length(x)
%
M = 2^(ceil(log2(M))); Nx = length(x);
L = ceil(2*pi/(wH-wL)*(M-1)/M); l = 0:L-1;
N = L*M; dw = 2*pi/N; k0 = floor(wL/dw);
k = k0:1:k0+(M-1); w = k*dw;
x = [reshape(x,1,Nx),zeros(1,N-Nx)];
x = (reshape(x,L,M))'; X = fft(x);
WN = exp(-1j*2*pi*k'*l/N);
k = mod(k,M); X = X(k+1,:); X = sum(X.*WN,2);
```

Figure 8.20 MATLAB function for the zoom-FFT algorithm.

The main computational complexity of the zoom-FFT algorithm is in the execution of the second and third steps above. The second step needs $O(LM\log M) = O(N\log M)$ complex operations using M as an integer power of 2. The third step requires $O(LM) = O(N)$ complex operations. In addition, we also need to compute N $\{W_N^{\ell k}\}$ complex numbers. Thus the total number of complex operations is $O(N\log M + 2N)$, which is substantially less than $O(N\log N)$ for the entire N-point FFT if $M \ll N$. This is the main advantage of the zoom-FFT over the entire FFT approach. However, unlike the CTA, we cannot precisely specify the lower and upper band edges, ω_L and ω_H respectively, due to the constraint $N = LM$. For large values of M and N, this is not of a serious concern.

8.7.4 A quick Fourier transform (QFT)

Another approach for removing redundant DFT computations can be developed if symmetries inherent in its definition are exploited. We discussed in detail various Cooley–Tukey based FFT algorithms that exploited periodicity and symmetry properties of the twiddle factor W_N^{nk} and developed arithmetically efficient algorithms for highly composite lengths. The possibility of using basic symmetry in the DFT definition was long recognized, however, Guo *et al.* (1998) recently provided a systematic development of such an approach and termed it Quick Fourier Transform (QFT). Although is not is as super efficient or extremely fast as Cooley–Tukey based FFT algorithms, nevertheless it is quicker than the direct approach and can be used for any length sequence.

Consider again the DFT in (8.1) which can be written as

$$
\begin{aligned}
X[k] &= \sum_{n=0}^{N-1} x[n]W_N^{nk} = \sum_{n=0}^{N-1} x[n]\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}nk} \\
&= \sum_{n=0}^{N-1} x[n]\left\{\cos\left(\frac{2\pi}{N}nk\right) - \mathrm{j}\sin\left(\frac{2\pi}{N}nk\right)\right\},
\end{aligned}
\tag{8.80}
$$

where we have indicated even-real and odd-imaginary parts of the twiddle factor. In QFT, circular-conjugate symmetries in signal $x[n]$ are also exploited. Using (7.113) we can express $x[n]$ as

$$
x[n] = x^{\mathrm{cce}}[n] + x^{\mathrm{cco}}[n] = x_{\mathrm{R}}^{\mathrm{ce}}[n] + \mathrm{j}x_{\mathrm{I}}^{\mathrm{ce}}[n] + x_{\mathrm{R}}^{\mathrm{co}}[n] + x_{\mathrm{I}}^{\mathrm{co}}[n],
\tag{8.81}
$$

where $x^{\mathrm{cce}}[n]$ and $x^{\mathrm{cco}}[n]$ are circularly-conjugate-even and circularly-conjugate-odd parts of $x[n]$ given in (7.114) and (7.115), respectively. Substituting (8.81) in (8.80) and noting that the sum of odd components over full signal length N is zero and the sum of even components over half the signal length is one half of the sum over the full length, we obtain

$$
\begin{aligned}
X[k] = 2\sum_{n=0}^{\frac{N}{2}-1} &\left[\left\{x_{\mathrm{R}}^{\mathrm{ce}}[n]\cos\left(\frac{2\pi}{N}nk\right) + x_{\mathrm{I}}^{\mathrm{co}}[n]\sin\left(\frac{2\pi}{N}nk\right)\right\}\right. \\
&\left. + \mathrm{j}\left\{x_{\mathrm{I}}^{\mathrm{ce}}[n]\cos\left(\frac{2\pi}{N}nk\right) - x_{\mathrm{R}}^{\mathrm{co}}[n]\sin\left(\frac{2\pi}{N}nk\right)\right\}\right],
\end{aligned}
\tag{8.82}
$$

which requires approximately half the number of real multiply or addition operations of direct computations. Furthermore, if k in (8.82) is replaced by $N - k$ then, using symmetry in sine and cosine in the k variable, it is easy to show that

$$X[N - k] = 2 \sum_{n=0}^{\frac{N}{2}-1} \left[\left\{ x_R^{ce}[n] \cos\left(\frac{2\pi}{N}nk\right) - x_I^{co}[n] \sin\left(\frac{2\pi}{N}nk\right) \right\} \right.$$

$$\left. + j \left\{ x_I^{ce}[n] \cos\left(\frac{2\pi}{N}nk\right) + x_R^{co}[n] \sin\left(\frac{2\pi}{N}nk\right) \right\} \right]. \qquad (8.83)$$

Thus if both $X[k]$ and $X[n - k]$ are needed then an additional reduction by a factor of two is obtained because both need the same intermediate products. This total reduction of about 25% is obtained without even exploiting any efficiencies afforded by composite length of the signal. If $N = 2^\nu$, then an approach similar to the divide-and-conquer can be employed to substantially reduce the total computational complexity. Guo *et al.* (1998) have developed a QFT algorithm based on discrete cosine and sine transforms to achieve $O(N\log_2 N)$ complexity. More details can be found in Guo *et al.* (1998).

8.7.5   Sliding DFT (SDFT)

Finally, we consider use and computation of the DFT in determining spectra of a very long sequence that has time-varying spectral properties. In such situations, instead of computing one large size DFT, we compute several shorter length DFTs at each successive time instance by sliding a fixed-size temporal window. Hence the technique is known as sliding DFT (SDFT). It provides a time-frequency plot or spectrogram (see Section 7.6.5), that captures the time-varying spectral properties of the signal.

Let $x[n]$, $n \geq 0$, be a long sequence and let $x_n[m]$ be an N-point segment at n created using a sliding window as shown in Figure 8.21, that is

$$x_n[m] \triangleq \begin{cases} x[n - N + 1 + m], & 0 \leq m \leq N - 1 \\ 0. & \text{otherwise} \end{cases}, \quad n \geq N - 1 \qquad (8.84)$$
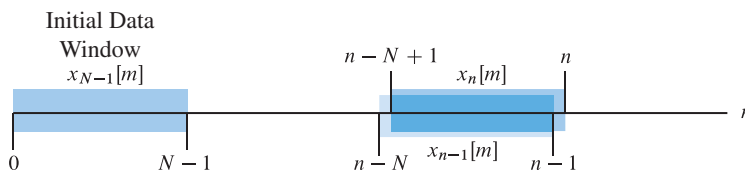
Figure 8.21 Initial data window and sliding data windows used in SDFT.

Then the time-dependent DFT of $x_n[m]$ is given by

$$X_n[k] = \sum_{m=0}^{N-1} x_n[m] W_N^{mk}$$

$$= \sum_{m=0}^{N-1} x[n - N + 1 + m] W_N^{mk}. \quad n \geq N - 1, \ 0 \leq k \leq N - 1 \quad (8.85)$$

It is straightforward to obtain a time-recursive relation for $X_n[k]$ (see Tutorial Problem 15) as

$$X_n[k] = \{X_{n-1}[k] + x[n] - x[n - N]\} W_N^{-k}, \quad n \geq N, \ 0 \leq k \leq N - 1 \quad (8.86)$$

starting with the first DFT

$$X_{N-1}[k] = \sum_{m=0}^{N-1} x[n] W_N^{mk}, \quad 0 \leq k \leq N - 1 \quad (8.87)$$

Thus, after the first DFT is computed using one of the algorithms previously discussed, the subsequent DFT values can be computed using N complex operations instead of O($N\log_2 N$) if an FFT algorithm is used at each n. It should be emphasized that the SDFT algorithm (8.86) essentially implements a rectangular data window in its recursive computation of the spectrum. It is not suited for any other types of data window as discussed in Section 7.6.3 because the window functions are not amenable to recursive implementation.

Although the SDFT in (8.86) provides a computational improvement over FFT algorithms, it suffers from one drawback in its implementation. If (8.86) is considered as a linear filter with $x[n]$ as the input and $X_n[k]$ as the output (where n is the time index), then its system function for each k is given by

$$H_k(z) = W_N^{-k} \frac{1 - z^{-N}}{1 - W_N^{-k} z^{-1}}, \quad 0 \leq k \leq N - 1 \quad (8.88)$$

which has a pole on the unit circle at $z = W_N^{-k} = e^{\frac{2\pi}{N}k}$. This means that even in an ideal situation the SDFT in (8.86) is critically stable. When the filter is implemented with finite precision (see Chapter 14), it can become unstable if its pole moves outside the unit circle. One approach to stabilizing the algorithm is to move the pole slightly inside the unit circle by replacing W_N^{-k} by $r W_N^{-k}$ in (8.86) where $r \approx 1$ but $r < 1$. Over the long interval, this approximation can result in an accumulated error in DFT values, which means that the algorithm should be reinitialized periodically using an FFT algorithm.

Learning summary

- Direct computation of the N-point DFT using the defining formula

$$X[k] = \sum_{n=0}^{N-1} x[n]\, W_N^{kn}, \quad k = 0, 1, \ldots, N-1$$

requires $O(N^2)$ complex operations. It is assumed that the twiddle factors $W_N^{kn} = e^{-2\pi kn/N}$ are obtained from a look-up table.

- Fast Fourier Transform (FFT) algorithms reduce the computational complexity from $O(N^2)$ to $O(N \log_2 N)$ operations. This immense reduction in complexity is obtained by a divide-and-conquer strategy which exploits the periodicity and symmetry properties of the twiddle factors.

- The decimation-in-time radix-2 FFT algorithm, which requires $N = 2^\nu$, is widely used because it is easy to derive, simple to program, and extremely fast. More complicated FFTs can reduce the number of operations, compared to the radix-2 FFT, only by a small percentage.

- For many years the time for the computation of FFT algorithms was dominated by multiplications and additions. The performance of FFTs on current computers depends upon the structure of the algorithm, the compiler, and the architecture of the machine.

- For applications which do *not* require all N-DFT coefficients, we can use algorithms based on linear filtering operations; these include Goertzel's algorithm and the chirp transform algorithm.

TERMS AND CONCEPTS

Bit-reversed order The sequential arrangement of data values in which the order is determined by reversing the bit arrangement from its natural order, that is, the bit arrangement $b_B \cdots b_1 b_0$ in natural order is reversed to $b_0 b_1 \cdots b_B$.

Butterfly computation A set of computations that combines the results of smaller DFTs into a larger DFT, or vice versa (that is, breaking a larger DFT up into smaller DFTs). Also known as a merging formula.

Chirp z-transform (CZT) A modified version of the CTA that can be used to compute z-transform over a spiral contour in the z-plane.

Chirp signal A sinusoidal signal with a frequency that grows linearly with time.

Chirp transform algorithm (CTA) A DFT algorithm that uses convolution with a chirp

signal as the computation strategy, which is useful in performing high-density narrowband spectral analysis.

Computational cost or complexity The number of arithmetic operations needed to compute one DFT coefficient (per sample cost) or all coefficients (algorithm cost). It is defined using several metrics: real/complex additions, real/complex multiplications, etc. It is stated using the $O(\cdot)$ notation.

Decimation-in-frequency (DIF) FFT A class of divide-and-conquer FFT algorithms that decompose the DFT sequence $X[k]$ into smaller sequences for DFT computations.

Decimation-in-time (DIT) FFT A class of divide-and-conquer FFT algorithms that decompose the signal sequence $x[n]$ into smaller sequences for DFT computations.

DFT matrix An $N \times N$ matrix formed using Nth roots of unity and denoted by $\mathbf{W}_N$.

Direct DFT algorithm A direct implementation of the DFT algorithm to compute coefficients at a computational cost proportional to N^2.

Discrete Fourier Transform (DFT) A transform-like operation on a finite-length N-point sequence $x[n]$ resulting in a finite-length N-point sequence $X[k]$ given by

$$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}.$$

Divide-and-conquer approach A computation scheme in which an exponential cost of computation is reduced significantly by dividing a long sequence into smaller sub-sequences, performing the computations on sub-sequences, and then combining results into the larger computation.

Fast Fourier Transform (FFT) The Cooley–Tukey class of efficient DFT computation algorithms designed to obtain all DFT coefficients as a block, with computational cost proportional to $N\log_2 N$.

FFTW algorithm Acronym for the Fastest Fourier Transform in the West algorithm which is a software library that optimizes FFT implementations for the best performance given a target computer architecture.

Goertzel's algorithm A recursive algorithm that is efficient for the computation of some DFT coefficients and that uses only the periodicity property of the twiddle factors.

Horner's rule An efficient polynomial computation technique in which exponential evaluations are replaced by nested multiplications.

In-place algorithm An algorithm that uses the same memory locations to store both the input and output sequences.

Merging formula An operation that combines two shorter-length DFTs into a longer-length DFT. Also known as a FFT butterfly.

Mixed radix FFT algorithm A divide-and-conquer algorithm in which length N can be factored as $N = R_1 R_2 \ldots R_M$ for splitting and merging.

Natural order The sequential arrangement of data values according to the order in which they are obtained in nature, that is, in order from $1, 2, \ldots$, and so on.

Prime factor algorithm (PFA) A family of more efficient algorithms for length N that can be factored into relatively prime factors in which efficiency is achieved by eliminating all twiddle factors.

Quick Fourier transform (QFT) An algorithm to compute DFT that systematically uses basic symmetries in the DFT definition to achieve efficiency through real arithmetic.

Radix-2 FFT algorithm The basic Cooley–Tukey algorithm that uses length N as a power of 2 for splitting sequences into two sub-sequences of equal length at each stage of the algorithm.

Radix-R FFT algorithm A divide-and-conquer algorithm that uses length N as a power of R for splitting sequences into R sub-sequences of equal length at each stage of the algorithm.

Reverse carry algorithm A recursive and efficient algorithm to determine the bit-reversed order that propagates a carry from right-to-left (or a reverse carry).

Shuffling operation A reordering of sequences performed prior to merging (DIT-FFT) or after merging (DIF-FFT) operations. For radix-2 FFT, this shuffling results in bit-reversed ordering.

Sliding DFT (SDFT) A recursive algorithm that computes shorter fixed-length DFTs at each n for a very-long length signal to obtain a time-frequency plot or spectrogram that captures time-varying spectral properties of the signal.

Split-radix FFT algorithm (SR-FFT) A specialized algorithm that for $N = 2^v$ uses radix-2 DIF-FFT for computation of even-indexed coefficients and radix-4 DIF-FFT for computation of odd-indexed

coefficients resulting in an overall faster algorithm.

Spectrogram A 2D time–frequency plot that captures time-varying spectral properties of a signal.

Twiddle factor The root-of-unity, $W_N = e^{-j\frac{2\pi}{N}}$, complex multiplicative constants in the butterfly operations of the Cooley–Tukey FFT algorithm.

Winograd Fourier transform algorithm (WFTA) An FFT algorithm that minimizes the number of multiplications at the expense of an increased number of additions.

Zoom FFT algorithm (ZFA) An efficient algorithm that computes DFT over a narrow band of frequencies along the unit circle without using the convolution approach used in CTA.

MATLAB functions and scripts

Name	Description	Page
dftdirect*	Direct computation of the DFT	436
fftrecur*	Recursive computation using divide & conquer	439
bitrev*	Bit-reversal algorithm based on Gold and Rader (1969)	446
fftditr2*	Decimation-in-time radix-2 FFT algorithm	449
gafft*	Goertzel's algorithm	461
fft	Fast algorithm for computation of the 1D-DFT	458
fft2	Fast algorithm for computation of the 2D-DFT	429
fftshift	Moves the zero-frequency component to the center	458
cta*	Chirp transform algorithm	464
czt*	Chirp z-transform	465
ifft	Fast algorithm for computation of the inverse 1D-DFT	458
ifft2	Fast algorithm for computation of the inverse 2D-DFT	429
ifftshift	Moves time-origin component to the center	458
zfa*	Zoom FFT algorithm	466

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A detailed treatment of FFT algorithms, at the same level as in this book, is given in Oppenheim and Schafer (2010), Proakis and Manolakis (2007), Mitra (2006), and Brigham (1988).
2. The classical introduction to the FFT is given by Cochran *et al.* (1967) in a paper entitled "What is the Fast Fourier Transform?" The history of the FFT is discussed by Cooley *et al.* (1967), Cooley (1992), and Heideman *et al.* (1984).
3. Van Loan (2000) provides a unified derivation and organization of most known FFT algorithms using a matrix factorization approach. A detailed algebraic treatment of FFT algorithms is given in Blahut (2010). Duhamel and Vetterli (1990) present a tutorial review of various FFT algorithms.
4. The two-dimensional FFT and its applications in image processing are discussed in Gonzalez and Woods (2008) and Pratt (2007). Applications of three-dimensional FFT to video processing can be found in Woods (2006).

Review questions

1. Explain the computational cost involved in computing an N-point DFT using direct computation.
2. What should be the general goal of an efficient DFT algorithm for a data length of N sample?
3. Provide the steps needed to compute an inverse DFT using a DFT algorithm. What is the total computational cost of this approach?
4. Which two basic properties of the DFT lead to efficient FFT algorithms?
5. How does the "divide and conquer" strategy result in an efficient computation of the DFT?
6. Which two basic strategies are used in the development of classic FFT algorithms?
7. Explain the decimation-in-time FFT algorithm using the matrix interpretation.
8. Explain the decimation-in-frequency FFT algorithm using the matrix interpretation.
9. In the algebraic development of the DIT-FFT (or DIF-FFT) algorithm, two key steps are required at each stage. Explain those steps.
10. What is the computational cost of a radix-2 DIT-FFT algorithm? That of a radix-2 DIF-FFT algorithm?
11. Explain DIT and DIF FFT butterflies and their role in the FFT algorithms.
12. What is bit-reversed ordering and how is it implemented using a reverse-carry algorithm?
13. Some alternative forms of radix-2 FFT algorithm are discussed in the chapter. Explain the advantages and disadvantages of each.
14. A signal has length $N = N_1 N_2 N_3$ where N_1, N_2, N_3 are all prime numbers. Determine the computational cost in computing its DFT using the divide-and-conquer approach.
15. Describe practical considerations needed in the implementation of FFT algorithms that use a divide-and-conquer approach.
16. What are the implementational factors that determine the speed of FFT algorithms on modern computer systems?
17. How does the FFTW algorithm implement DFT computations?
18. What is the basic principle used in the development of Goertzel's algorithm?
19. Describe the advantages and disadvantages of Goertzel's algorithm over the FFT algorithms.
20. What is the basic idea used in the development of the CTA?
21. Describe advantages of the CTA over the FFT algorithms.
22. What is CZT and how is it different from the CTA?
23. What is the basic idea used in the development of the zoom-FFT?
24. For which applications is the zoom-FFT more suitable than the basic FFT algorithm?
25. Describe the basic properties used in development of the QFT.
26. What is the advantage of the QFT over the FFT?
27. What is the main approach used in development of the sliding DFT?

28. Explain why the sliding DFT can become unstable and what measures are taken to stabilize the algorithm?

Problems

Tutorial problems

1. Using the `tic` and `toc` functions we will investigate trends in the computation times for the direct computation of the DFT. Write a MATLAB script that computes and plots computation times for the `dftdirect` function for $N = 2^\nu$ where $2 \leq \nu \leq 10$. For this purpose generate an N-point complex-valued signal as `x = randn(1,N) + 1j*randn(1,N)`. Describe the resulting trend in the computational complexity of the direct DFT computations.

2. Consider the matrix divide-and-conquer approaches in Section 8.2 for $N = 4$.
   (a) Write the four equations in matrix form and change the order of input terms to even and odd order to develop the DIT matrix algorithm.
   (b) Write the four equations and change the order of equations to even and odd order to develop the DIF matrix algorithm.

3. Consider the DIF matrix-computation approach for $N = 8$ given in (8.16) and (8.17).
   (a) Develop the algorithm by completing all three stages of decimation.
   (b) Using steps developed in part (a) write a recursive MATLAB function `X = difrecur(x)`.
   (c) Verify your function on the sequence $x[n] = \{1, 2, 3, 4, 5, 4, 3, 2\}$.
   ↑

4. Let $x[n] = \{1, 2, 3, 4, 5, 4, 3, 2\}$ be an 8-point sequence.
   ↑
   (a) Decimate $x[n]$ into $a[n]$ by choosing every other sample and compute its 4-point DFT in $A[k]$, for $0 \leq k \leq 7$.
   (b) First circularly shift $x[n]$ to the left by one sample and then decimate the result into $b[n]$ by choosing every other sample and compute its 4-point DFT in $B[k]$, for $0 \leq k \leq 7$.
   (c) Now combine $A[k]$ and appropriately modified $B[k]$ due to circular shifting into $X[k]$.
   (d) Compare your results in (c) with the 8-point DFT of $x[n]$ and explain the DFT interpretation of $X[k]$ in terms of $A[k]$ and $B[k]$.

5. Consider the computation of twiddle factors $W_N^{q\ell}$, for some q and ℓ.
   (a) For $N = 16$ and $q = 1$, determine $W_N^{q\ell}$, $0 \leq \ell \leq 8$ using direct calculations and determine the number of complex multiplications. Assume that exponentiation is obtained using continued multiplications.
   (b) Use the recursion formula $W_N^{q\ell} = W_N^q W_N^{q(\ell-1)}$ to compute the ℓth coefficient from the $(\ell - 1)$th coefficient. Now determine the number of complex multiplications.

6. Following the procedure given in DIF derivation for the even-indexed $X[2k]$ in (8.36) verify the formula for $X[2k+1]$ given in (8.37).

7. Let N be a power of 2. The DIF algorithm can also be obtained by splitting the DFT sum (8.1) into two sums. Let N be a power of 2.

   (a) Starting with (8.1), show that $X[k]$ can be written as a sum over n from 0 to $\frac{N}{2} - 1$.

   (b) For even-indexed k and for odd-indexed k, show that the sum in (a) above results in an $\frac{N}{2}$-point DFT.

   (c) Verify that the resulting equations in (b) constitute the DIF FFT algorithm as described in the chapter.

8. Using the flow graph of Figure 8.13 and following the approach used in developing the `fftditr2` function, develop a radix-2 DIF-FFT function `X = fftdifr2(x)` for power-of-2 length N.

9. The FFT flow graphs in Figures 8.6 and 8.13 can be modified to compute inverse DFT.

   (a) Transpose the DIT-FFT flow graph in Figure 8.6 and replace each multiplier in the input of every butterfly (which is of the form W_N^r, including unity) by $W_N^{-r}/2$. Show that the resulting flow graph resembles the DIF-FFT flowgraph and computes the inverse DFT.

   (b) Transpose the DIF-FFT flow graph in Figure 8.13 and replace each multiplier in the input of every butterfly (which is of the form W_N^r, including unity) by $W_N^{-r}/2$. Show that the resulting flow graph resembles the DIT-FFT flow graph and computes the inverse DFT.

10. Consider a 6-point DIT-FFT that uses a mixed-radix implementation. There are two approaches.

    (a) In the first approach take three 2-point DFTs and then combine results to obtain the 6-point DFT. Draw a flow graph of this approach and properly label all relevant path gains as well as input/output nodes. How many real multiplications and additions are needed.

    (b) In the second approach take two 3-point DFTs and then combine results to obtain the 6-point DFT. Draw a flow graph of this approach and properly label all relevant path gains as well as input/output nodes. How many real multiplications and additions are needed.

11. In the development of a DIF FFT algorithm, the even-indexed sequence $X[2k]$ can be thought of as choosing every other sample (decimation by 2) of the DFT $X[k]$ while the odd-indexed sequence $X[2k+1]$ can be thought of as first circular left-shifting by one followed by choosing every other sample (decimation by 2) of the DFT $X[k]$.

    (a) Show that the $\frac{N}{2}$-point inverse DFT of $X[2k]$, that is, the equivalent time-domain periodic signal $x_E[n]$, is given by

$$x_E[n] = x[n] + x\left[n + \frac{N}{2}\right], \quad 0 \le n \le \frac{N}{2} - 1.$$

(b) Similarly, show that the $\frac{N}{2}$-point inverse DFT of $X[2k + 1]$, that is, the equivalent time-domain periodic signal $x_O[n]$, is given by

$$x_E[n] = \left(x[n] - x\left[n + \frac{N}{2}\right] \right) W_N^n, \ 0 \le n \le \frac{N}{2} - 1.$$

12. Let $N = 15 = 3 \times 5$. In this problem we will develop $N = 15$ Cooley–Tukey DIT-FFT algorithms.
    (a) Decompose $x[n]$ into five sub-sequences of length 3 by choosing every third sample starting with $n = 0$, $n = 1$, and $n = 2$. Develop a complete set of equations to determine the 15-point DFT $X[k]$ by merging five 3-point DFTs.
    (b) Decompose $x[n]$ into three sub-sequences of length 5 by choosing every fifth sample starting with $n = 0$, $n = 1$, ..., $n = 4$. Develop a complete set of equations to determine the 15-point DFT $X[k]$ by merging three 5-point DFTs.
    (c) Determine the total number of multiplications and additions needed to implement the above algorithms.

13. Let $N = 16 = 4 \times 4$. In this problem we will develop an $N = 16$ radix-4 Cooley–Tukey DIT-FFT algorithm.
    (a) Decompose $x[n]$ into four sub-sequences of length 4 by choosing every fourth sample starting with $n = 0$, $n = 1$, $n = 2$, and $n = 3$. Develop a complete set of equations to determine the 16-point DFT $X[k]$ by merging four 4-point DFTs.
    (b) Determine the total number of multiplications and additions needed to implement the radix-4 FFT algorithm.
    (c) Verify that the number of multiplications is reduced by half compared to the radix-2 algorithm.

14. A simple modification to the definitions in (8.68) leads to the CZT algorithm.
    (a) By substituting (8.70) quantities in (8.68) show that (8.67) computes the z-transform of $x[n]$ on a spiral contour given by (8.71) and the resulting algorithm is given by (8.72).
    (b) Following the computational aspects and procedure discussed for CTA, develop a MATLAB function `[X,z] = czta(x,M,wL,wH,R,r)` that computes the CZT along the spiral contour given by (8.71).

15. Consider the SDFT $X_n[k]$ in (8.85) in which a rectangular data window is used.
    (a) Show that (8.85) can be expressed recursively as given in (8.86) by breaking the sum and adjusting the needed terms.
    (b) Instead of the rectangular data window, an exponential window of the form

$$w_e[m] = \begin{cases} \lambda^{N-1-m}, & 0 \le m \le N - 1 \\ 0, & \text{otherwise} \end{cases}$$

is used to obtain $x_n[m] = w_e[m]x[n - N + 1 + m]$ where $0 < \lambda < 1$. Determine the new recursive SDFT algorithm that is similar to (8.86).

Basic problems

16. Consider the inverse DFT given in (8.2).

    (a) Show that (8.2) can also be written as

    $$x[n] = \frac{1}{N} j \left\{ \sum_{k=0}^{N-1} (jX^*[k]) W_N^{kn} \right\}^*, \quad n = 0, 1, \ldots, N-1. \qquad (8.89)$$

    (b) The quantity inside the curly brackets is the DFT $y[n]$ of the sequence $jX^*[k]$; thus, the inverse DFT of $X[k]$ is $x[n] = (1/N)(jy^*[n])$. Note that if $c = a + jb$ then $jc^* = b + ja$. Using this interpretation draw a block diagram that computes IDFT using a DFT block that has separate real and imaginary input/output ports.

    (c) Develop a MATLAB function `x = idft(X,N)` using the `fft` function. Verify your function on signal $x[n] = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

17. Direct multiplication of two complex numbers $(a + jb)(c + jd)$ requires four real multiplications and two real additions. By properly arranging terms show that it is possible to obtain the above multiplication using three real multiplications and five real additions.

18. Using the `tic` and `toc` functions we will investigate trends in the computation times for the recursive computation of the DFT. Write a MATLAB script that computes and plots computation times for the `fftrecur` function for $N = 2^\nu$ where $2 \leq \nu \leq 10$. For this purpose generate an N-point complex-valued signal as `x = randn(1,N) + 1j*randn(1,N)`. Describe the resulting trend in the computational complexity of the recursive DFT computations and compare it with that in Problem 1.

19. Let $N = 3^\nu$.

    (a) Derive a radix-3 DIT-FFT algorithm for this case.

    (b) Draw the complete flow-chart for $N = 27$.

    (c) Determine the total number of complex multiplications needed to implement the algorithm in part (b) above.

20. Draw a complete flow graph of a 16-point radix-2 DIT-FFT algorithm. In your flow-graph indicate required branch multiplier values as powers of W_{16} except those that are -1. Input values should be in bit-reversed order while output values should be in normal order. From your flow graph determine the total number of real multiplications and additions needed to compute all DFT values.

21. The N-point DFT $X[k]$ of a sequence $x[n]$ is zero for $k_0 \leq k \leq N-k_0$. A new LN-point DFT $Y[k]$ is formed using the following construction:

    $$Y[k] = \begin{cases} X[k], & 0 \leq k \leq k_0 - 1, \\ 0, & k_0 \leq k \leq LN - k_0, \\ X[k+N-LN]. & LN - k_0 + 1 \leq k \leq LN - 1 \end{cases}$$

    Let $y[n]$ be the LN-point IDFT of $Y[k]$.

(a) Show that $y[Ln] = x[n]/L$, $0 \leq n \leq N - 1$.

(b) Determine $y[n]$ for $X[k] = [2, 0, 0, 2]$, $N = 4$, and $L = 2$.

22. In the implementation of (8.19) in the DIT-FFT algorithm, by mistake, we set $a[n] = x[2n + 1]$ and $b[n] = x[2n]$, while the rest of the steps are implemented correctly. Can we recover the true DFT $X[k]$ in (8.23)? Provide the necessary steps for this recovery.

23. A 16-point Cooley–Tukey type FFT algorithm has a twiddle factor of W_{16}^{12} in its fourth stage. Explain whether this algorithm implements a DIT or DIF approach.

24. Consider the DIT-FFT flow graph of Figure 8.6.

(a) How many paths in the flow graph begin at the input node $x[1]$ and terminate on the output node $X[2]$? From $x[4]$ to $X[7]$? What is your conclusion about the number of paths from every input node to every output node? Why does it make sense?

(b) What are the total gains along each of the paths given above?

(c) For the output node $X[4]$, trace paths from every input node that terminate on it. Using the gains along these paths, verify that $X[4] = \sum_{n=0}^{7} x[n] W_8^{4n}$.

25. Develop a radix-3 DIT-FFT algorithm and draw a flow graph for $N = 9$ in which the output is in the normal order while the input is in nonnormal order. What is the form of the input ordering?

26. Consider the flow graph in Figure 8.10 which implements a DIT-FFT algorithm with both input and output in natural order. Let the nodes at each stage be labeled as $s_m[k]$, $0 \leq m \leq 3$ with $s_0[k] = x[k]$ and $s_3[k] = X[k]$, $0 \leq k \leq 7$.

(a) Express $s_m[k]$ in terms of $s_{m-1}[k]$ for $m = 1, 2, 3$.

(b) Write a MATLAB function $X = fftalt8(x)$ that computes an 8-point DFT using the equations in part (a).

(c) Compare the coding complexity of the above function with that of the fftditr2 function on page 449 and comment on its usefulness.

27. Consider the general FFT algorithm given in Section 8.5. Let $N_1 = \frac{N}{2}$ and $N_2 = 2$. Show that equations (8.52), (8.54), and (8.55) lead to the DIT-FFT algorithm of (8.23).

28. Let $N = 15 = 3 \times 5$. In this problem we will develop $N = 15$ Cooley–Tukey DIF-FFT algorithms.

(a) Decompose $X[k]$ into five sub-sequences of length 3 by choosing every third sample starting with $k = 0$, $k = 1$, and $k = 2$. Develop a complete set of equations to determine five 3-point DFTs.

(b) Decompose $X[k]$ into three sub-sequences of length 5 by choosing every fifth sample starting with $k = 0, k = 1, \ldots, k = 4$. Develop a complete set of equations to determine three 5-point DFTs.

(c) Determine the total number of multiplications and additions needed to implement the above algorithms.

29. Let the sequence $x[n]$ be of length L and we wish to compute an N-point DFT of $x[n]$ where $L \ll N$. Assume that the first $L = 2$ signal values $x[0]$ and $x[1]$ are nonzero.

(a) Draw a radix-2 $N = 16$-point DIT-FFT flow-chart in which only those paths originating from the nonzero signal values are retained.

(b) Draw a radix-2 $N = 16$-point DIF-FFT flow-chart in which only those paths originating from the nonzero signal values are retained.

(c) Determine the total number of complex multiplications in each of the above flow graphs. Which algorithm gives the lesser number of multiplications?

(d) Develop a general rule in terms of L and N for selecting a DIT- or DIF-FFT algorithm in FFT *input pruning*.

30. The Goertzel algorithm can be used to detect single tone sinusoidal signals. We want to detect single tone signals of frequencies 490, 1280, 2730, and 3120 Hz generated by an oscillator operating at 8kHz. The Goertzel algorithm computes only one relevant sample of an N-point DFT for each tone to detect it.

(a) What should be the minimum value of DFT length N so that each tone is detected with no leakage in the adjacent DFT bins. The same algorithm is used for each tone.

(b) Modify the `gafft` function on page 461 and write a new MATLAB function `X = gafft_vec(x,N,k)` that computes DFT values at the indices given in the vector `k`.

(c) Generate one second duration samples for each tone and use the above function on each set of samples to obtain the corresponding `X` vector. How would you set the threshold to detect each signal?

31. The implementation of CTA given in the chapter uses a convolution operation. For large values of M and N a FFT-based approach is efficient. For this, two issues must be carefully considered. First, the size L of the FFT must be $L \geq (M + N)$ and second, the negative time values of the impulse response must be properly mapped to the positive side before taking its DFT. Modify the `CTA` function and write a new function `ctafft` that uses the `fft` function.

32. Let $x_c(t) = \cos(20\pi t) + \cos(18\pi t) + \cos(22\pi t)$. It is sampled at a rate of 40 Hz to obtain 128 samples in $x[n]$, $0 \leq n \leq 127$.

(a) Take a 128-point FFT of $x[n]$ and plot the magnitude spectra over $8 \leq F \leq 12$ Hz.

(b) To improve clarity in the spectral plot, take a 1024-point FFT of $x[n]$ and plot the magnitude spectra over $8 \leq F \leq 12$ Hz. Compare this plot with the above plot.

(c) Now use the `cta` function and choose parameters to display the magnitude spectra over $8 \leq F \leq 12$ Hz. Compare this plot with the above two plots.

(d) In your opinion which approach has the smallest number of computations with a better display?

Assessment problems

33. Consider again the inverse DFT given in (8.2).

(a) Replace k by $\langle -k \rangle_N$ in (8.2) and show that the resulting summation is a DFT expression, that is, IDFT$\{X[k]\} = \frac{1}{N}$DFT$\{X[\langle -k \rangle_N]\}$.

(b) Develop a MATLAB function `x = IDFT(X,N)` using the `fft` function that uses the above approach. Verify your function on signal $x[n] = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

34. Using the `tic` and `toc` functions we will investigate trends in the computation times for the DIT-FFT algorithm. Write a MATLAB script that computes and plots computation times for the `fftditr2` function for $N = 2^{\nu}$ where $2 \leq \nu \leq 10$. For this purpose generate an N-point complex-valued signal as `x = randn(1,N) + 1j*randn(1,N)`. Describe the resulting trend in the computational complexity of the DIT-FFT algorithm and compare it with that in Problem 1.

35. Suppose we need any $K \leq N$ DFT values of the N-point DFT. We have two choices: the direct approach or the radix-2 DIT-FFT algorithm. At what minimum value of K will the FFT algorithm become more efficient than the direct approach? Determine these minimum values for $N = 128$, 1024, and 8192.

36. Draw a complete flow graph of a 16-point radix-2 DIF-FFT algorithm. In your flow graph indicate required branch multiplier values as powers of W_{16} except those that are -1. Input values should be in normal order while output values should be in bit-reversed order. From your flow graph determine the total number of real multiplications and additions needed to compute all DFT values.

37. A 64-point Cooley–Tukey type FFT algorithm has a twiddle factor of W_{64}^{30} in its first stage. Explain whether this algorithm implements a DIT or DIF approach.

38. Consider a 6-point DIF-FFT that uses a mixed-radix implementation. There are two approaches.

(a) In the first approach combine two inputs in three sequences and take 3-point DFTs to obtain the 6-point DFT. Draw a flow graph for this approach and properly label all relevant path gains as well as input/output nodes. How many real multiplications and additions are needed.

(b) In the second approach combine three inputs in two sequences and take 2-point DFTs to obtain the 6-point DFT. Draw a flow graph for this approach and properly label all relevant path gains as well as input/output nodes. How many real multiplications and additions are needed.

39. In this problem we will investigate the transpose of the DIF-FFT flow graph defined by butterfly (8.45).

(a) Express $X_{m-1}[p]$ and $X_{m-1}[q]$ in terms of $X_m[p]$ and $X_m[q]$. Draw the resulting flow graph as a butterfly.

(b) Starting with the output nodes in Figure 8.13 replace each DIF-FFT butterfly by the one created in part (a) above using the appropriate values of p, q, and r. The result should be a flow graph from $X[k]$ in bit-reversed order to $x[n]$ in normal order. Verify that this flow graph computes the inverse DFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-nk}.$$

(c) Modify twiddle factors and gain values in the flow graph of (b) so that it computes the DFT $X[k]$ from $x[n]$. Compare this modified flow graph with the DIT-FFT flow graph in Figure 8.6 and comment on your observation.

40. Let the sequence $x[n]$ be of length L and we wish to compute K points of an N-point DFT of $x[n]$ where $L \ll N$ and $K \ll N$. Assume that the first $L = 2$ signal values $x[0]$ and $x[1]$ are nonzero. Furthermore, we wish to compute only the first $K = 4$ DFT values, that is, $X[0]$ through $X[3]$.

   (a) Draw a radix-2 $N = 16$-point DIT-FFT flow-chart in which only those paths originating from the nonzero signal values and terminating on the first K DFT values are retained.

   (b) Draw an $N = 16$-point DIF-FFT flow-chart in which only those paths originating from the nonzero signal values and terminating on the first K DFT values are retained.

   (c) Determine the total number of complex multiplications in each of the above flow graphs. Which algorithm gives the lesser number of multiplications?

   (d) Develop a general rule in terms of L, K, and N for selecting a DIT- or DIF-FFT algorithm in FFT *input/output pruning*.

41. The 2D-DFT $X[k, \ell]$ of size $M \times N$ image $x[m, n]$ was introduced in (7.210).

   (a) Express (7.210) so that $X[k, \ell]$ can be computed using an N-point 1D-DFT of its rows followed by an M-point 1D-DFT of its columns. This expression should look similar to the one in (8.50).

   (b) A close comparison of your expression in (a) with (8.50) should indicate that the twiddle-factor multiplication explained in (8.54) is missing. Explain why this step was necessary in the algorithm of (8.53) through (8.55).

42. Consider the general FFT algorithm given in Section 8.5. Let $N_1 = 2$ and $N_2 = \frac{N}{2}$. Show that equations (8.52), (8.54), and (8.55) lead to the DIT-FFT algorithm of (8.38)–(8.40).

43. The `gafft` function in Figure 8.15 computes one sample of the DFT.

   (a) Modify this function and write a new MATLAB function `X = gafft_vec(x,N,k)` that computes DFT values at the indices given in the vector `k`.

   (b) Verify your function using $x[n] = \cos(0.5\pi n)$, $0 \le n \le N - 1$ for values of $N = 8$, 16, and 32 and various indices in k. For verification use the `fft` function.

44. Let $x_c(t) = \cos(20\pi t) + \cos(18\pi t) + \cos(22\pi t)$. It is sampled at a rate of 40 Hz to obtain 128 samples in $x[n]$, $0 \le n \le 127$.

   (a) Take a 128-point FFT of $x[n]$ and plot the magnitude spectra over $8 \le F \le 12$ Hz.

   (b) To improve clarity in the spectral plot, take a 1024-point FFT of $x[n]$ and plot the magnitude spectra over $8 \le F \le 12$ Hz. Compare this plot with the above plot.

(c) Now use the `zfa` function and choose parameters to display the magnitude spectra over $8 \leq F \leq 12$ Hz. Compare this plot with the above two plots.

(d) In your opinion, which approach has the smallest number of computations with a better display?

Review problems

45. In this problem we will investigate differences in the speeds of DFT and FFT algorithms when stored twiddle factors are used.

(a) Write a function `W = dft_matrix(N)` that computes the DFT matrix W_N given in (8.8).

(b) Write a function `X = dftdirect_m(x,W)` that modifies the `dftdirect` function given on page 436 using the matrix `W` from (a). Using the `tic` and `toc` functions compare computation times for the `dftdirect` and `dftdirect_m` function for $N = 128, 256, 512, 1024$, and 2048. For this purpose generate an N-point complex-valued signal as `x = randn(1,N) + 1j*randn(1,N)`.

(c) Write a function `X = fftrecur_m(x,W)` that modifies the `fftrecur` function given on page 439 using the matrix `W` from (a). Using the `tic` and `toc` functions compare computation times for the `fftrecur` and `fftrecur_m` function for $N = 128, 256, 512, 1024$, and 2048. For this purpose generate an N-point complex-valued signal as `x = randn(1,N) + 1j*randn(1,N)`.

46. The FFT algorithm can be used to perform fast interpolation of data values which are equispaced in time. Let $x(t)$ be a function defined over a $0 \leq t \leq T$ interval. Then from the Fourier analysis we know that it can be approximated by the CTFS

$$
x(t) \approx x_N(t) = \sum_{k=-N}^{N} c_k e^{2\pi kt/T},
$$

which has $2N + 1$ CTFS coefficients $\{c_k\}$. Let us assume that we have available data values $x[n] \triangleq x(t_n) = x_N(t_n)$ at $(2N + 1)$ equispaced instances $t_n = [T/(2N + 1)]n$, $n = 0, 1, \ldots, 2N$ and we want to interpolate the signal $x(t)$ using these data values.

(a) Show that the CTFS coefficients $\{c_k\}$ can be interpreted as $2N + 1$-point DFT coefficients of the data values.

(b) Develop a FFT-based interpolation scheme that uses the above interpretation.

(c) Let $x(t) = e^{\sin(t)+\cos(t)}$, $0 \leq t \leq 2\pi$. Using $N = 2$, that is five data values, obtain and plot the interpolation $x_2(t)$ and compare it with $x(t)$.

(d) Repeat part (c) for $N = 4$ and comment on your results.

47. As discussed in the chapter, the SR-FFT algorithm implements a radix-2 approach for the even-indexed terms $X[2k]$ but uses a radix-4 approach for the odd-indexed terms $X[2k + 1]$.

(a) Show that the $X[2k]$ terms can be expressed as the $\frac{N}{2}$-point DFT

$$X[2k] = \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + x\left[n + \frac{N}{2}\right] \right) W_{N/2}^{kn} \qquad (8.90)$$

for $k = 0, 1, \ldots, \frac{N}{2} - 1$.

(b) Show that the odd-indexed terms of $X[k]$ can be expressed as the $\frac{N}{4}$-point DFTs

$$X[4k + 1] = \sum_{n=0}^{\frac{N}{4}-1} \left\{ \left(x[n] - x\left[n + \frac{N}{2}\right] \right) - j\left(x\left[n + \frac{N}{4}\right] - x\left[n + \frac{3N}{4}\right] \right) \right\}$$
$$\times W_N^n W_{N/4}^{kn}, \qquad (8.91a)$$

$$X[4k + 3] = \sum_{n=0}^{\frac{N}{4}-1} \left\{ \left(x[n] - x\left[n + \frac{N}{2}\right] \right) + j\left(x\left[n + \frac{N}{4}\right] - x\left[n + \frac{3N}{4}\right] \right) \right\}$$
$$\times W_N^{3n} W_{N/4}^{kn}. \qquad (8.91b)$$

(c) Draw a flow graph of the SR-FFT algorithm for $N = 16$ using the equations developed in parts (a) and (b). Indicate all appropriate multiplicative constants.

(d) Determine the total number of multiplications needed to implement the flow graph in part (c) and compare it with those for a radix-2 DIF-FFT algorithm.

48. Dual-tone multifrequency (DTMF) is a generic name for a push-button signaling scheme used in Touch-Tone and telephone banking systems in which a combination of high-frequency tone and a low-frequency tone represent a specific digit or the characters "∗" and "#." In one version of this scheme, there are seven frequencies arranged as shown below, to accommodate 12 characters. The system operates at a sampling rate of 8 kHz.

	Col 1: 1209 Hz	Col 2: 1336 Hz	Col 3: 1477 Hz	Col 4: 1633 Hz
Row 1: 697 Hz	1	2	3	A
Row 2: 770 Hz	4	5	6	B
Row 3: 852 Hz	7	8	9	C
Row 4: 941 Hz	∗	0	#	D

DTMF digit = row tone + column tone

(a) Write a MATLAB function x = sym2TT(S) that generates samples of high and low frequencies of one-half second duration, given a symbol S according to the above expression.

(b) Modify the `gafft` function on page and write a new MATLAB function `X = gafft_vec(x,N,k)` that computes DFT values at the indices given in the vector `k`.

(c) Using the above modified function now develop a MATLAB function `S = TT2sym(x)` that detects the symbol `S` given the touch-tone signal in `x`.

9 Structures for discrete-time systems

As we discussed in Chapter 2, any LTI can be implemented using three basic computational elements: adders, multipliers, and unit delays. For LTI systems with a rational system function, the relation between the input and output sequences satisfies a linear constant-coefficient difference equation. Such systems are practically realizable because they require a finite number of computational elements. In this chapter, we show that there is a large collection of difference equations corresponding to the same system function. Each set of equations describes the same input-output relation and provides an algorithm or structure for the implementation of the system. Alternative structures for the same system differ in computational complexity, memory, and behavior when we use finite precision arithmetic. In this chapter, we discuss the most widely used discrete-time structures and their implementation using MATLAB. These include direct-form, transposed-form, cascade, parallel, frequency sampling, and lattice structures.

Study objectives

After studying this chapter you should be able to:

- Develop and analyze practically useful structures for both FIR and IIR systems.
- Understand the advantages and disadvantages of different filter structures and convert from one structure to another.
- Implement a filter using a particular structure and understand how to simulate and verify the correct operation of that structure in MATLAB.

9.1   Block diagrams and signal flow graphs

Every practically realizable LTI system can be described by a set of difference equations, which constitute a computational algorithm for its implementation. Basically, a computational or implementation *structure* for a discrete-time system is a pictorial block diagram representation of the computational algorithm using delays, adders, and multipliers. A system structure serves as a basis for the following tasks:

- Development of *software* (that is, a program) that implements the system on a general purpose computer or a special purpose digital signal processor.
- Design and implementation of a *hardware* architecture that can be used to implement the system using discrete components or VLSI technology.

In Section 2.3.3 we introduced three basic computational elements that are used in implementing a practically realizable discrete-time system. The addition element is used to sum two or more sequences, the multiply element is used to scale a sequence by a constant value, and the unit-delay element is used to delay (or shift to the right) a sequence by one sample. These elements are shown in Figure 2.6 as block diagram elements in the left column and as signal flow graphs in the right column.

To illustrate these concepts, we consider a first-order IIR system described by the difference equation

$$y[n] = b_0 x[n] + b_1 x[n-1] - a_1 y[n-1]. \tag{9.1}$$

The system function is given by

$$H(z) = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}. \tag{9.2}$$

Figure 9.1 shows both the block diagram and signal flow graph implementations of the difference equation (9.1). The block diagram shows the operations described by the difference equation using basic computational elements while the signal flow graph provides an equivalent graphical representation of the flow of signals and of their operations. Note that, in the case of the signal flow graph, the input branch applies the external signal $x[n]$ to the system at the left branch node and the output of the signal is available at the output branch connected to the right branch node. Furthermore, signal flow graphs not only provide a

(a)   (b)
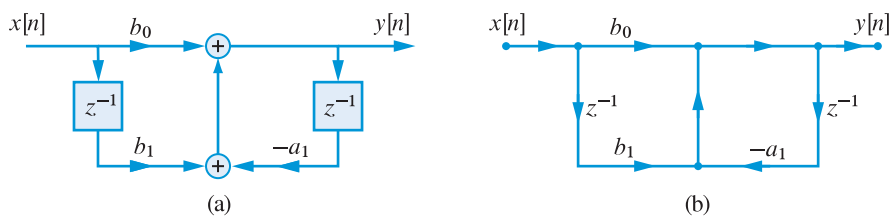
Figure 9.1 Structure for the first-order IIR system in (9.1): (a) block diagram, (b) signal flow graph (normal form).

9.1 Block diagrams and signal flow graphs
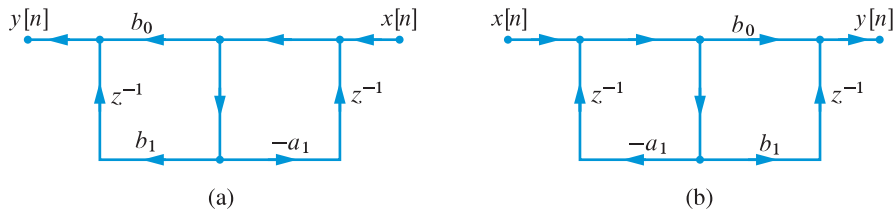
Figure 9.2 Structure for the first-order IIR system in (9.1): (a) signal flow graph after application of the transposition procedure, (b) signal flow graph after flipping (transposed form).

graphical representation but can also be manipulated in a way similar to mathematical equations to obtain equivalent alternative structures.

Transposition of linear flow graphs The objective of this chapter is to develop alternative structures for the same system function. One approach in achieving this is through manipulation of signal flow graphs using a procedure known as *transposition*. The resulting flow graph is termed as the *transposed form* while the original is called the *normal form*. Transposed structures can be derived using the *transposition theorem* for flow graphs. According to this theorem we can derive an equivalent structure from a given realization by the following set of operations:

1. reverse all branch directions;
2. replace branch nodes by summing nodes and vice versa; and
3. interchange the input and output nodes.

Applying the above three steps to the signal flow graph in Figure 9.1, we obtain the diagram in Figure 9.2(a). Since the customary approach is to show the input branch on the left and the output one on the right, we flip the diagram in (a) to obtain the resulting transposed form in Figure 9.2(b). To verify that the new structure represents the same system function in (9.2), denote the center node signal by $v[n]$. Then at the left summing node we have

$$v[n] = x[n] - a_1 v[n-1] \quad \text{or} \quad V(z) = \frac{1}{1 + a_1 z^{-1}} X(z), \qquad (9.3)$$

and at the right summing node we have

$$y[n] = b_0 v[n] + b_1 v[n-1] \quad \text{or} \quad Y(z) = (b_0 + b_1 z^{-1}) V(z). \qquad (9.4)$$

Combining (9.3) and (9.4), the system function is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{Y(z)}{V(z)} \frac{V(z)}{X(z)} = \frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}}, \qquad (9.5)$$

which is same one as in (9.2). Thus the normal form in Figure 9.1(b) is equivalent to the transposed form in Figure 9.2(b). For a proof of the transposition theorem using Tellegen's theorem for flow graphs see Claasen and Mecklenbräuker (1978) or Crochiere and Rabiner (1983); for a proof using a state-space approach see Jackson (1970a, 1996).

Alternative structures for the same system differ in *computational complexity*, *memory*, and behavior when we use *finite precision arithmetic*. In this chapter we use the system function and several of its manipulations to obtain different structures for the implementation of IIR and FIR systems.

9.2   IIR system structures

We begin with the development of various structures for IIR systems having rational system functions with real coefficients. We will consider three useful structures: a direct form which is obtained *directly* from the system function and has two variations; a cascade form which is obtained by expressing the system function as a *product* of second-order sections; and a parallel form which is obtained by expressing the system function as a *sum* of second-order sections. Each of these structures also has its transposed form variations. FIR systems can be considered as a special case of IIR systems with only numerator polynomials. However, because of their importance we treat FIR systems separately in Section 9.3.

9.2.1   Direct form structures

These structures are the easiest to obtain given a system function or difference equation. Consider an Nth-order causal system described by the following difference equation

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]. \tag{9.6}$$

The corresponding system function is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\displaystyle\sum_{k=0}^{M} b_k z^{-k}}{1 + \displaystyle\sum_{k=1}^{N} a_k z^{-k}}. \tag{9.7}$$

Direct form I A simple inspection of the difference equation (9.6) leads to the structure shown in Figure 9.3 for $N = M = 2$, which is a straightforward implementation of the sum of products in the difference equation. This structure is called a *direct form I* structure because it can be written directly from the system function or the difference equation by simple inspection; in other words the coefficients $\{a_k, b_k\}$ are used "directly" to form the structure (note that, given a system function, coefficients $\{a_k\}$ are entered with a negative value). This structure requires $(M + N)$ delay elements, $(M + N + 1)$ multiplications, and $(M + N)$ additions.

The structure of Figure 9.3 is the cascade connection of two systems. The first is a nonrecursive system with difference equation

9.2 IIR system structures
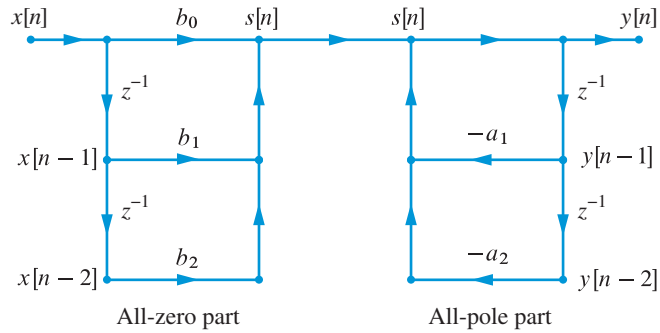
Figure 9.3 Direct form I structure for implementation of an Nth order IIR system with $N = M = 2$.

$$s[n] = \sum_{k=0}^{M} b_k x[n - k],\tag{9.8}$$

and system function

$$H_1(z) = \frac{S(z)}{X(z)} = \sum_{k=0}^{M} b_k z^{-k}. \quad \text{(all-zero)}\tag{9.9}$$

The second is a recursive system with difference equation

$$y[n] = -\sum_{k=1}^{N} a_k y[n - k] + s[n],\tag{9.10}$$

and system function

$$H_2(z) = \frac{Y(z)}{S(z)} = \frac{1}{1 + \sum_{k=1}^{N} a_k z^{-k}}. \quad \text{(all-pole)}\tag{9.11}$$

The overall system function, obtained by implementing first the all-zero part and then the all-pole part, is given by

$$H(z) = \frac{Y(z)}{X(z)} = \frac{Y(z)}{S(z)} \frac{S(z)}{X(z)} = H_2(z)H_1(z).\tag{9.12}$$

The steps given in (9.8) and (9.10) are implemented in the MATLAB function `y=filterdf1(b,a,x)` shown in Figure 9.4. It assumes that the initial conditions on both $x[n]$ and $y[n]$ are zero. This function is not necessarily the best or the most efficient but is given for educational purposes only. Tutorial Problem 4 extends this function to include arbitrary initial conditions on $x[n]$ and $y[n]$.

```
function [y] = filterdf1(b,a,x)
% Implementation of Direct Form I structure (Normal Form)
%  with zero initial conditions
% [y] = filterdf1(b,a,x)
M = length(b)-1; N = length(a)-1; K = max(M,N);
a0 = a(1); a = reshape(a,1,N+1)/a0;
b = reshape(b,1,M+1)/a0; a = a(2:end);
Lx = length(x); x = [zeros(K,1);x(:)];
Ly = Lx+K; y = zeros(1,Ly);
for n = K+1:Ly
    sn = b*x(n:-1:n-M);
    y(n) = sn - a*y(n-1:-1:n-N);
end
y = y(K+1:Ly);
```

Figure 9.4 MATLAB function for the direct form I structure.
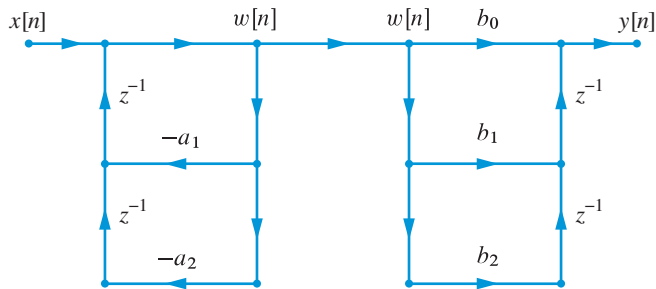
Figure 9.5 Transposed direct form I structure for the implementation of an N-th order system with $N = M = 2$.

Transposed direct form I structure The structure given in Figure 9.3 is in the normal form. Using the transposition theorem (see Section 9.1) we obtain the structure shown in Figure 9.5. It is a simple matter to verify that the structure shown in Figure 9.5 can be implemented by the following set of difference equations:

$$w[n] = -\sum_{k=1}^{N} a_k w[n-k] + x[n], \tag{9.13a}$$

$$y[n] = \sum_{k=0}^{M} b_k w[n-k]. \tag{9.13b}$$

Tutorial Problem 5 explores a MATLAB implementation of (9.13) to simulate the transposed direct form I structure.

Direct form II Since, in theory, the order of the interconnected systems does not affect the overall system function, we can equivalently express (9.12) as

$$H(z) = H_1(z)H_2(z),\tag{9.14}$$

that is, we first implement the recursive part (poles) and then the nonrecursive part (zeros). In this case, we have

$$Y(z) = H_1(z)H_2(z)X(z) = H_1(z)W(z),\tag{9.15}$$

where

$$W(z) = H_2(z)X(z) = \frac{1}{1 + \sum\limits_{k=1}^{N} a_k z^{-k}} X(z).\tag{9.16}$$

Cross-multiplying the terms in (9.16) we obtain

$$W(z) + \sum_{k=1}^{N} a_k z^{-k} W(z) = X(z),\tag{9.17}$$

which easily leads to the following difference equation

$$w[n] = -\sum_{k=1}^{N} a_k w[n-k] + x[n].\tag{9.18}$$

The output of the overall system is

$$Y(z) = \sum_{k=0}^{M} b_k z^{-k} W(z),\tag{9.19}$$

or equivalently

$$y[n] = \sum_{k=0}^{M} b_k w[n-k].\tag{9.20}$$

The structure specified by difference equations (9.18) and (9.20), known as a *direct form II* structure, is shown in Figure 9.6 for $N = M = 2$. This structure is in the *normal* form and is also called the *canonical direct form* because it requires the minimum possible number of delays, which is given by $\max(N, M)$. We emphasize that although the direct form I and direct form II structures are theoretically equivalent, there can be differences in their outputs when implemented using finite-precision arithmetic. For the implementation of the normal direct form II structure of (9.18) and (9.20) see Problem 18.
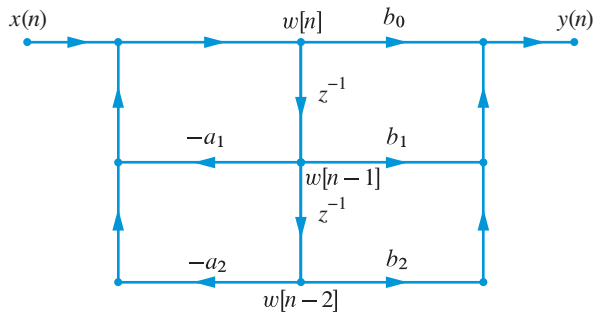
Figure 9.6 Direct form II structure for implementation of an Nth order system. For convenience, we assume that $N = M = 2$. If $N \neq M$, some of the coefficients will be zero.
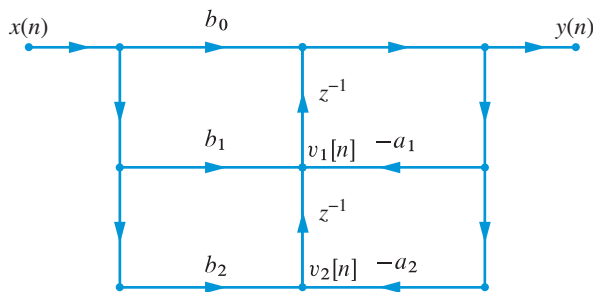
Figure 9.7 Transposed direct form II structure for realization of an Nth order system with $N = 2$.

Transposed direct form II structure The most widely used structure is the *transposed direct form II*, which is obtained by transposing the direct form II structure. Application of the transposition theorem yields the structure shown in Figure 9.7. For simplicity, we assume that $N = M$; otherwise, we set $N = \max(M, N)$. The key aspects of this structure are illustrated in the following example.

Example 9.1

From a simple inspection of the flow graph in Figure 9.7, we obtain the following difference equations:

$$y[n] = v_1[n-1] + b_0 x[n], \tag{9.21a}$$

$$v_1[n] = v_2[n-1] - a_1 y[n] + b_1 x[n], \tag{9.21b}$$

$$v_2[n] = b_2 x[n] - a_2 y[n]. \tag{9.21c}$$

We note that to update the internal variables $v_k[n]$ we need the present values of the input $x[n]$ and the output $y[n]$. For this reason, we first compute the output $y[n]$; the updating of $v_k[n]$ can be done in any order. Taking the z-transform converts each difference equation into an algebraic equation

$$Y(z) = z^{-1}V_1(z) + b_0X(z), \tag{9.22a}$$

$$V_1(z) = z^{-1}V_2(z) - a_1Y(z) + b_1X(z), \tag{9.22b}$$

$$V_2(z) = b_2X(z) - a_2Y(z). \tag{9.22c}$$

Substituting (9.22c) into (9.22b), and the result into (9.22a) we obtain

$$Y(z) = b_0X(z) + b_1z^{-1}X(z) + b_2z^{-2}X(z) - a_1z^{-1}Y(z) - a_2z^{-2}Y(z).$$

This corresponds to a second-order system with system function

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{1 + a_1z^{-1} + a_2z^{-2}},$$

which is identical to the system function of the structure in Figure 9.6.   ∎

It is a simple matter to verify that the general transposed direct form II structure shown in Figure 9.7 can be implemented by the following set of difference equations:

$$y[n] = v_1[n-1] + b_0x[n], \tag{9.23a}$$

$$v_k[n] = v_{k+1}[n-1] - a_ky[n] + b_kx[n], \quad k = 1,\ldots,N-1 \tag{9.23b}$$

$$v_N[n] = b_Nx[n] - a_Ny[n], \tag{9.23c}$$

where the initial conditions now are on the "internal" variables $v_k[-1]$, $k = 1,\ldots,N$ instead of on the input and output signals. Typically we set $v_k[-1] = 0, k = 1,\ldots,N$.

The MATLAB function `y=filterdf2t(b,a,x,v)` shown in Figure 9.8 implements (9.23). If the fourth input argument `v` is omitted then initial conditions are assumed to be zero. The transposed direct form II structure is also used in the implementation of MATLAB's built-in function `y=filter(b,a,x,v)`. Tutorial Problem 6 explores a MATLAB function to convert direct form I initial conditions $x[-1],\ldots,x[-M]$ and $y[-1],\ldots,y[-N]$ to direct form II initial conditions $v_1[-1],\ldots,v_{\max\{M,N\}}$. The corresponding built-in function in MATLAB is `v=filtic(b,a,yic,xic)`.

Example 9.2   IIR direct form structures

Consider an IIR system with system function

$$H(z) = \frac{10 + z^{-1} + 0.9z^{-2} + 0.8z^{-3} - 5.8z^{-4}}{1 - 2.54z^{-1} + 3.24z^{-2} - 2.06z^{-3} + 0.66z^{-4}}. \tag{9.24}$$

Since the coefficients in the system function correspond directly to the branch values in the direct form structures, it is easy to draw these structures by inspection. Figure 9.9 shows all four direct form structures. It is interesting to observe that normal direct form I and transposed direct form II (or transposed direct form I and normal direct form II) appear to be similar in the placement of coefficients and signal flow directions; however, the signal flow graphs are not exactly the same.   ∎

```
function y=filterdf2t(b,a,x,v)
% Implementation of Direct Form II structure (Transposed)
%  with arbitrary initial conditions
% [y] = filterdf2t(b,a,x,v)

N=length(a)-1; M=length(b)-1; K=max(N,M);
L=length(x); y=zeros(L,1);

if nargin < 4, v=zeros(K,1); end
if N>M, b=[b' zeros(1,N-M)]'; else
a=[a' zeros(1,M-N)]'; end

for n=1:L
   y(n)=v(1)+b(1)*x(n);
   for k=1:K-1
      v(k)=v(k+1)-a(k+1)*y(n)+b(k+1)*x(n);
   end
   v(K)=b(K+1)*x(n)-a(K+1)*y(n);
end
```

Figure 9.8 MATLAB function for the transposed direct form II structure.

The direct form structures were obtained by "direct" inspection of the rational system function (9.2). Using the pole-zero and partial-fraction expansion representations of $H(z)$ we obtain alternative structures called the cascade and parallel structures.

9.2.2   Cascade form structures

We recall from (3.83) that any rational system function, as in (9.7), with real coefficients can be expressed in pole-zero form as follows:

$$H(z) = b_0 \frac{\displaystyle\prod_{k=1}^{M_1}\left(1 - z_k z^{-1}\right) \prod_{k=1}^{M_2}\left(1 - z_k z^{-1}\right)\left(1 - z_k^* z^{-1}\right)}{\displaystyle\prod_{k=1}^{N_1}\left(1 - p_k z^{-1}\right) \prod_{k=1}^{N_2}\left(1 - p_k z^{-1}\right)\left(1 - p_k^* z^{-1}\right)}, \tag{9.25}$$

where $M = M_1 + 2M_2$ and $N = N_1 + 2N_2$. Complex conjugate pairs of poles (zeros) can be combined into second-order terms with real coefficients. If N_1 or M_1 are odd numbers then we include a pole or zero at $z = 0$, respectively, to make them even numbers.

(a) Normal direct form I

(b) Transposed direct form I

(c) Normal direct form II

(d) Transposed direct form II
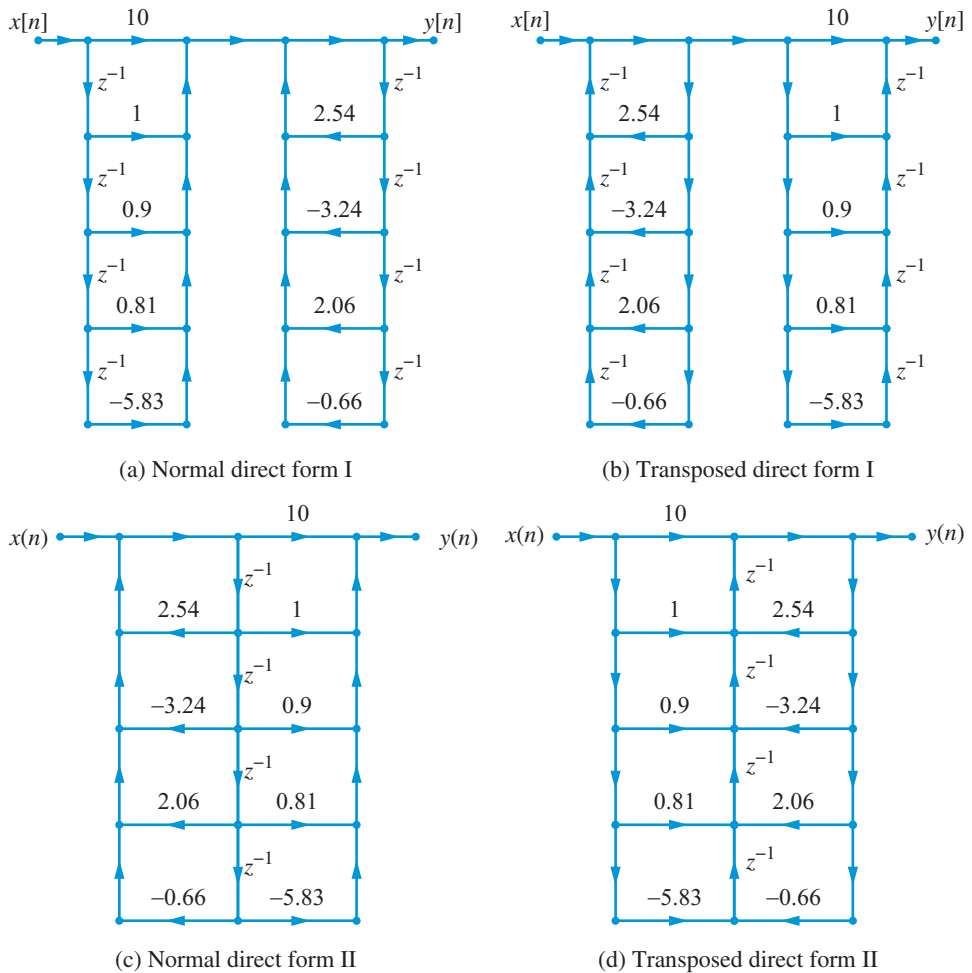
Figure 9.9 Direct form structures for the system in Example 9.2

Now we can combine pairs of real poles and real zeros into second-order terms. Assuming, for simplicity that $N = M$, we can express (9.25) as

$$H(z) \triangleq G \prod_{k=1}^{K} \frac{B_{k0} + B_{k1}z^{-1} + B_{k2}z^{-2}}{1 + A_{k1}z^{-1} + A_{k2}z^{-2}} \triangleq G \prod_{k=1}^{K} H_k(z), \tag{9.26}$$

where G is the overall system gain, $H_k(z)$ are the second-order sections with arbitrary gains such that $b_0 = G \prod_{k=1}^{K} B_{k0}$, and $N = M = 2K$. If N is odd, one of the B_{k2} coefficients and one of the A_{k2} coefficients will be zero. The additional complexity due to two extra coefficients is outweighed by the modular structure of (9.26). Indeed, the decomposition in (9.26) allows us to use the same function or hardware module to implement the higher-order system. The resulting cascade structure is illustrated as a block diagram in Figure 9.10.
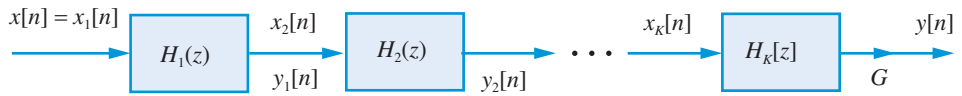
Figure 9.10 Structure of cascade form for the realization of an Nth order system.
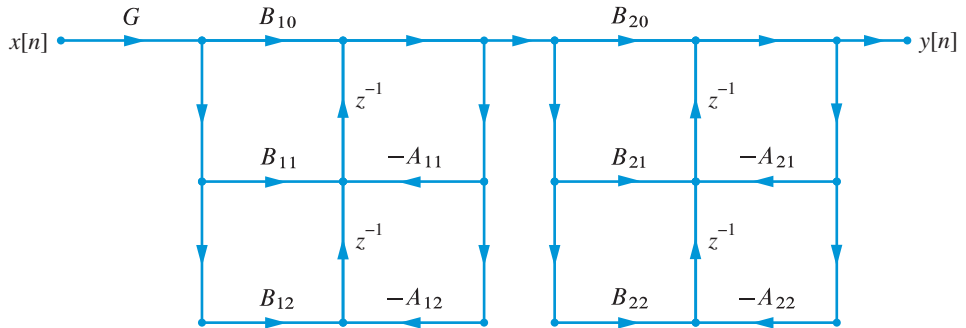
Figure 9.11 Cascade structure for $M = N = 2$ using transposed direct form II second-order sections.

The second-order $\{H_k(z)\}$ sections in (9.26) can be implemented using either the normal direct form II or the transposed direct form II structures. Figure 9.11 shows a cascade structure for $M = N = 4$ using transposed direct form II second-order sections. The pairing of poles and zeros and the ordering of the resulting second-order sections can be done in many different ways. The resulting systems are theoretically equivalent when we use infinite precision arithmetic; however, the behavior may significantly change with finite precision arithmetic (see Section 15.4).

Finally, we note that using the identity

$$B_{k0} + B_{k1}z^{-1} + B_{k2}z^{-2} = B_{k0}(1 + \tilde{B}_{k1}z^{-1} + \tilde{B}_{k2}z^{-2}), \qquad (9.27)$$

and incorporating the factors B_{k0} into G in (9.26), we can avoid one multiplication per section. However, this approach is not recommended because we cannot individually scale input to each second-order section, as is needed for fixed-point implementation as we shall see in Section 15.4.

Given the coefficients $\{b_k\}$ and $\{a_k\}$ of the direct form system function $H(z)$, the MAT-LAB function `[sos,G]=tf2sos(b,a)` from the SP toolbox computes the overall gain G in `G` and the second-order section coefficients in the matrix `sos` which is a $K \times 6$ matrix whose kth row contains the numerator and denominator coefficients $\{B_{k,\ell}\}$, and $\{A_{k,\ell}\}$, $k = 1, \ldots, K$, $\ell = 0, 1, 2$. If `sos=tf2sos(b,a)` is invoked then the overall gain G is applied to the first second-order section. Similarly, the `[b,a]=sos2tf(sos,G)` or `[b,a]=sos2tf(sos)` converts cascaded second-order sections to direct form system function coefficients.

The difference equations for the cascade form (with normal direct form II sections) are

$$y_0[n] = x[n], \tag{9.28a}$$

$$w_k[n] = -A_{k1}w_k[n-1] - A_{k2}w_k[n-2] + y_{k-1}[n], \quad k = 1, \ldots, K \tag{9.28b}$$

$$y_k[n] = B_{k0}w_k[n] + B_{k1}w_k[n-1] + B_{k2}w_k[n-2], \quad k = 1, \ldots, K \tag{9.28c}$$

$$y[n] = Gy_K[n]. \tag{9.28d}$$

Problem 20 explores a MATLAB function `y=filtercf(sos,G,x)` to simulate the cascade structure using steps given in (9.28). A similar SP toolbox function invoked by `y=sosfilt(sos,x)`, that incorporates the overall gain G in second-order section coefficients $\{B_{k,\ell}\}$, is available in MATLAB.

Example 9.3   IIR cascade form

Consider the IIR system given in Example 9.2 and repeated here for convenience:

$$H(z) = \frac{10 + z^{-1} + 0.9z^{-2} + 0.8z^{-3} - 5.8z^{-4}}{1 - 2.54z^{-1} + 3.24z^{-2} - 2.06z^{-3} + 0.66z^{-4}}. \tag{9.29}$$

We obtain the coefficients of the second-order sections and the scaling constant using the function `tf2sos`:

```
>> a = [1 -2.54 3.24 -2.06 0.66];
>> b = [10 1 0.9 0.81 -5.83];
>> [sos,G] = tf2sos(b,a)
sos =
    1.0000    0.1000   -0.7199    1.0000   -1.1786    0.7246
    1.0000   -0.0000    0.8099    1.0000   -1.3614    0.9109
G =
    10
```

Hence the system function for the cascade form is given by

$$H(z) = 10 \times \frac{1 + 0.1z^{-1} - 0.7199z^{-2}}{1 - 1.1786z^{-1} + 0.7246z^{-2}} \times \frac{1 + 0z^{-1} + 0.8099z^{-2}}{1 - 1.3614z^{-1} + 0.9109z^{-2}}, \tag{9.30}$$

which corresponds to the $\{\tilde{B}_{k,\ell}\}$ coefficients given in (9.27). Figure 9.12 shows the cascade form structure using transposed direct form II second-order sections. ∎

9.2.3   Parallel form structures

Parallel form structures are based on a partial fraction expansion representation of the system function and hence result in a *sum* of second-order sections as opposed to the
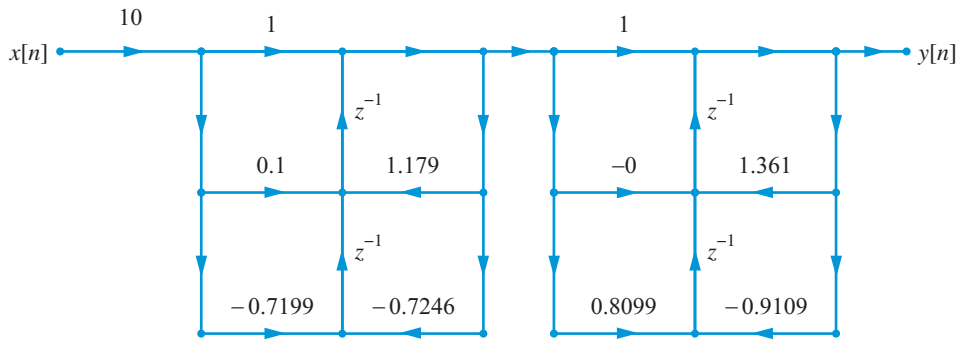
Figure 9.12 Cascade form structure in Example 9.3.
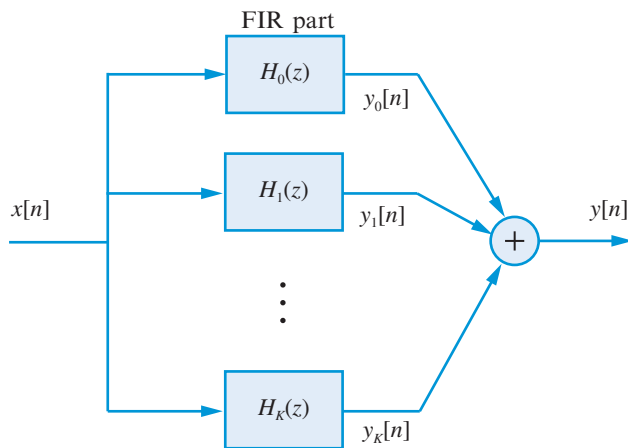
Figure 9.13 Parallel form structure of an Nth-order IIR system containing K second-order sections and the FIR part $H_0(z)$.

product of second-order sections as in the cascade form. Combining both real and complex conjugate pairs of partial factors in (3.42), we obtain

$$H(z) = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{K} \frac{B_{k0} + B_{k1} z^{-1}}{1 + A_{k1} z^{-1} + A_{k2} z^{-2}}. \qquad (9.31)$$

The first summation is not included if $M < N$. If the number of poles is odd, we add a pole at zero and we set $K = (N + 1)/2$. The implementation of (9.31) leads to a parallel interconnection of an FIR system $H_0(z)$ and K second-order systems $H_k(z)$, $k = 1, \ldots, K$ as shown as a block diagram in Figure 9.13.

The second-order sections in (9.31) are typically implemented using the transposed direct form II structure (see Example 9.4). Unlike the cascade form, the second-order sections are unique because the pairing of poles and their corresponding residues must be

consistent. However, the ordering of the resulting second-order sections can be done in many different ways. The resulting systems are theoretically equivalent when we use infinite precision arithmetic; however, the behavior can change with finite precision arithmetic (see Section 15.4).

MATLAB does not have a built-in function to determine the second-order sections of the parallel form in (9.31). However, it is easy to obtain these coefficients using the `residuez` function discussed in Section 3.3 along with some additional processing, as shown in the following example.

Example 9.4   IIR parallel form

Consider the IIR system given in Example 9.2:

$$H(z) = \frac{10 + z^{-1} + 0.9z^{-2} + 0.8z^{-3} - 5.8z^{-4}}{1 - 2.54z^{-1} + 3.24z^{-2} - 2.06z^{-3} + 0.66z^{-4}}. \tag{9.32}$$

The following MATLAB script shows how to determine the parameters of the parallel form. We first compute residues and poles and then combine complex-conjugate residue/pole pairs using the `residuez` function to obtain the second-order sections. The result is

```
>> b = [10 1 0.9 0.81 -5.83]; a = [1 -2.54 3.24 -2.06 0.66];
>> [R,p,C] = residuez(b,a); C
C =
   -8.8333
>> [B1,A1] = residuez(R(1:2),p(1:2),[]);
>> B1 = real(B1)
B1 =
   27.0624    89.7028
>> A1 = real(A1)
A1 =
    1.0000   -1.3614    0.9109
>> [B2,A2] = residuez(R(3:4),p(3:4),[]);
>> B2 = real(B2)
B2 =
   -8.2291   -90.4461
>> A2 = real(A2)
A2 =
    1.0000   -1.1786    0.7246
```

Thus, the system function for the parallel form is given by

$$H(z) = -8.83 + \frac{27.06 + 89.70z^{-1}}{1 - 1.36z^{-1} + 0.91z^{-2}} + \frac{-8.23 - 90.45z^{-1}}{1 - 1.18z^{-1} + 0.72z^{-2}}. \tag{9.33}$$

The resulting structure using transposed direct form II second-order sections is shown in Figure 9.14. Problem 21 explores a MATLAB function `tf2pf` that incorporates steps used
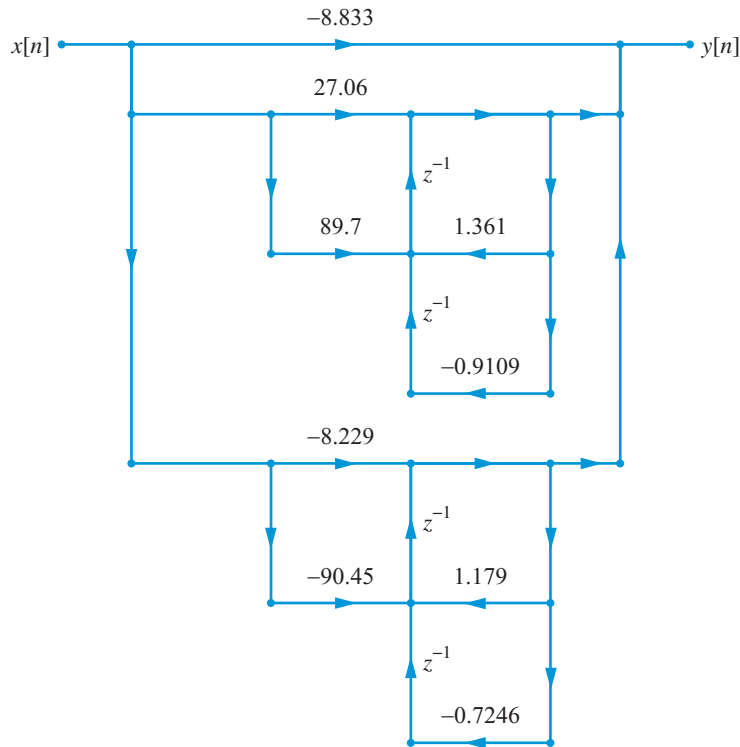
Figure 9.14 Parallel form structure with transposed direct form II sections in Example 9.4.

in this example to obtain parallel form coefficients as well as an inverse function `pf2tf` that converts the parallel form back to the direct form. ∎

Using a direct form II for the second-order sections (with $b_{k2} = 0$), leads to the following set of difference equations:

$$w_k[n] = -a_{k1}w_k[n-1] - a_{k2}w[n-2] + x[n], \qquad k = 1, \ldots, K \qquad (9.34a)$$

$$y_k[n] = b_{k0}w_k[n] + b_{k1}w_k[n-1], \qquad k = 1, \ldots, K \qquad (9.34b)$$

$$y[n] = \sum_{k=0}^{M-N} C_k x[n-k] + \sum_{k=1}^{K} y_k[n]. \qquad (9.34c)$$

Again it is easy to program the steps in (9.34) to obtain a MATLAB function that can simulate the parallel structure. Such a function, called `filterpf`, is explored in Problem 25. There is no similar SP toolbox function in MATLAB.

9.3 FIR system structures

Causal FIR systems, with real coefficients, are characterized by the moving-average difference equation

$$y[n] = \sum_{k=0}^{M} b_k x[n-k], \tag{9.35}$$

which leads to the finite-duration impulse response

$$h[n] = \begin{cases} b_n, & n = 0, 1, \dots, M \\ 0, & \text{otherwise} \end{cases} \tag{9.36}$$

or the all-zero (except for poles at $z = 0$) system function

$$H(z) = \frac{Y(z)}{X(z)} = \sum_{n=0}^{M} b_n z^{-n} = \sum_{n=0}^{M} h[n] z^{-n}. \tag{9.37}$$

The order of the filter in the above representations is M and the filter is always stable. We first consider direct form and cascade form structures which are similar to but simpler than IIR structures because they contain only feedforward paths. Since there is no partial fraction expansion of $H(z)$ in (9.36), we do not have a parallel structure of the IIR form. However, using the reconstruction formula (7.72) a parallel structure of second-order sections, called *frequency sampling form*, can be obtained. Furthermore, using symmetries in the impulse response, FIR filters can have an exact linear-phase response which leads to a structure called *linear-phase form*. Linear-phase filters are desirable in many applications.

9.3.1 Direct form

From (9.37) it is obvious that the system function has unity denominator. Therefore, both direct form I and II structures are the same and lead to the realization shown in Figure 9.15(a) for $M = 4$; it has a series of delay units that are tapped and the resulting values are linearly combined to obtain the output. Hence this structure is also known as a *tapped delay-line* or a *transversal line* structure. Using the transposition theorem of Section 9.1, a transposed direct form structure can be obtained which is shown in Figure 9.15(b).

In MATLAB a direct form structure can be implemented in many different ways. The `y=filter(b,1,x)` is one approach while `y = conv(b,x)` is another, both using built-in MATLAB functions. A direct implementation (9.35) is given in `y=filterfirdf(b,x)` and shown in Figure 9.16, which is a modification of the `filterdf1` function in Section 9.2.1.

(a)

(b)
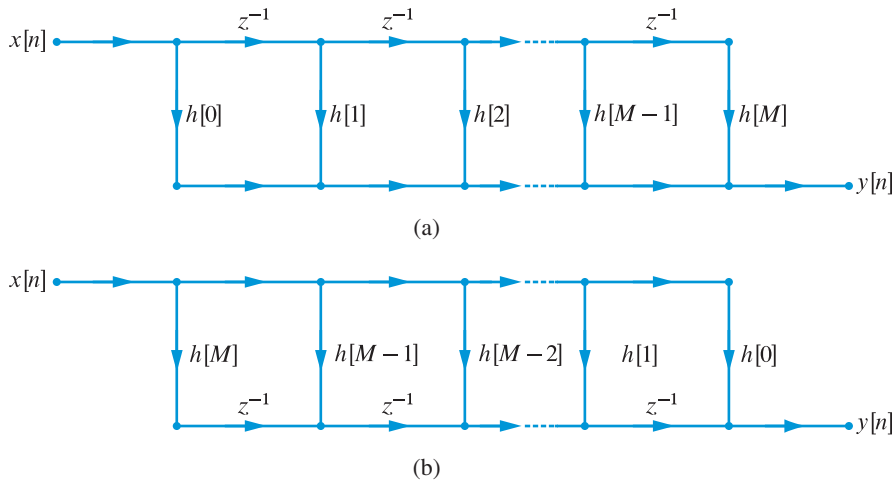
Figure 9.15 Direct form structure for the realization of an Mth-order FIR system: (a) normal form, (b) transposed form.

```
function [y] = filterfirdf(b,x)
% Implementation of FIR Direct Form structure (Normal Form)
% [y] = filterfirdf(b,a,x)
M = length(b)-1; b = reshape(b,1,M+1);
Lx = length(x); x = [zeros(K,1);x(:)];
Ly = Lx+M; y = zeros(1,Ly);
for n = K+1:Ly
     y(n) = b*x(n:-1:n-M);
end
y = y(K+1:Ly);
```

Figure 9.16 MATLAB function for the FIR direct form structure.

9.3.2   Cascade form

The system function $H(z)$ in (9.37) can be factored into second-order sections with real coefficients as

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n} \triangleq G \prod_{k=0}^{K} \left(1 + \tilde{B}_{k1}z^{-1} + \tilde{B}_{k2}z^{-2}\right), \tag{9.38}$$

where $M = 2K$. If M is odd, one of the $\tilde{B}_{k2}$ coefficients will be zero. The resulting second-order sections are connected in cascade wherein each second-order section can be implemented using normal or transposed form. Figure 9.17 shows a cascade form structure for $M = 6$ with transposed form sections.

The MATLAB function $[\text{sos},\text{G}]= \text{tf2sos}(\text{h},1)$ computes scaling factor G in G and the second-order section coefficients $\{B_{k,\ell}\}$, $\ell = 1, 2$ in matrix sos (which also contains
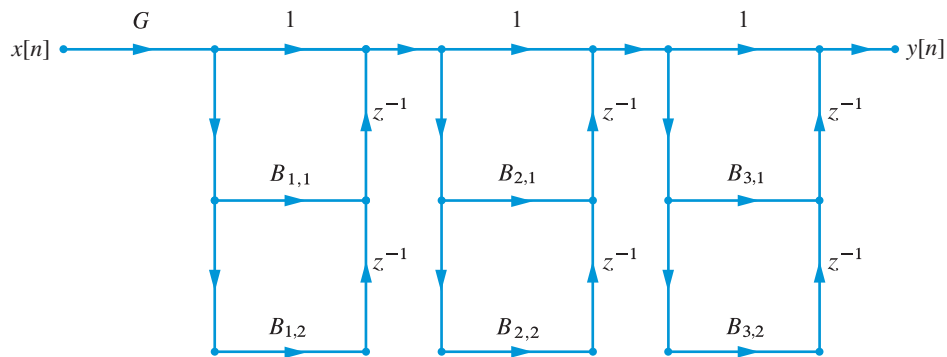
Figure 9.17 Cascade form structure for realization of a 6th-order FIR system.

the trivial denominator coefficients), given the system function $H(z)$ or impulse response $\{h[n]\}$. Similarly, the function `h=sos2tf(sos,G)` converts cascade form coefficients to impulse response coefficients.

The difference equations for the cascade form implementation are

$$y_0[n] = x[n], \tag{9.39a}$$

$$y_k[n] = y_{k-1}[n] + \sum_{\ell=1}^{2}\tilde{B}_{k\ell}y_{k-1}[n - \ell], \quad k = 1,\ldots,K \tag{9.39b}$$

$$y[n] = Gy_K[n]. \tag{9.39c}$$

The MATLAB function `y=sosfilt(sos,G,x)` can be used to implement the FIR cascade form, in which `sos` contains the appropriate FIR cascade coefficients $\{B_{k,\ell}\}$ and $\{A_{k,\ell}\}$. Another approach that uses only the necessary $\{B_{k,\ell}\}$ coefficients is explored in Problem 36.

9.3.3 Direct form for linear-phase FIR systems

In Section 5.3 we discussed the importance of linear phase in signal filtering operations. As we will see in Section 10.2, FIR systems with impulse responses satisfying the conditions

$$h[n] = \pm h[M - n], \quad 0 \leq n \leq M \tag{9.40}$$

have linear-phase response. When (9.40) is satisfied with the plus sign, the impulse response is termed as a symmetric response, while with the negative sign it is termed as an anti symmetric response.

Using these conditions on the impulse response, it is possible to obtain direct form structures that reduce the number of multiplications by almost half. There are four such possible (but similar) structures: M even or odd and $h[n]$ symmetric or antisymmetric. These systems are termed as type-I through type-IV systems, respectively (see Section 10.2
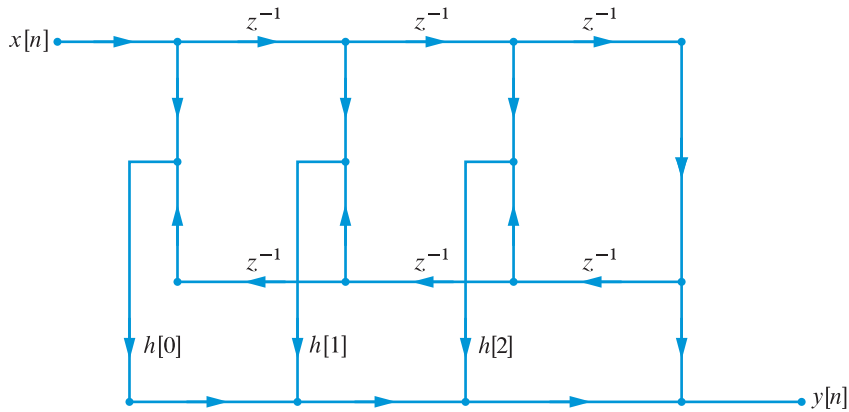
Figure 9.18 Linear-phase form structure for realization of a 6th-order type-I FIR system.

for more details). Consider first the type-I system for which M is an even integer and $h[n]$ is symmetric. Then the output can be manipulated as

$$
y[n] = \sum_{k=0}^{M} h[k]x[n-k]
$$

$$
= \sum_{k=0}^{\frac{M}{2}-1} h[k]x[n-k] + h\left[\frac{M}{2}\right]x\left[n-\frac{M}{2}\right] + \sum_{k=\frac{M}{2}+1}^{M} h[k]x[n-k]
$$

$$
= \sum_{k=0}^{\frac{M}{2}-1} h[k]x[n-k] + h\left[\frac{M}{2}\right]x\left[n-\frac{M}{2}\right] + \sum_{k=0}^{\frac{M}{2}-1} h[M-k]x[n-M+k]
$$

$$
= \sum_{k=0}^{\frac{M}{2}-1} h[k]\Big(x[n-k]+x[n-M+k]\Big) + h\left[\frac{M}{2}\right]x\left[n-\frac{M}{2}\right]. \tag{9.41}
$$

Thus the total number of multiplications is $\frac{M}{2}+1$ instead of $M+1$ for the standard direct form which is about 50% reduction. Figure 9.18 shows the structure suggested by (9.41) for $M=6$.

Similar manipulations for type-II systems when M is an odd integer and $h[n]$ is symmetric give

$$
y[n] = \sum_{k=0}^{\frac{M-1}{2}} h[k]\Big(x[n-k]+x[n-M+k]\Big). \tag{9.42}
$$

For type-III (even M and antisymmetric $h[n]$), the output is given by

$$
y[n] = \sum_{k=0}^{\frac{M}{2}-1} h[k]\Big(x[n-k]-x[n-M+k]\Big), \tag{9.43}
$$
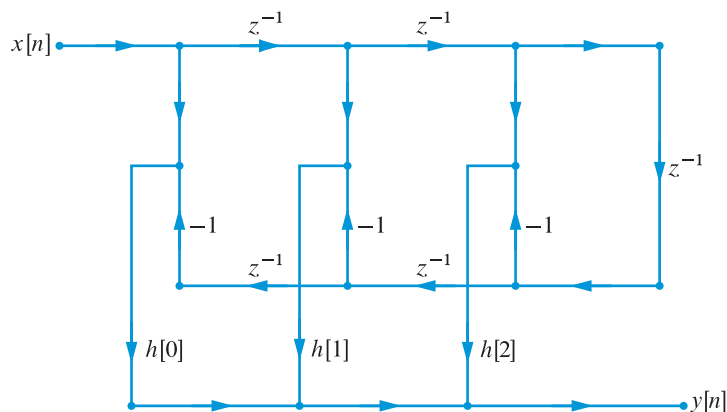
Figure 9.19 Linear-phase form structure for the realization of a 5th-order type-IV FIR system.

and finally for type-IV (odd M and antisymmetric $h[n]$), the output is given by

$$y[n] = \sum_{k=0}^{\frac{M-1}{2}} h[k]\Big(x[n-k] - x[n-M+k]\Big). \tag{9.44}$$

Figure 9.19 shows the linear-phase structure suggested by (9.44) for $M = 5$.

To implement the FIR linear-phase form in MATLAB we need to first determine the type of the system and then use one of the (9.41)–(9.44) equations. This can be accomplished by modifying the `filterfirdf` function and is explored in Tutorial Problem 7.

Example 9.5 FIR direct and cascade form structures

Consider the following system function $H(z)$ of a linear-phase FIR system:

$$H(z) = 5 - 10z^{-1} + 5z^{-2} - 20z^{-3} + 35z^{-4} - 20z^{-5} + 5z^{-6} - 10z^{-7} + 5z^{-8}. \tag{9.45}$$

We will determine the following structures and compare the amount of multiplication required to implement each of them.

Direct form: The difference equation from $H(z)$ in (9.45) is given by

$$y[n] = 5x[n] - 10x[n-1] + 5x[n-2] - 20x[n-3] + 35x[n-4]$$
$$- 20x[n-5] + 5x[n-6] - 10x[n-7] + 5x[n-8]. \tag{9.46}$$

The direct-form structure implied by (9.46) in the normal form is shown in Figure 9.20(a), while the transposed form is shown in Figure 9.20(b). Both structures require nine multiplications.
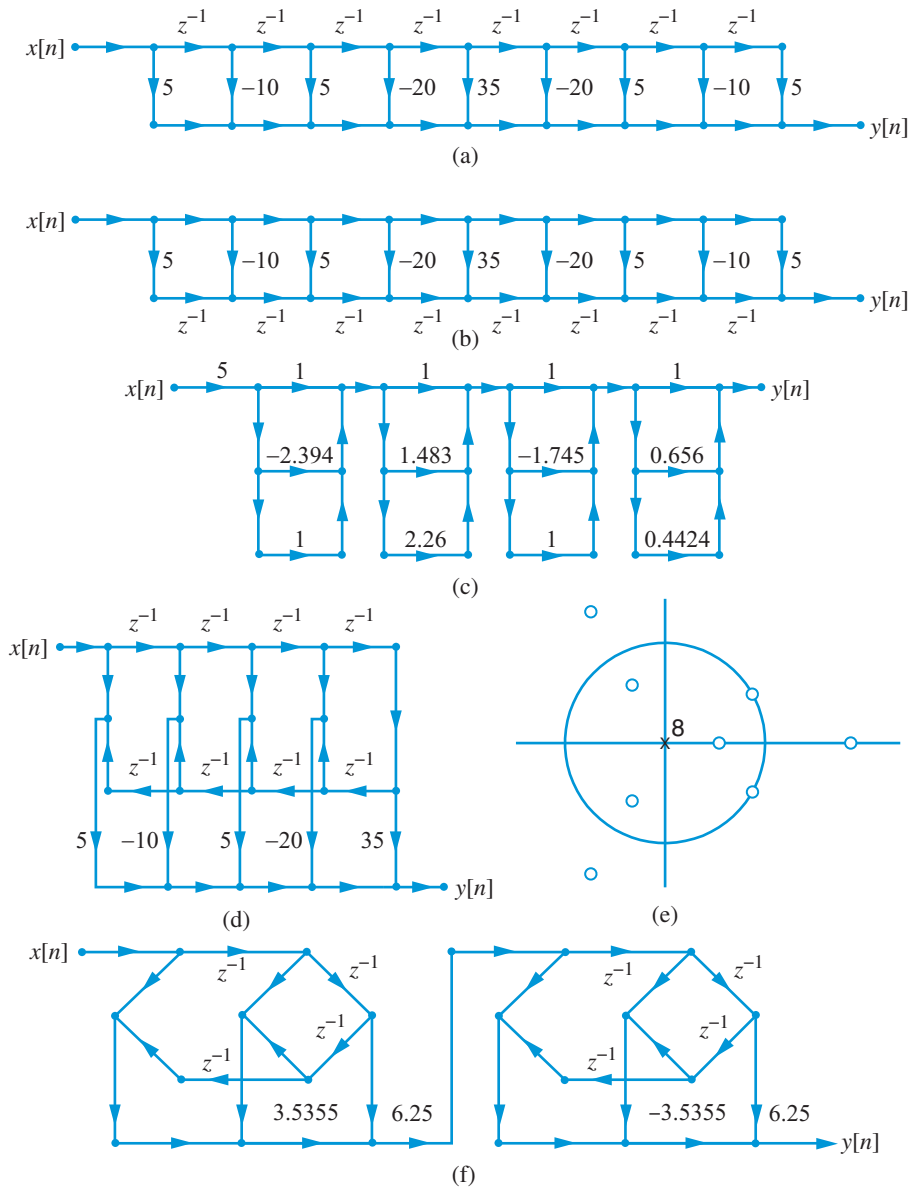
Figure 9.20 Structures for the FIR system in Example 9.5: (a) normal direct form, (b) transposed direct form, (c) normal cascade form, (d) linear-phase form, (e) pole-zero diagram, and (f) cascade of linear-phase sections.

Cascade form: The coefficients of the second-order sections required in the cascade form can be computed using the `tf2sos` function as follows:

```
>> [sos,G] = tf2sos(b,a)
>> sos =
```

```
       1.0000   -2.3940    1.0000    1.0000        0         0
       1.0000    1.4829    2.2604    1.0000        0         0
       1.0000   -1.7450    1.0000    1.0000        0         0
       1.0000    0.6560    0.4424    1.0000        0         0
G =
       5
```

Hence the system function in the cascade form is given by

$$H(z) = 5 \left(1 - 2.394z^{-1} + z^{-2}\right)\left(1 + 1.4829z^{-1} + 2.2604z^{-2}\right)$$
$$\times \left(1 - 1.745z^{-1} + z^{-2}\right)\left(1 + 0.656z^{-1} + 0.4424z^{-2}\right). \qquad (9.47)$$

The cascade-form structure implied by (9.47) in the transposed form is shown in Figure 9.20(c) which also requires nine multiplications.

Linear-phase form: The difference equation in (9.46) can be written as

$$y[n] = 5\big(x[n] + x[n - 8]\big) - 10\big(x[n - 1] + x[n - 7]\big)$$
$$+ 5\big(x[n - 2] + x[n - 6]\big) - 20\big(x[n - 3] + x[n - 5]\big) + 35x[n - 4]. \qquad (9.48)$$

Figure 9.20(d) shows the resulting linear-phase structure that requires five multiplications.

Cascade of linear-phase sections: To obtain this structure, we have to factorize $H(z)$ and then combine factors so that the resulting sections exhibit symmetry. These symmetry conditions cause zeros of $H(z)$ to have certain *mirror symmetry*. In Chapter 10 we show that if there is a zero of $H(z)$ at $z = z_0 = r_0 \angle \theta_0$ then for linear-phase response, there must be a zero at $1/z_0 = (1/r_0) \angle -\theta_0$. For real-valued systems, all zeros must occur in complex-conjugate pairs, which means that there must also be zeros at $z_0^* = r_0 \angle -\theta_0$ and at $1/z_0^* = (1/r_0) \angle \theta_0$. Thus as a general rule there must be a group of four zeros. One exception to this rule is when zeros are on a unit circle or on a real line, in which case they occur in a group of two. Finally, if a zero is on the unit circle *and* on the real line, that is, $z_0 = \pm 1$, then these zeros satisfy both the linear-phase and real-value constraints. In conclusion, after factorization we must combine the appropriate zero groups to obtain either a fourth-order, a second-order or a first-order section.

The zero-pole pattern of $H(z)$ is shown in Figure 9.20(e) from which we observe that there is one group of four zeros, one group of two zeros on the unit circle, and one group of two zeros on the real axis. We can combine two groups of two zeros to obtain a cascade of fourth-order sections. The following MATLAB script illustrates these steps:

```
>> B1 = conv(sos(1,1:3),sos(3,1:3))
B1 =
    1.0000   -4.1390    6.1775   -4.1390    1.0000
>> B2 = conv(sos(2,1:3),sos(4,1:3))
B2 =
    1.0000    2.1390    3.6757    2.1390    1.0000
```

The resulting system function is

$$H(z) = 5\left(1 - 4.139z^{-1} + 6.1775z^{-2} - 4.139z^{-3} + z^{-4}\right)$$
$$\times \left(1 + 2.139z^{-1} + 3.6757z^{-2} + 2.139z^{-3} + z^{-4}\right). \quad (9.49)$$

The cascade structure containing fourth-order linear-phase sections suggested by (9.49) is shown in Figure 9.20(f). This structure also needs five multiplications, however, its advantage over the general linear-phase form is in its modularity. ∎

9.3.4 Frequency-sampling form

This FIR system structure is different from others in that it is obtained using $(M + 1)$ equispaced samples $H[k]$ of its frequency response $H(e^{j\omega})$, hence the name frequency-sampling form. Recall from Section 7.3 that these samples are the $(M + 1)$-point DFT values of the impulse response $h[n]$ in (9.36). Furthermore, from (7.72) these DFT values are sufficient in order to reconstruct the system function $H(z)$. For convenience let the length of the impulse response be $N \triangleq M + 1$. Then from (7.72) we have

$$H(z) = \frac{1 - z^{-N}}{N} \sum_{k=0}^{N-1} \frac{H[k]}{1 - z^{-1}e^{j\frac{2\pi}{N}k}}, \quad H[k] = H(z)\big|_{z=e^{j2\pi k/N}}, \quad (9.50)$$

which suggests a parallel form structure that contains first-order recursive sections with complex coefficients. The system is still FIR because the N poles on the unit circle are cancelled by the N roots of the numerator polynomial $1 - z^{-N}$.

To avoid complex coefficients in the structure and the resultant complex arithmetic, we can use the symmetry properties of the DFT values and the conjugate symmetry of the poles $\{e^{j\frac{2\pi}{N}k}\}$. It can be easily shown that using these symmetries, the system function $H(z)$ in (9.50) is given by (see Tutorial Problem 13)

$$H(z) = \frac{1 - z^{-N}}{N}\left\{\frac{H[0]}{1 - z^{-1}} + \frac{H[\frac{N}{2}]}{1 + z^{-1}} + \sum_{k=1}^{K} 2|H[k]|H_k(z)\right\}, \quad (9.51)$$

where $K = N/2 - 1$ for N even or $K = (N - 1)/2$ for N odd, and where the second-order sections $H_k(z)$, for $k = 1, 2, \ldots, K$ are given by

$$H_k(z) = \frac{\cos(\angle H[k]) - z^{-1}\cos(\angle H[k] - \frac{2\pi k}{N})}{1 - 2\cos(\frac{2\pi k}{N})z^{-1} + z^{-2}}. \quad (9.52)$$

Note that the DFT coefficients $H[0]$ and $H[N/2]$ are real-valued for a real system and that if N is an odd integer then the term containing $H[N/2]$ is absent. Figure 9.21 shows a frequency-sampling form structure for $N = 4$ (or a 3rd-order) FIR system based on (9.51) and (9.52).
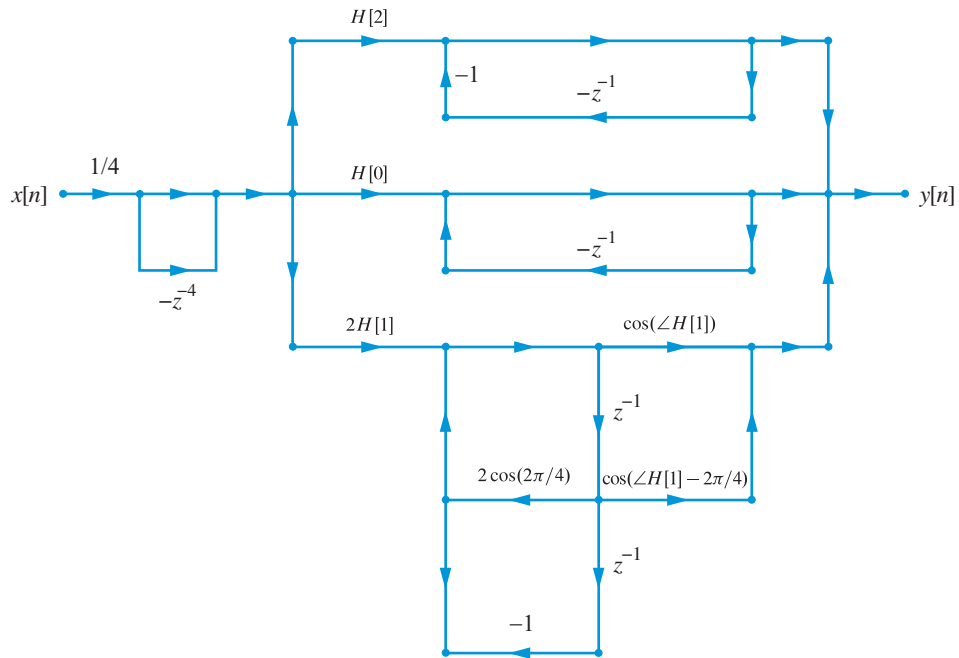
Figure 9.21 Frequency-sampling form structure for realization of a 3rd-order FIR system.

The frequency sampling structure is generally used to implement digital filters designed via frequency-sampling techniques (Section 10.4). Its advantage lies in the implementation of narrowband digital filters. In such filters only a small number of the frequency samples are nonzero, thereby substantially reducing the number of second-order sections and the computational complexity even for a large filter order.

One practical problem with this structure is that the second-order sections are marginally stable since their poles are on the unit circle. Due to finite precision implementation some of the poles can actually move outside the unit circle, which results in an unstable system. Even with an infinite precision the output can become unbounded over a long duration requiring a reset in the operation. One approach to eliminate this problem is an approximate implementation in which the frequency response is sampled on, and the poles are moved to, a circle $|z| = r$ where $r < 1$ but close to 1. The resulting output is not exact but is very close to the true output. This approach and the MATLAB implementation of the frequency-sampling structure are explored in Problem 41.

Example 9.6   FIR frequency-sampling form

The samples of the frequency response of a causal narrowband lowpass FIR filter of length $N = 33$ are given by

$$H[k] = H(e^{j\frac{2\pi}{33}k}) = e^{-j\frac{32\pi}{33}k} \times \begin{cases} 1, & k = 0, 1, 2, 31, 32 \\ 0.5, & k = 3, 30 \\ 0. & \text{otherwise} \end{cases} \tag{9.53}$$
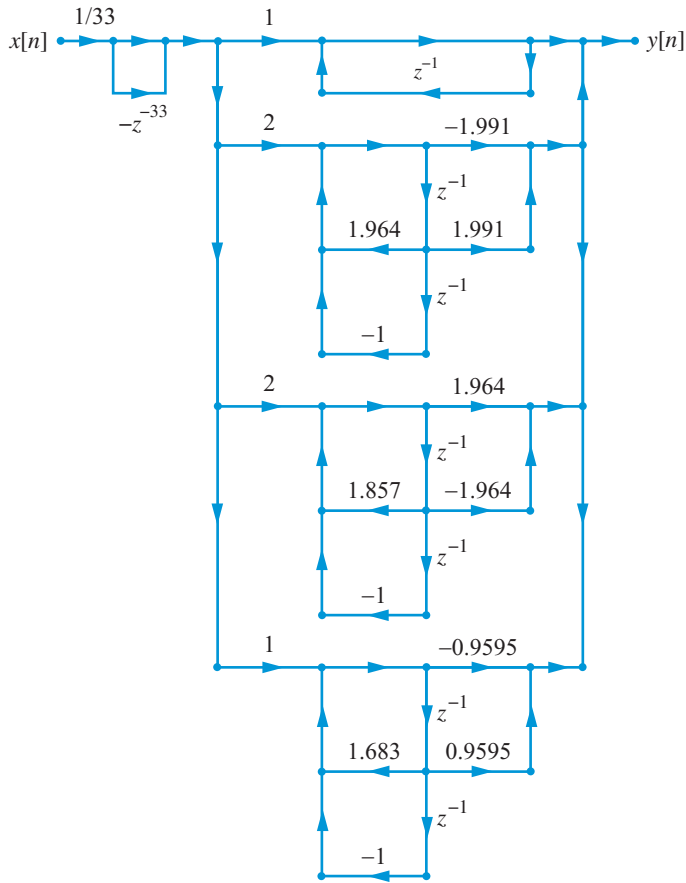
Figure 9.22 Frequency-sampling structure in Example 9.6.

The direct form structure would require 33 multiplications while the linear-phase form would require 17 multiplications. Since there are only seven nonzero frequency samples, the frequency-sampling form would require only three second-order sections resulting in a lower multiplication count. After computing the quantities in (9.51) and (9.52), the system function is given by

$$H(z) = \frac{1 - z^{-33}}{33} \left[\frac{1}{1 - z^{-1}} + \frac{-1.99 + 1.99z^{-1}}{1 - 1.964z^{-1} + z^{-2}} \right.$$

$$\left. + \frac{1.964 - 1.964z^{-1}}{1 - 1.857z^{-1} + z^{-2}} + \frac{-1.96 + 1.96z^{-1}}{1 - 1.683z^{-1} + z^{-2}} \right]. \tag{9.54}$$

The resulting structure is shown in Figure 9.22 which requires only nine multiplications.

9.4 Lattice structures

In this section we provide a brief introduction to lattice structures, which are used in the analysis and synthesis of speech signals (Rabiner and Schafer (2010)) and adaptive filtering (Manolakis *et al.* (2005)). Lattice structures are preferred in these applications over the structures discussed so far because of their modularity, their capability to represent speech signals using a small number of bits, and their ease of real-time coefficient adjustments in adaptive filtering. An *all-zero* lattice models an FIR system while an *all-pole* lattice models an IIR system.

9.4.1 All-zero lattice structure

The typical all-zero lattice structure, shown in Figure 9.23, consists of M sections. The mth section has two outputs and two inputs related by the equations

$$f_m[n] = f_{m-1}[n] + k_m g_{m-1}[n-1], \quad m = 1, 2, \ldots, M \tag{9.55a}$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]. \quad m = 1, 2, \ldots, M \tag{9.55b}$$

The overall-structure input and output are given by

$$x[n] = f_0[n] = g_0[n], \tag{9.56a}$$

$$y[n] = f_M[n]. \tag{9.56b}$$

To understand the basic idea we note that the equations for the first section are

$$f_1[n] = x[n] + k_1 x[n-1], \tag{9.57a}$$

$$g_1[n] = k_1 x[n] + x[n-1]. \tag{9.57b}$$

If we use only the first section and define $y[n] = f_1[n]$, then we have an FIR filter

$$y[n] = a_0^{(1)} x[n] + a_1^{(1)} x[n-1], \tag{9.58}$$
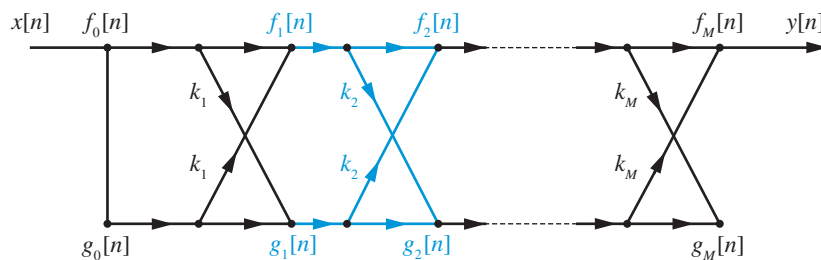
Figure 9.23 An Mth-order all-zero lattice structure.

with coefficients

$$a_0^{(1)} \triangleq 1, \quad a_1^{(1)} \triangleq k_1. \tag{9.59}$$

If we add another section, we have

$$
\begin{aligned}
f_2[n] &= f_1[n] + k_2 g_1[n-1] \\
&= (x[n] + k_1 x[n-1]) + k_2(k_1 x[n-1] + x[n-2]) \\
&= x[n] + (k_1 + k_1 k_2)x[n-1] + k_2 x[n-2],
\end{aligned} \tag{9.60}
$$

which is equivalent to an FIR filter

$$y[n] \triangleq f_2[n] = a_0^{(2)} x[n] + a_1^{(2)} x[n-1] + a_2^{(2)} x[n-2], \tag{9.61}$$

with coefficients

$$a_0^{(2)} \triangleq 1, \quad a_1^{(2)} \triangleq k_1(1+k_2), \quad a_2^{(2)} \triangleq k_2. \tag{9.62}$$

Similarly, we can show that the lower output of the second section is given by

$$g_2[n] = a_2^{(2)} x[n] + a_1^{(2)} x[n-1] + a_0^{(2)} x[n-2]. \tag{9.63}$$

In general, the outputs of the mth section correspond to two FIR filters with the same coefficients but in reverse order:

$$f_m[n] = \sum_{i=0}^{m} a_i^{(m)} x[n-i], \quad m = 1, 2, \dots, M \tag{9.64a}$$

$$g_m[n] = \sum_{i=0}^{m} a_{m-i}^{(m)} x[n-i]. \quad m = 1, 2, \dots, M \tag{9.64b}$$

The system functions of these all-zero FIR filters are given by

$$A_m(z) \triangleq \frac{F_m(z)}{F_0(z)} = \sum_{i=0}^{m} a_i^{(m)} z^{-i}, \quad a_0^{(0)} = 1 \tag{9.65a}$$

$$B_m(z) \triangleq \frac{G_m(z)}{G_0(z)} = \sum_{i=0}^{m} a_{m-i}^{(m)} z^{-i} \triangleq \sum_{i=0}^{m} b_i^{(m)} z^{-i}. \tag{9.65b}$$

We can easily show that, since $b_i^{(m)} = a_{m-i}^{(m)}$, we have

$$B_m(z) = z^{-m} A_m(1/z), \tag{9.66}$$

which is called the *image* polynomial of $A_m(z)$. The fundamental question is how to deter-
mine the lattice structure coefficients k_m, $m = 1, 2, \ldots, M$ from the impulse response of
the FIR system

$$H(z) = \sum_{k=0}^{M} h[k] z^{-k}. \tag{9.67}$$

To comply with (9.65a), we first define the coefficients

$$a_k^{(M)} = h[k]/h[0]. \quad k = 0, 1, \ldots, M \tag{9.68}$$

Careful inspection of (9.59) and (9.62) suggests that the last lattice coefficient is given by

$$k_M = a_M^{(M)}. \tag{9.69}$$

This point provides the basis for the recursive calculation of the remaining coefficients
in decreasing order of the index. Taking the z-transform of (9.55) and using (9.65), we
obtain

$$A_m(z) = A_{m-1}(z) + k_m z^{-1} B_{m-1}(z), \tag{9.70a}$$

$$B_m(z) = k_m A_{m-1}(z) + z^{-1} B_{m-1}(z), \tag{9.70b}$$

with $A_0(z) = B_0(z) = 1$ and $H(z) = h[0]A_M^{(M)}(z)$. Solving the first equation for $B_{m-1}(z)$,
substituting into the second equation, and solving for $A_{m-1}(z)$ yields

$$A_{m-1}(z) = \frac{1}{1 - k_m^2} [A_m(z) - k_m B_m(z)], \tag{9.71}$$

assuming that $k_m^2 \neq 1$. Thus, given $A_m(z)$, we first obtain $k_m = a_m^{(m)}$ and $B_m(z) = z^{-m} A_m(1/z)$; then, we determine $A_{m-1}(z)$ from (9.71) and $k_{m-1} = a_{m-1}^{(m-1)}$. We illustrate
this process with a simple example.

Example 9.7 Lattice parameters for a third-order FIR system
For simplicity we consider a *normalized* FIR system, that is, with $h[0] = 1$. Hence, let the
system function $H(z)$ be

$$H(z) = A_3(z) = 1 + 0.06z^{-1} - 0.42z^{-2} + 0.50z^{-3}. \tag{9.72}$$

From (9.68) and (9.69) we conclude that $k_3 = a_3 = 0.50$. From (9.65b), the image
polynomial of $A_3(z)$ is

$$B_3(z) = 0.50 - 0.42z^{-1} + 0.06z^{-2} + z^{-3}. \tag{9.73}$$

Using (9.72), (9.73) and (9.71) we obtain

$$A_2(z) = \frac{1}{1-k_3^2}\left[A_3(z) - k_3 B_3(z)\right] = 1 + 0.36z^{-1} - 0.6z^{-2}. \tag{9.74}$$

Repeating the above steps for $m = 2$, we have

$$k_2 = a_2^{(2)} = -0.6, \tag{9.75a}$$

$$B_2(z) = -0.6 + 0.36z^{-1} + z^{-2}, \tag{9.75b}$$

$$A_1(z) = \frac{1}{1-k_2^2}\left[A_2(z) - k_2 B_2(z)\right] = 1 + 0.9z^{-1}, \tag{9.75c}$$

which implies that $k_1 = 0.9$. Therefore, the coefficients of the lattice structure are

$$k_1 = 0.90, \quad k_2 = -0.60, \quad k_3 = 0.50. \tag{9.76}$$

■

The steps used in the above example can be incorporated in a simple MATLAB function to determine lattice coefficients $\{k_i\}$ and the gain $G \triangleq h[0]$ from the FIR impulse response $h[n]$. These steps are given by (9.68), (9.69), (9.65b), and (9.71). If we collect all coefficients of $A_m(z)$ and $B_m(z)$ into two row vectors, $\mathbf{a}_m$ and $\mathbf{b}_m$, we can easily implement the algorithm specified by (9.71) using the MATLAB function `k=fir2lat(h)` shown in Figure 9.24. The algorithm fails if $k_m = \pm 1$.

The conversion from lattice coefficients to FIR filter coefficients can be done using (9.70a), starting with $A_0(z) = 1$ and $B_0(z) = 1$ and terminating with $A_M(z)$. Then we set $H(z) = GA_M(z)$ by choosing G to control the gain of the filter at a desired level. This procedure is implemented by the MATLAB function `lat2fir` in Figure 9.25. As an example, the lattice coefficients obtained in Example 9.7 can be converted back to the $A(z)$ polynomial using

```
function [k,G] = fir2lat(h)
% FIR to Lattice Conversion

G = h(1); a = h/G;
M = length(h)-1; k(M) = a(M+1);
for m = M:-1:2
    b = fliplr(a);
    a = (a-k(m)*b)/(1-k(m)^2); a = a(1:m);
    k(m-1) = a(m);
end
```

Figure 9.24 Function for converting FIR filter coefficients to lattice coefficients.

```
function h = lat2fir(k,G)
% Lattice to FIR Conversion
a = 1; b = 1; M = length(k);
for m = 1:1:M
    a = [a,0]+k(m)*[0,b];
    b = fliplr(a);
end
h = G*a;
```

Figure 9.25 Function for converting lattice coefficients to FIR filter coefficients.

```
function y = azlatfilt(k,G,x)
% All-zero Lattice Filter Implementation
M = length(k); f = zeros(1,M); g = f;
x = G*x; y = zeros(size(x));
oldg = zeros(1,M); oldx = 0;
for n=1:length(x)
    f(1) = x(n)+k(1)*oldx;
    g(1) = k(1)*x(n)+oldx;
    oldx = x(n);
    for m = 2:M
        f(m) = f(m-1)+k(m)*oldg(m-1);
        g(m) = k(m)*f(m-1)+oldg(m-1);
        oldg(m-1) = g(m-1);
    end
    y(n) = f(M);
end
```

Figure 9.26 Function for implementation of an all-zero FIR lattice filter.

```
>> k = [0.9,-0.6,0.5]; G = 1;
>> h = lat2fir(k,G)
h =
    1.0000    0.0600    -0.4200    0.5000
```

Hence $A(z) = 1 + 0.06z^{-1} - 0.42z^{-2} + 0.5z^{-3}$ as expected.

Figure 9.26 shows the MATLAB function `azlatfilt` for implementation of the all-zero lattice filter structure shown in Figure 9.23. This structure, which is described by (9.55), requires only M memory locations; however, to simplify programming, we have used two separate arrays `g` and `oldg` to store both $g_m[n]$ and $g_m[n-1]$. Problem 40 discusses a function that uses only one array.
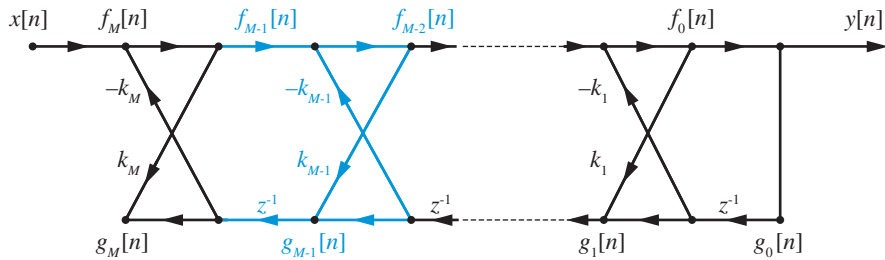
Figure 9.27　An all-pole lattice structure.

9.4.2　All-pole lattice structure

To develop the all-pole lattice structure we note that the FIR system

$$y[n] = x[n] + k_1 x[n-1] \tag{9.77}$$

has the all-zero system function

$$A_1(z) = \frac{Y(z)}{X(z)} = 1 + k_1 z^{-1}. \tag{9.78}$$

If we interchange the roles of input and output in (9.77) we obtain a recursive system

$$y[n] = x[n] - k_1 y[n-1], \tag{9.79}$$

with an all-pole system function given by

$$H_1(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + k_1 z^{-1}} = \frac{1}{A_1(z)}. \tag{9.80}$$

In the all-zero lattice section (9.55) the input is $f_{m-1}[n]$ and the output is $f_m[n]$. Interchanging the role of input and output yields the all-pole lattice section

$$f_{m-1}[n] = f_m[n] - k_m g_{m-1}[n-1], \tag{9.81a}$$

$$g_m[n] = k_m f_{m-1}[n] + g_{m-1}[n-1]. \tag{9.81b}$$

Since the output $f_{m-1}[n]$ is obtained from the input $f_m[n]$, the sections of the Mth order all-pole lattice structure are arranged in decreasing index order from $m = M$ to $m = 1$. The resulting all-pole lattice structure is shown in Figure 9.27.

The all-pole lattice filter for $M = 2$ is described by the following equations

$$f_2[n] = x[n],$$
$$f_1[n] = f_2[n] - k_2 g_1[n-1],$$
$$g_2[n] = k_1 f_1[n] + g_1[n-1],$$
$$f_0[n] = f_1[n] - k_1 g_0[n-1],$$
$$g_1[n] = k_1 f_0[n] + g_0[n-1],$$
$$y[n] = f_0[n] = g_0[n]. \tag{9.82}$$

```
function y = aplatfilt(k,G,x)
% All-pole Lattice Filter Implementation
M = length(k); g = zeros(1,M); f = g;
oldy = 0; x = G*x; y = zeros(size(x));
for n = 1:length(x)
    f(M) = x(n);
    for i = 1:M-1
        m = M+1-i;
        f(m-1) = f(m)-k(m)*g(m-1);
        g(m) = k(m)*f(m-1)+g(m-1);
    end
    y(n) = f(1)-k(1)*oldy;
    g(1) = k(1)*y(n)+oldy;
    oldy = y(n);
end
```

Figure 9.28 Function for implementation of an all-pole IIR lattice filter.

After some simple substitutions and algebraic manipulations we obtain

$$y[n] = -k_1(1 + k_2)y[n-1] - k_2y[n-2] + x[n], \tag{9.83a}$$

$$g_2[n] = k_2y[n] + k_1(1 + k_2)y[n-1] + y[n-2], \tag{9.83b}$$

which are the difference equations for an all-pole system and an all-zero system with

$$H_{ap}(z) = \frac{Y(z)}{X(z)} = \frac{1}{A_2(z)}, \quad H_{az}(z) = \frac{G_2(z)}{Y(z)} = z^{-2}A_2(1/z), \tag{9.84}$$

where

$$A_2(z) \triangleq 1 + a_1^{(2)}z^{-1} + a_2^{(2)}z^{-2}. \tag{9.85}$$

We note that, in general, the coefficients of the all-zero system are identical to the coefficients of the all-pole system, except that they occur in reverse order. Furthermore, the two lattice structures are characterized by the same set $k_1, k_2, \ldots, k_M$ of lattice parameters. As a result, the conversion between direct form and lattice parameters can be done using the algorithms described in Figures 9.24 and 9.25.

Figure 9.28 shows the MATLAB function `aplatfilt` for implementation of the all-pole lattice filter structure shown in Figure 9.27. Note that in this case, we do not need the array `oldg` because we first use the sample $g_m[n-1]$ and then we compute the new value $g_m[n]$. In contrast, in the all-zero lattice, we first compute $g_m[n]$ and then we use $g_m[n-1]$.
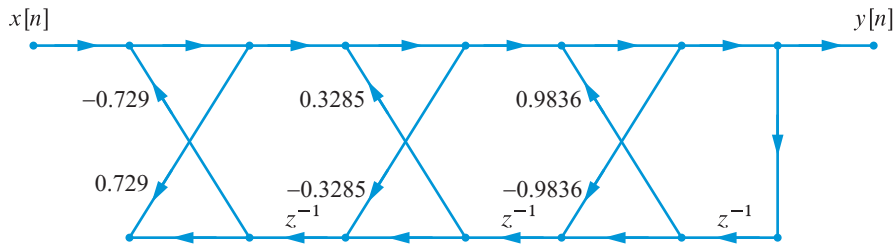
Figure 9.29 An all-pole lattice structure in Example 9.8.

Example 9.8   **Lattice parameters for a third-order all-pole IIR system**

Consider the recursive all-pole system given by

$$y[n] = x[n] + 0.9y[n-1] + 0.81y[n-2] - 0.729y[n-3]. \tag{9.86}$$

Using the `fir2lat` function we obtain the all-pole lattice parameters.

```
>> a = [1,-0.9,-0.81,0.729]
a =
    1.0000    -0.9000    -0.8100    0.7290
>> k = fir2lat(a)
k =
   -0.9836    -0.3285    0.7290
```

The resulting all-pole lattice structure is shown in Figure 9.29. ∎

9.4.3   Further discussion

The all-zero and all-pole lattice structures require twice the number of multiplications per output sample as the direct forms. However, they have two unique advantages compared to direct form structures. The first advantage follows from the following theorem:

Theorem 9.4.1 The roots of the polynomial $A_M(z)$ are inside the unit circle if and only if

$$|k_m| < 1. \quad m = 1, 2, \ldots, M \tag{9.87}$$

A proof of this theorem is given in Gray, Jr. and Markel (1973) and Manolakis *et al.* (2005). If the lattice parameters satisfy (9.87), the FIR system (9.55) is minimum-phase and the all-pole system (9.81) is stable. This property is useful for checking the stability of all-pole lattice filters used to model the human vocal tract for speech analysis and compression applications (see Rabiner and Schafer (2010)). The second advantage is that lattice structures are insensitive to quantization of the *k*-parameters.

The transfer function from $f_M[n]$ to $g_M[n]$ in the all-pole lattice system of Figure 9.27 can be written as

$$\frac{G_M(z)}{F_M(z)} = \frac{G_M(z)}{G_0(z)} \frac{F_0(z)}{F_M(z)} = \frac{B_M(z)}{A_M(z)}, \tag{9.88}$$

because $F_0(z) = G_0(z)$. Therefore, from (9.66) and (9.88), we conclude that

$$\frac{G_M(z)}{F_M(z)} = \frac{z^{-M}A_M(1/z)}{A_M(z)} = \frac{a_M + a_{M-1}z^{-1} + \cdots + z^{-M}}{1 + a_1 z^{-1} + \cdots + a_M z^{-M}}, \tag{9.89}$$

which is the system function of an allpass filter (see Section 5.9).

When the system function has both poles and zeros, we can extend the all-pole lattice structure in Figure 9.27 with a "ladder" section to obtain a *lattice-ladder* pole-zero structure; see Proakis and Manolakis (2007) and Tutorial Problem 14.

9.5 Structure conversion, simulation, and verification

Thus far, we have presented different structures for implementation of an FIR or IIR filter with the same system function or impulse response. In this section, we briefly discuss how to convert between different structures and how to verify that a given structure has been implemented correctly.

Conversion Usually, filter design packages provide the impulse response of an FIR filter or the coefficients of second-order sections for IIR filters. However, on several occasions, the structure specified by the design software is different from the structure chosen for implementation of the filter. Hence, it is important to be able to convert between different structures.

Structures that are generally preferred in practice and are available for both FIR and IIR systems are the direct and cascade form structures. The zero-pole representation is also very desirable. Figure 9.30 concisely describes conversion paths available between these representations using built-in MATLAB functions. The `tf2sos` function converts the direct form structure in the form of a system function (which MATLAB calls a transfer function) to the cascade form using second-order sections while the function `sos2tf` converts the cascade form to the direct form. Similarly, the `tf2zp` function converts the direct form to the zero-pole form while the `zp2tf` function does the opposite conversion. Finally, the two functions `sos2zp` and `zp2sos` convert cascade form to zero-pole form and vice versa. Conversion from direct form to parallel form and vice versa for an IIR system is obtained using the `residuez` function and is explored in Problem 21. A similar conversion for frequency-sampling form for an FIR system is explored in Tutorial Problem 13.

Simulation and verification After obtaining the coefficients of a given structure, the next task is to simulate and verify that a system with these coefficients works correctly. We have discussed many MATLAB functions for simulation of these structures. The `filter` function implements the direct form structure for the IIR and FIR systems. The `sosfilt`

Transfer (System) Function (TF)

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=0}^{N} a_k z^{-k}}$$

sos2tf

tf2sos

tf2zp

zp2tf

$$H(z) = \prod_{k=1}^{K} \frac{b_{k0} + b_{k1} z^{-1} + b_{k2} z^{-2}}{1 + a_{k2} z^{-1} + a_{k2} z^{-2}}$$

Second-Order Sections (SOS)

sos2zp

zp2sos

$$H(z) = b_0 \frac{\prod_{k=1}^{M} (1 - z_k z^{-1})}{\prod_{k=1}^{N} (1 - p_k z^{-1})}$$

Zero-Pole (ZP) Pattern

$$SOS = \begin{bmatrix} b_{10} & b_{11} & b_{21} & 1 & a_{11} & a_{21} \\ b_{20} & b_{21} & b_{22} & 1 & a_{21} & a_{22} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{K0} & b_{K1} & b_{K2} & 1 & a_{K1} & a_{K2} \end{bmatrix}$$

$$z = \begin{bmatrix} z_1 & z_2 & \cdots & z_M \end{bmatrix}$$
$$p = \begin{bmatrix} p_1 & p_2 & \cdots & p_N \end{bmatrix}$$
$$b = \begin{bmatrix} b_0 & b_1 & \cdots & b_M \end{bmatrix}$$
$$a = \begin{bmatrix} 1 & a_1 & \cdots & a_N \end{bmatrix}$$

Figure 9.30 Summary of MATLAB functions for converting between different structures for implementation of LTI systems. The parallel structure is obtained from the system function using the `residuez` function.

function implements the IIR cascade form while the `azlatfilt` and `aplatfilt` functions implement FIR and IIR lattice filters, respectively. Other simulations are explored in Problems 7, 41, and 25.

Verification of the converted structure can be obtained in many different ways. The simplest is to (a) convert the given structure to direct form I, (b) use the `filter` function to implement this structure, (c) excite the structure in (b) and the structure to be tested with the same input and compare the corresponding outputs. Typical test inputs include the unit step, sinusoidal sequences, or random sequences. Another way is to obtain a well known response like an impulse or a step response from one structure and compare it with that from from another structure. The following example demonstrates how we can use the impulse response to verify a direct form to cascade structure conversion.

Example 9.9

An IIR system is described by the system function

$$H(z) = \frac{1 - 3z^{-1} + 11z^{-2} - 27z^{-3} + 18z^{-4}}{16 + 12z^{-1} + 2z^{-2} - 4z^{-3} - z^{-4}}. \tag{9.90}$$

The cascade structure is obtained using the following script:

```
>> b = [1,-3,11,-27,18]; a = [16,12,2,-4,-1];
>> [sos,G] = tf2sos(b,a)
G =
    0.0625
sos =
    1.0000   -3.0000    2.0000    1.0000   -0.2500   -0.1250
    1.0000    0.0000    9.0000    1.0000    1.0000    0.5000
```

Hence the cascade structure is

$$H(z) = 0.0625 \left(\frac{1 - 3z^{-1}2z^{-2}}{1 - 0.25z^{-1} - 0.125z^{-2}} \right) \left(\frac{1 - 9z^{-2}}{1 + z^{-1} + 0.5z^{-2}} \right). \tag{9.91}$$

To verify that the above structure is the correct one we will compute impulse response from two structures and compare them using the following script:

```
>> n = 0:1000; delta = double(n==0);
>> h_df = filter(b,a,delta);
>> h_cf = G*sosfilt(sos,delta);
>> difference = max(abs(h_df-h_cf))
difference =
    2.9976e-15
```

Since the difference between two impulse responses over 100 samples is within the numerical accuracy of MATLAB, the two structures are equivalent.   ■

Similar verifications can be carried out for any structure conversions and some of these are explored in Problems.

Learning summary

- Any system with a rational system function can be implemented using a finite number of adders, multipliers, and unit delays. A specific interconnection of these basic elements, represented by a block diagram, a flow graph, or a set of equations, provides a realization structure for the system.

- The most widely used structures include the direct form I, direct form II, cascade form, parallel form, lattice form, and the transposed version of all these structures.

- Transposed structures are obtained using the transposition theorem, which states that reversing the branch directions, replacing branch nodes by adders and vice versa, and interchanging the input and output nodes yields a system with the same system function.

- The best structure for implementation of FIR systems is a transposed direct form. The best structure for floating-point implementation of IIR systems is a transposed direct form II, whereas the best structure for fixed-point implementation is a cascade interconnection of transposed direct form II sections.

- The lattice structure for FIR (all-zero) filters and IIR (all-pole) filters is primarily used in speech modeling and adaptive filtering applications.

- MATLAB provides a large number of functions to convert between different structures, to implement a given structure, and to verify that specific implementations works properly.

TERMS AND CONCEPTS

Adder A basic computational unit that sums two or more sequences.

All-pole system An IIR system that has only non trivial poles or a system function with the zeroth-order numerator polynomial and a higher-order denominator polynomial in z^{-1}.

All-pole lattice structure A lattice structure containing feedforward and feedback paths that represents the all-pole IIR system.

All-zero system An FIR system that has only non trivial zeros or a system function that is a higher-order polynomial in z^{-1}.

All-zero lattice structure A lattice structure containing only feedforward paths that represents an all-zero FIR system.

Basic elements Computational units like adders, multipliers, and delays that are used in a structure.

Block diagram A pictorial illustration that shows operations described by the difference equation using interconnections between adder, multipliers, and delay elements.

Canonical structure A structure that is implemented using the minimum possible number of delay elements. Direct form II is a canonical structure.

Cascade form A structure that is obtained by expressing the system function as a product of second-order sections. Both FIR and IIR systems have cascade form structures.

Direct form structures A structure that is obtained directly from the system function or the difference equation. Both FIR and IIR systems have direct form structures.

Direct form I The basic version of the direct form that can be obtained directly from the system function or the difference equation.

Direct form II A canonical version of the direct form that is obtained from the direct form I by properly arranging the numerator and denominator parts of the system function. Possible only for the IIR systems.

Frequency-sampling form A parallel structure for the FIR system that uses equispaced samples of the frequency response. Most

useful when the FIR system is a narrowband filter.

Lattice structure An arrangement of basic computational elements in the form of a trellis. Most useful in analysis and synthesis of speech signals and adaptive filtering.

Lattice-ladder structure A lattice-like structure for an IIR system with a rational system function, in which the all-pole lattice is extended with a ladder part to account for the numerator polynomial.

Linear-phase form A variation in the direct form for the linear-phase FIR system that takes advantage of symmetries in the impulse response to reduce multiplication by about 50%.

Linear-phase system A discrete-time system whose phase response is a linear function of the frequency ω which results in an even or odd symmetry in its impulse response. Only an FIR system can have a causal linear-phase impulse response.

Multiplier A basic computational unit that scales a sequence by a constant value.

Normal form A straightforward implementation of the difference equation of a discrete-time system.

Parallel form A structure that is obtained by expressing the system function as a sum of second-order sections. Only IIR systems have parallel form structures.

Second-order sections A rational function whose numerator and or denominator order is at most two. For real systems, these are

formed by combining complex-conjugate numerator and denominator factors (cascade form) or by combining complex-conjugate pole/residue pairs (parallel form).

Signal flow graph A graphical description of the flow of signals and of their operations in the form of signal nodes, directed branches with gain or delays, summing nodes, and branch nodes.

Structures Interconnections of basic computational elements that implement a difference equation of a discrete-time system.

Tapped delay-line Another name for the direct form FIR structure because it gives an appearance of a long delay line that is tapped and the resulting values are linearly combined to obtain the output. Also known as a transversal line.

Transposed form An alternative structure obtained from the given structure using the transposition operation.

Transposition procedure An operation that derives an equivalent structure from the normal one by reversing its branch directions, replacing its branch nodes by summing nodes and vice versa, and interchanging its input and output nodes.

Transversal line Another name for the direct form FIR structure which gives an appearance of a line lying across. Also known as a tapped-delay line.

Unit delay element A basic computational unit that delays (shifts to the right) a sequence by one sample.

MATLAB functions and scripts

Name	Description	Page
filterdf1*	Implementation of direct form I (normal form)	490
filterdf2*	Implementation of direct form II (transposed form)	494
filter	Implementation of direct form II (transposed form)	493
filtic	Computation of initial conditions for filter	493
tf2sos	Direct form to cascade form conversion	496
sos2tf	Cascade form to direct form conversion	496
sosfilt	Implementation of cascade form	497
residuez	Computation of residues needed in parallel form	499
conv	FIR system implementation using convolution	501
filterfirdf*	Direct form implementation of FIR system	502
fir2lat*	FIR direct form to lattice form conversion	514
lat2fir*	Lattice form to FIR direct form conversion	515
azlatfilt*	All-zero lattice form implementation	515
aplatfilt*	All-pole lattice form implementation	517
tf2zp	Direct form to zero-pole form conversion	519
zp2tf	Zero-pole form to direct form conversion	519
sos2zp	Cascade form to zero-pole form conversion	519
zptsos	Zero-pole form to cascade form conversion	519
filtercf*	Implementation of cascade form	497
filterpf*	Implementation of parallel form	500
tf2pf*	Direct form to parallel form conversion	499
pf2tf*	Parallel form to direct form conversion	500

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A detailed treatment of structures for discrete-time systems, at the same level as in this book, is given in Oppenheim and Schafer (2010), Proakis and Manolakis (2007), and Mitra (2006).
2. The practical implementation of digital filters for real-time applications by programming digital signal processors in C, C++ or assembly language is discussed in Kuo and Gan (2005), Kuo *et al.* (2006), Chassaing and Reay (2008), and Welch *et al.* (2006).

Review questions

1. What is a system computational structure and which tasks does it perform?
2. Describe three basic elements and their functions that are used in system structures.
3. Explain the block diagram and signal flow graph representations of a system structure and difference between these two representations.
4. What is a transposition structure and how is it obtained from a normal structure?
5. Any discrete-time system can be realized using many structures. Do all these structures, when implemented using practical devices, have the same computational complexity, memory requirements, or input/output behavior? Explain.
6. What is a direct form structure of an IIR system and how many different forms does it have?
7. How is the direct form II structure obtained from the direct form I structure?
8. The direct form II structure is known as a canonical structure. Explain why?
9. Given a difference equation of a real discrete-time system, explain how you would obtain a cascade form structure containing real-valued second-order sections.
10. It is argued that for a rational system function, its cascade form structure is not unique. Do you agree or disagree? Explain.
11. What is the advantage of the cascade form over the direct form?
12. Given a difference equation of a real discrete-time system, explain how you would obtain its parallel form structure containing real-valued second-order sections.
13. The second-order sections of the cascade and parallel form do not have the same structure. In general, the parallel form second-order section has one branch missing. Explain why?
14. The feedback part of most of the second-order sections in both the cascade and parallel forms is the same. Explain why is this possible.
15. For a rational system function, the parallel form structure is unique. Do you agree or disagree? Explain.
16. What is the advantage of the parallel form over the cascade form?
17. Describe the relative advantages and disadvantages of the direct form, cascade form, and parallel form of an IIR system.
18. Explain why an FIR system does not have a direct form II similar to that for an IIR system.
19. Is there a difference in the second-order sections of an FIR cascade form and IIR cascade form? Explain.
20. An FIR system does not have a parallel form similar to that for an IIR system. Do you agree or disagree? Explain why.
21. What is a linear-phase FIR system and what is its effect on the impulse response?
22. How many different types of linear-phase FIR system are possible and why?
23. What is the advantage of the linear-phase form over the direct form for a linear-phase FIR system?

24. Which FIR system representation is used in the frequency-sampling form and why does it lead to a parallel structure?
25. The parallel structure of the frequency-sampling form contains second-order sections with feedback paths. This implies that the system is a recursive system. Do you agree or disagree? Explain.
26. Does the frequency-sampling form of an FIR system provide a (BIBO) stable structure? If not how can it be stabilized?
27. In which applications are the lattice structures preferred and what are their advantages?
28. Given an FIR system function, how does one obtain all-zero lattice coefficients?
29. What is the condition on the all-zero lattice coefficients for its structure to exist?
30. The all-pole lattice coefficients can be obtained using the procedure that computes all-zero lattice coefficients. Explain why?
31. Which structure is used for an IIR system with poles and zeros?

Problems

Tutorial problems

1. A discrete-time system is described by the following signal flow graph:
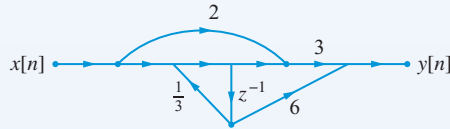
   (a) Determine the difference equation relating output $y[n]$ to the input $x[n]$.
   (b) Determine the impulse response of the system.
2. Consider the following two signal flow graphs:
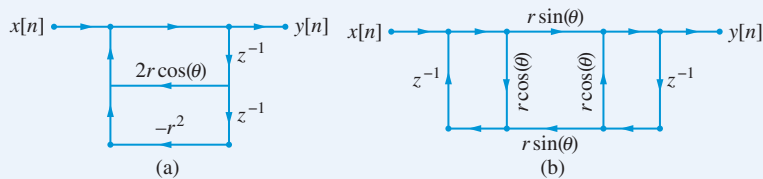
   Determine whether they represent the same discrete-time system.
3. Consider the discrete-time system given by

$$y[n] = 3 \sum_{m=0}^{5} \tfrac{1}{3}^m x[n-m] + \sum_{m=0}^{6} \tfrac{1}{2}^m y[n-m].$$

   Determine and draw the following structures:
   (a) Direct form I (normal),
   (b) Direct form II (normal),

(c) Direct form I (transposed),

(d) Direct form II (transposed).

4. The book toolbox function `filterdf1` implements the IIR direct form I structure with zero initial conditions.

  (a) Modify this function so that it can incorporate initial conditions on the input as well as the output signal. The form of the function should be `y=filterdf1(b,a,x,yi,xi)`. Design the function so that it can use the first three, first four, or all five input arguments. Arguments not provided are assumed to be zero.

  (b) Solve the difference equation

$$y[n] = \tfrac{1}{4}^n u[n] + \tfrac{3}{2}y[n-1] - \tfrac{1}{2}y[n-2], \ n \geq 0 \qquad (9.92)$$

  with initial conditions $y[-2] = 10$ and $y[-1] = 4$.

  (c) Using your MATLAB function in part (a) on the difference equation quantities in part (b) compute $y[n]$ for $0 \leq n \leq 500$ and compare it with your solution in part (b). This will verify that your new `filterdf1` function is correctly implemented.

5. Consider the IIR transposed direct form I structure given in Figure 9.5 and implemented by (9.13).

  (a) Using the `filterdf1` on page 490 as a guide, develop a MATLAB function `y=filterdf1t(b,a,x)` that implements (9.13). Assume zero initial conditions.

  (b) Consider the difference equation (9.92) with zero initial conditions. Determine $y[n]$, $0 \leq n \leq 500$ using your function in part (a) and also using the `filterdf1` function. Compare your results to verify that your `filterdf1t` function is correctly implemented.

6. This problem explores a MATLAB program that converts initial conditions $x[-1], \ldots,$ $x[-M]$ and $y[-1], \ldots, y[-N]$ (direct form I) to $v_1[-1], \ldots, v_{\max\{M,N\}}$ for transpose direct form II structure. It has the same functionality as the built-in MATLAB function `filtic`.

  (a) Show that (9.23b) and (9.23c) can be put in the matrix equation form

$$v[n] = Av[n-1] + bx[n] + ay[n],$$

  where $-N \leq n \leq -1$, $N = \max(M, N)$, and

$$v[n] = \begin{pmatrix} v_1[n] \\ \vdots \\ v_N[n] \end{pmatrix}, \ b = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix}, \ a = \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix}.$$

  (b) Determine matrix A.

  (c) Using the matrix equation in (a) write a MATLAB function `[v]=filteric(b,a,yic,xic)` that computes the direct form II initial conditions using the direct form I initial conditions in the order as given in the beginning of this problem.

  (d) For the difference equation and the initial conditions given in Problem 4, determine the direct form II initial condition vector v and verify it using the built-in MATLAB function `filtic`.

7. The `filterfirdf` implements the FIR direct form structure.
   (a) Develop a new MATLAB function `y=filterfirlp(h,x)` that implements the FIR linear-phase form given its impulse response in `h`. This function should first check if `h` is one of type-I through type-IV and then simulate the corresponding (9.41) through (9.44) equations. If `h` does not correspond to one of the four types then the function should display an appropriate error message.
   (b) Verify your function on each of the following FIR systems:

$$h_1[n] = \{\underset{\uparrow}{1}, 2, 3, 2, 1\},$$

$$h_2[n] = \{\underset{\uparrow}{1}, -2, 3, 3, -2, 1\},$$

$$h_3[n] = \{\underset{\uparrow}{1}, -2, 0, 2, -1\},$$

$$h_4[n] = \{\underset{\uparrow}{1}, 2, 3, 2, 1\},$$

$$h_5[n] = \{\underset{\uparrow}{1}, 2, 3, -2, -1\}.$$

   For verification determine the first ten samples of the step responses using your function and compare them with those from the `filter` function.

8. A discrete-time system is given by

$$H(z) = \frac{1 - 2.55z^{-1} + 4.4z^{-2} - 5.09z^{-3} + 2.41z^{-4}}{1 + 0.26z^{-1} - 0.38z^{-2} - 0.45z^{-3} + 0.23z^{-4}}.$$

   Determine and draw each of the following structures:
   (a) Cascade form with second-order sections in normal direct form I,
   (b) Cascade form with second-order sections in transposed direct form I,
   (c) Cascade form with second-order sections in normal direct form II,
   (d) Cascade form with second-order sections in transposed direct form II.

9. The following numerator and denominator arrays in MATLAB represent the system function of a discrete-time system in direct form:

```
b = [1,-2.61,2.75,-1.36,0.27],
a = [1,-1.05,0.91,-0.8,0.38].
```

   Determine and draw the parallel form structure with second-order section in direct form II.

10. An IIR system is given by

$$H(z) = 4.32 \frac{1 + 2.39z^{-1} + 2.17z^{-2}}{1 - 0.91z^{-1} + 0.28z^{-2}} \frac{1 - 0.33z^{-1} + 1.32z^{-2}}{1 - 1.52z^{-1} + 0.69z^{-2}} \frac{1}{1 + 0.2z^{-1}}.$$

   Determine and draw the following structures:
   (a) Direct form II (normal),
   (b) Direct form I (normal),
   (c) Parallel form with transposed second-order sections.

11. A discrete-time system is described by the difference equation

$$y[n] = 1.9x[n] + 3.94x[n-1] + 4.72x[n-2] + 2.71x[n-3] + 0.61x[n-4].$$

Determine and draw the following structures:
(a) Direct form,
(b) Cascade form,
(c) Frequency-sampling form, and
(d) Lattice form.

12. The FIR system is given by

$$H(z) = 1 + 1.61z^{-1} + 1.74z^{-2} + 1.61z^{-3} + z^{-4}.$$

Determine and draw the following structures:
(a) Direct form II (transposed),
(b) Cascade form,
(c) Linear-phase form,
(d) Frequency-sampling form, and
(e) Lattice form.

13. The frequency-sampling form is developed using (9.50) which uses complex arithmetic.
(a) Using the symmetry conditions of the DFT and root locations, show that (9.50) can be expressed by (9.51) and (9.52) which use real arithmetic.
(b) Develop a MATLAB function `[G,sos]=firdf2fs(h)` that determines frequency-sampling form parameters given in (9.51) and (9.52) given the impulse response in `h`. The matrix `sos` should contain second-order section coefficients in the form similar to the `tf2sos` function while `G` array should contain the respective gains of second-order sections. Incorporate the coefficients for the $H[0]$ and $H[N/2]$ terms in `sos` and `G` arrays.
(c) Verify your function using the frequency-sampling form developed in Example 9.6.

14. Consider a general IIR system with zeros and poles given by

$$H(z) = \frac{C(z)}{A(z)} = \frac{\displaystyle\sum_{m=0}^{M} c_m z^{-m}}{1 + \displaystyle\sum_{n=0}^{N} a_n z^{-n}} \triangleq \frac{1}{A_N(z)} \times C_M(z),$$

where we assume that $M \leq N$. For the denominator part $1/A_N(z)$, an all-pole lattice with coefficients $\{k_n\}$, $1 \leq n \leq N$ can be constructed. The numerator part $C_M(z)$ is incorporated by adding a *ladder* section after tapping the $\{g_m[n]\}$ nodes to form the output

$$y[n] = \sum_{m=0}^{M} d_m g_m[n].$$

(a) Draw a lattice-ladder structure for $M = N = 3$ using the all-pole lattice part and the ladder part expressed in $y[n]$ above.

(b) Show that the Ladder coefficients $\{d_m\}$ are given by

$$C_M(z) = \sum_{m=0}^{M} d_m B_M(z),$$

where $B_M(z)$ is given by (9.66).

(c) Show that the ladder coefficients $\{d_m\}$ can be computed recursively by

$$d_m = c_m + \sum_{k=m}^{M} d_k a_{k-m}^{(k)}, \quad m = M, M-1, \ldots, 0$$

for computation of the ladder coefficients.

(d) Determine and draw the lattice-ladder structure for the following system:

$$H(z) = \frac{10 - 2z^{-1} - 4z^{-2} + 6z^{-3}}{1 + 0.9z^{-1} + 0.81z^{-2} + 0.729z^{-3}}.$$

Basic problems

15. A discrete-time system is described by the following signal flow graph:
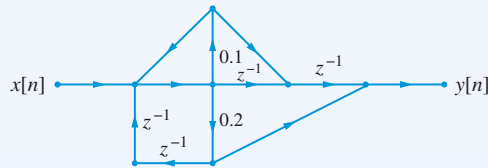
(a) Determine the system function of the system.

(b) Determine the difference equation representation.

16. Two signal flow graphs are shown below.

(a)

(b)
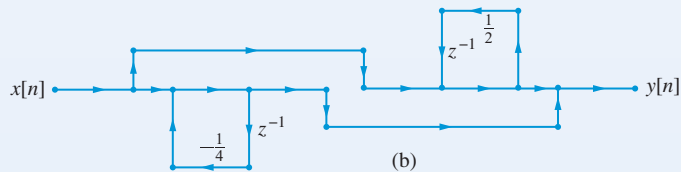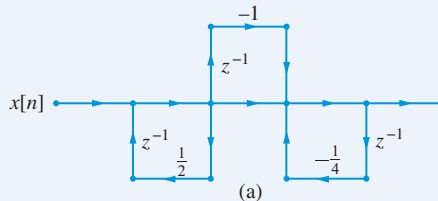
Determine the system function corresponding to each signal flow graph and verify that they represent the same discrete-time system.

17. Consider the discrete-time system given by

$$H(z) = \frac{5 + 10z^{-1} + 15z^{-2} + 10z^{-3} + 5z^{-4}}{1 + 1.35z^{-1} + 1.05z^{-2} + 0.6z^{-3} - 0.12z^{-5}}.$$

Determine and draw the following structures:

(a) Direct form I (normal),
(b) Direct form II (normal),
(c) Direct form I (transposed),
(d) Direct form II (transposed).

18. Consider the IIR normal direct form II structure given in Figure 9.6 and implemented by (9.18) and (9.20).

(a) Using the `filterdf1` on page 490 as a guide, develop a MATLAB function `y=filterdf2(b,a,x)` that implements the normal direct form II structure. Assume zero initial conditions.

(b) Consider the difference equation (9.92) with zero initial conditions. Determine $y[n]$, $0 \leq n \leq 500$ using your function in part (a) and also using the `filterdf1` function. Compare your results to verify that your `filterdf2` function is correctly implemented.

19. A discrete-time system is given by

$$H(z) = \frac{1 - 3.39z^{-1} + 5.76z^{-2} - 6.23z^{-3} + 3.25z^{-4}}{1 + 1.32z^{-1} + 0.63z^{-2} + 0.4z^{-3} + 0.25z^{-4}}.$$

Determine and draw each of the following structures:

(a) Cascade form with second-order sections in normal direct form I,
(b) Cascade form with second-order sections in transposed direct form I,
(c) Cascade form with second-order sections in normal direct form II,
(d) Cascade form with second-order sections in transposed direct form II.

20. Consider the IIR cascade form structure given in Figure 9.11 and implemented by (9.28).

(a) Develop a MATLAB function `y=filtercf(sos,G,x)` that implements (9.28). Assume zero initial conditions.

(b) Consider the cascade form structure given in (9.30). Using your function in part (a) compute the the impulse response $h[n]$, $0 \leq n \leq 100$ of the system. Also compute the impulse response using the `filter` function and the direct form coefficients in (9.29). Compare your two impulse response sequences to verify that your `filtercf` function is correctly implemented.

21. The IIR parallel form is given by (9.31) which is obtained by performing partial fraction expansion of the rational function $H(z)$ and then combining complex-conjugate or two real partial factors. Example 9.4 provides the necessary detail.

(a) Using Example 9.4 as a guide develop a MATLAB function `[sos,C]=tf2pf(b,a)`. The array `C` contains the coefficients $\{C_k\}$ while the $K \times 5$ matrix `sos` contains the second-order section coefficients. In particular you will have to use the `residuez` function twice and the `cplxpair` function to organize complex-conjugate pairs.

(b) Write a MATLAB function `[b,a]=pf2tf(sos,C)` that converts the parallel form coefficients into the direct form coefficients. You may again have to use the `residuez` multiple times.

(c) Verify your functions in (a) and (b) above using the IIR system given in Example 9.4.

22. The following numerator and denominator arrays in MATLAB represent the system function of a discrete-time system in direct form:

```
b = [3.96,6.37,8.3,4.38,2.07],
a = [1,0.39,-0.93,-0.33,0.34].
```

Determine and draw the parallel form structure with second-order sections in direct form II.

23. An IIR system is given by

$$H(z) = \frac{376.63 - 89.05z^{-1}}{1 - 0.91z^{-1} + 0.28z^{-2}} + \frac{-393.11 + 364.4z^{-1}}{1 - 1.52z^{-1} + 0.69z^{-2}} + \frac{20.8}{1 + 0.2z^{-1}}.$$

Determine and draw the following structures:
(a) Direct form II (normal),
(b) Direct form I (normal),
(c) Cascade form with transposed second-order sections.

24. An IIR system is given by

$$H(z) = 5(1 - 0.05z^{-1})\frac{1 - 2.61z^{-1} + 1.77z^{-2}}{1 - 0.32z^{-1} + 0.56z^{-2}}\frac{1 - 0.81z^{-1} + 1.7z^{-2}}{1 + 0.93z^{-1} + 0.58z^{-2}}.$$

Determine and draw the following structures:
(a) Direct form II (normal),
(b) Direct form I (normal),
(c) Parallel form with transposed second-order sections.

25. Consider the IIR parallel form structure given in Figure 9.13 and implemented by (9.31).
(a) Develop a MATLAB function `y=filterpf(sos,C,x)` that implements (9.31). Assume zero initial conditions. The description of `sos` and `C` is given in Problem 21.
(b) Consider the parallel form structure given in (9.33). Using your function in part (a) compute the the impulse response $h[n]$, $0 \le n \le 100$ of the system. Also compute the impulse response using the `filter` function and the direct form coefficients in (9.32). Compare your two impulse response sequences to verify that your `filterpf` function is correctly implemented.

26. A discrete-time system is described by the difference equation

$$y[n] = 5.9x[n] + 1.74x[n-1] + 5.42x[n-2] + 5.42x[n-3] + 1.74x[n-4]$$
$$+ 5.9x[n-5].$$

Determine and draw the following structures:
(a) Direct form,
(b) Cascade form,
(c) Linear-phase form,

(d) Frequency-sampling form, and
(e) Lattice form.

27. The FIR system is given by

$$H(z) = 6.45 + -4.32z^{-1} + -8.32z^{-2} + 7.86z^{-3} + 3.02z^{-4} + -3.19z^{-5}.$$

Determine and draw the following structures
(a) Direct form II (transposed),
(b) Cascade form,
(e) Frequency-sampling form, and
(f) Lattice form.

Assessment problems

28. A discrete-time system is described by the following signal flow graph:
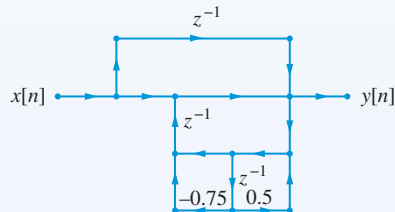
(a) Determine the difference equation representation.
(b) Determine the impulse response of the system.

29. Two signal flow graphs are shown below.

(a)

(b)
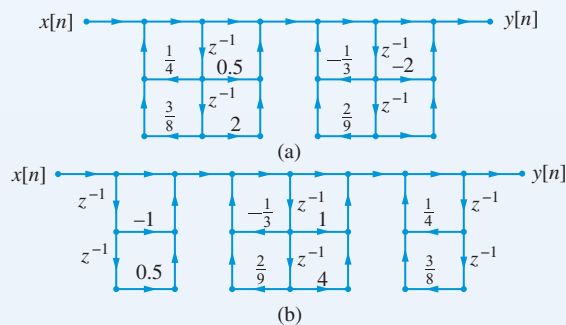
Determine the difference equation relating $y[n]$ to $x[n]$ corresponding to each signal flow graph and determine if they represent the same discrete-time system.

30. Consider the discrete-time system given by

$$y[n] = x[n] + 2.58x[n-1] + 3.55x[n-2] + 2.41x[n-3] + 0.98x[n-4]$$
$$+ 0.08x[n-5] + 0.61y[n-1] + 0.43y[n-2] - 0.32y[n-3]$$
$$- 0.06y[n-4] + 0.06y[n-5].$$

Determine and draw the following structures:
- **(a)** Direct form I (normal),
- **(b)** Direct form II (normal),
- **(c)** Direct form I (transposed),
- **(d)** Direct form II (transposed).

31. The following numerator and denominator arrays in MATLAB represent the system function of a discrete-time system in direct form:

```
b = [1,-2.61,2.75,-1.36,0.27], a = [1,-1.05,0.91,-0.8,0.38].
```

Determine and draw each of the following structures:
- **(a)** Cascade form with second-order sections in normal direct form I,
- **(b)** Cascade form with second-order sections in transposed direct form I,
- **(e)** Cascade form with second-order sections in normal direct form II,
- **(f)** Cascade form with second-order sections in transposed direct form II.

32. The system function of an IIR system is given by

$$H(z) = \frac{0.42 - 0.39z^{-1} - 0.05z^{-2} - 0.34z^{-3} + 0.4z^{-4}}{1 + 0.82z^{-1} + 0.99z^{-2} + 0.28z^{-3} + 0.2z^{-4}}.$$

Determine and draw the parallel form structure with second-order sections in direct form II.

33. Consider the following system function

$$H(z) = 2 \frac{2 + 1.12z^{-1} + 1.08z^{-2}}{1 + 1.06z^{-1} + 0.98z^{-2}} \frac{1 - 1.28z^{-1} + 0.42z^{-2}}{1 + 1.68z^{-1} + 0.8z^{-2}}.$$

Determine and draw the following structures:
- **(a)** Direct form I (transposed),
- **(b)** Direct form II (transposed),
- **(c)** Parallel form (transposed direct form II sections)

34. An IIR system is given by

$$H(z) = \left(37.8 - 2.05z^{-1}\right) + \frac{-28.64 + 18.86z^{-1}}{1 - 0.32z^{-1} + 0.56z^{-2}} + \frac{-5 - 12.31z^{-1}}{1 - 0.93z^{-1} + 0.58z^{-2}}.$$

Determine and draw the following structures:
- **(a)** Direct form II (normal),
- **(b)** Direct form I (normal),
- **(c)** Cascade form with transposed second-order sections.

35. The following structure represents an IIR system:
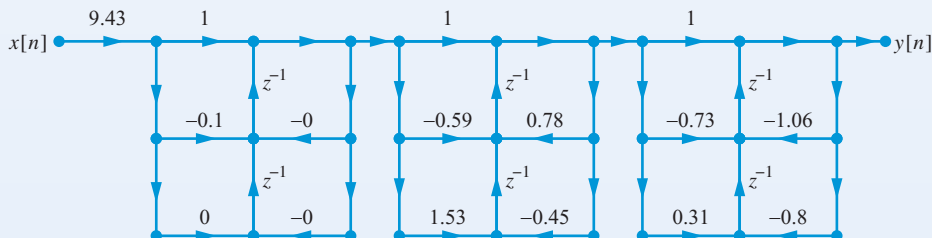
Determine and draw the following structures:

(a) Direct form II (normal),

(b) Direct form I (normal),

(c) Parallel form with transposed second-order sections.

36. Consider the FIR cascade form structure given in Figure 9.17 and implemented by (9.25).

(a) Develop a MATLAB function y=filterfircf(B,G,x) that implements (9.39).

(b) Consider the difference equation

$$y[n] = 5 \sum_{m=0}^{5} (0.9)^m x[n-m], \ n \geq 0 \qquad (9.93)$$

Determine the cascade form coefficients using the tf2sos function and express $H(z)$ in cascade form.

(c) Let $x[n] = (0.5)^n$, $0 \leq n \leq 100$. Determine $y[n]$ using your function in part (a). Also compute $y[n]$ using the sosfilt function. Compare your results to verify that your filterfircf function is correctly implemented.

37. A discrete-time system is described by the difference equation

$$y[n] = 6.17x[n] - 2.78x[n-1] + 2.96x[n-2] - 0.06x[n-3] + 1.61x[n-4]$$
$$+ 1.07x[n-5].$$

Determine and draw the following structures:

(a) Direct form,

(b) Cascade form,

(c) Frequency-sampling form, and

(d) Lattice form.

38. The FIR system is given by

$$H(z) = 5.84 - 20.99z^{-1} + 35.3z^{-2} - 35.3z^{-3} + 20.99z^{-4} + 5.84z^{-5}.$$

Determine and draw the following structures:

(a) Direct form II (transposed),

(b) Cascade form,

(c) Linear-phase form,

(d) Frequency-sampling form, and

(e) Lattice form.

39. Consider the FIR system function $H(z) = \left(1 - 3z^{-1} + z^{-2}\right)^5$.
Determine and draw the following structures:

(a) Direct form structure,

(b) Cascade of first-order sections,

(c) Cascade of second-order sections,

(d) Cascade of fifth-order sections, each with different coefficients,

(e) Linear-phase form,

(f) Cascade of five linear-phase forms.

40. Modify the fir2latc function to use only one g array.

Review problems

41. The frequency-sampling form is given by (9.51) and (9.52) and a MATLAB function to determine its coefficients is explored in Problem 13.

    (a) Develop a MATLAB function `y=filterfirfs(G,sos,x)` that implements the frequency-sampling structure with coefficients in arrays `G` and `sos`.

    (b) The above function may not provide a bounded response for some sequences. By moving pole locations in (9.50) away from the unit circle to a circle of radius $r < 1$ we can obtain a stable but approximate implementation. Develop a set of equations similar to (9.51) and (9.52) that contains the radius parameter r.

    (c) Modify the `firdf2fs` function in Tutorial Problem 13 so that it incorporates the approximate approach developed in part (b) above.

    (d) Modify the `filterfirfs` function so that it implements the approximate frequency-sampling structure obtained in part (c) above.

10 Design of FIR filters

The term "filter" is used for LTI systems that alter their input signals in a prescribed way. Frequency-selective filters, the subject of this chapter, are designed to pass a set of desired frequency components from a mixture of desired and undesired components or to shape the spectrum of the input signal in a desired way. In this case, the filter design specifications are given in the frequency domain by a desired frequency response. The filter design problem consists of finding a practically realizable filter whose frequency response best approximates the desired ideal magnitude and phase responses within specified tolerances.

The design of FIR filters requires finding a polynomial frequency response function that best approximates the design specifications; in contrast, the design of IIR filters requires a rational approximating function. Thus, the algorithms used to design FIR filters are different from those used to design IIR filters. In this chapter we concentrate on FIR filter design techniques while in Chapter 11 we discuss IIR filter design techniques. The design of FIR filters is typically performed either directly in the discrete-time domain using the windowing method or in the frequency domain using the frequency sampling method and the optimum Chebyshev approximation method via the Parks–McClellan algorithm.

Study objectives

After studying this chapter you should be able to:

- Understand how to set up specifications for design of discrete-time filters.
- Understand the conditions required to ensure linear phase in FIR filters and how to use them to design FIR filters by specifying their magnitude response.
- Design FIR filters with linear phase using the windowing method, the frequency sampling method, and the Parks–McClellan algorithm.
- Understand operation and use of the MATLAB filter design and analysis tool.

10.1 The filter design problem

Design of frequency-selective discrete-time filters for practical signal processing applications involves, in general, the following five stages:

1. **Specification:** Specify the desired frequency response function characteristics to address the needs of a specific application.
2. **Approximation:** Approximate the desired frequency response function by the frequency response of a filter with a polynomial or a rational system function. The goal is to meet the specifications with minimum complexity, that is, by using the filter with the lowest number of coefficients.
3. **Quantization:** Quantize the filter coefficients at the required fixed-point arithmetic representation. Quantization of filter coefficients and its implications for performance are discussed in Chapter 15.
4. **Verification:** Check whether the filter satisfies the performance requirements by simulation or testing with real data. If the filter does not satisfy the requirements, return to Stage 2, or reduce the performance requirements and repeat Stage 4.
5. **Implementation:** Implement the system obtained in hardware, software, or both. Structures for the realization of discrete-time systems have been discussed in Chapter 9.

The design procedure outlined by the above stages can be used to design a practical filter that meets the prescribed signal-processing requirements. In this chapter we consider primarily the tasks involved in the first two stages.

10.1.1 Filter specifications

Design of frequency-selective filters usually starts with a specification of their frequency response function. The standard types of ideal frequency-selective filter, shown in Figure 5.9, either pass or eliminate a region of the input spectrum perfectly and they have abrupt ("instantaneous" or "brick wall") transitions between passbands and stopbands. However, as we have discussed in Section 5.4, ideal filters cannot be implemented in practice; therefore, they have to be approximated by practically realizable filters. Practical filters differ from ideal filters in several respects. More specifically, in practical filters (a) the passband responses are not perfectly flat, (b) the stopband responses cannot completely reject (eliminate) bands of frequencies, and (c) the transition between passband and stopband regions takes place over a finite transition band.

The specifications of practical filters usually take the form of a tolerance diagram, as shown in Figure 10.1 for a lowpass filter; similar diagrams exist for the other filter types. The passband edge is denoted by ω_p and the stopband edge is denoted by ω_s. The phase response is either left completely unspecified or may be required to be linear (see Section 10.2). Since we focus on filters with real impulse responses, we need to prescribe the specifications only in the interval $0 \leq \omega \leq \pi$. We note that practical filters may have passband and stopband ripples and they exhibit a gradual (smooth) "roll-off" in the transition band.
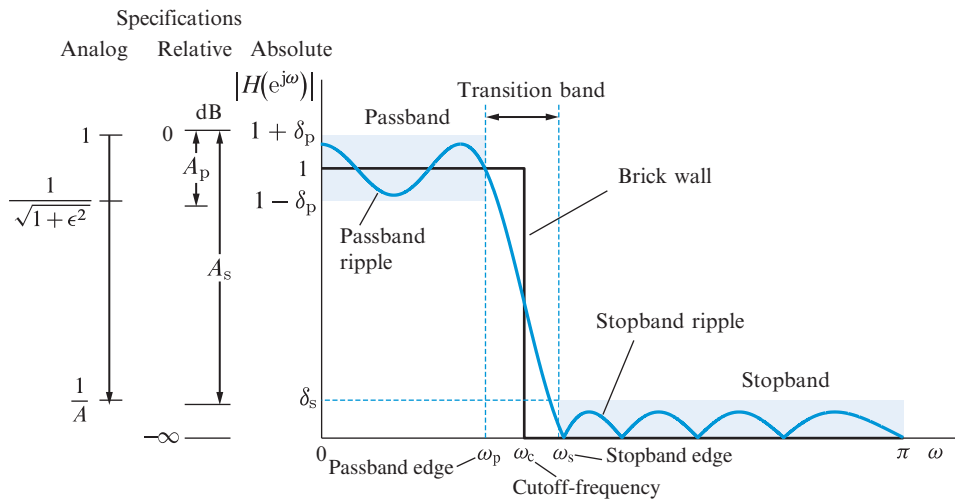
Figure 10.1 Example of tolerance diagram for a lowpass filter.

Absolute specifications In the passband, the magnitude response is required to approximate unity with an error of $\pm\delta_p$, that is, it is specified by

$$1 - \delta_p \leq \left|H(e^{j\omega})\right| \leq 1 + \delta_p, \quad 0 \leq \omega \leq \omega_p \tag{10.1}$$

where $\delta_p \ll 1$ for a well designed filter. In the stopband, we require that the magnitude response approximates zero with an error of $\pm\delta_s$, $\delta_s \ll 1$; thus, we have

$$\left|H(e^{j\omega})\right| \leq \delta_s. \quad \omega_s \leq \omega \leq \pi \tag{10.2}$$

The peak ripple values δ_p and δ_s specify the acceptable tolerances in terms of absolute values, hence the term absolute specifications.

Relative specifications Frequently we define the magnitude of the allowable ripples using relative specifications. The relative specifications are defined by

$$\frac{1 - \delta_p}{1 + \delta_p} \leq \left|H(e^{j\omega})\right| \leq 1, \quad 0 \leq \omega \leq \omega_p \tag{10.3}$$

and

$$\left|H(e^{j\omega})\right| \leq \frac{\delta_s}{1 + \delta_p}, \quad \omega_s \leq \omega \leq \pi \tag{10.4}$$

respectively. If we define the *passband ripple* A_p and the *stopband attenuation* A_s in logarithmic units (dB) by the formulas

$$A_p \triangleq 20 \log_{10}\left(\frac{1 + \delta_p}{1 - \delta_p}\right), \quad A_s \triangleq 20 \log_{10}\left(\frac{1 + \delta_p}{\delta_s}\right) \approx -20 \log_{10}(\delta_s), \tag{10.5}$$

(since $\delta_p \ll 1$) we obtain the following relative tolerance specifications

$$-A_p \leq \left|H(e^{j\omega})\right|, \text{ (in dB)} \leq 0, \qquad\qquad 0 \leq \omega \leq \omega_p \qquad (10.6a)$$

$$\left|H(e^{j\omega})\right|, \text{ (in dB)} \leq -A_s. \qquad\qquad \omega_s \leq \omega \leq \pi \qquad (10.6b)$$

Note that the quantities A_p and A_s are positive and, for a well designed filter, typically $A_p \simeq 0$ and $A_s \gg 1$. The relationship between absolute and relative specifications is further discussed in Tutorial Problem 1.

Continuous-time (analog) filter specifications In practical applications, the passband and stopband edge frequencies are specified in Hz. The values of the normalized frequencies ω_p and ω_s are calculated from the sampling frequency F_s and the edge frequencies F_{pass} and F_{stop} by

$$\omega_p = 2\pi \frac{F_{pass}}{F_s}, \qquad \omega_s = 2\pi \frac{F_{stop}}{F_s}. \qquad (10.7)$$

Since the design of IIR filters is usually done by converting analog filters into equivalent digital filters, we note that analog filters are traditionally specified using the quantities ϵ and A as shown in Figure 10.1. These quantities are defined by

$$20 \log_{10}\left(\sqrt{1 + \epsilon^2}\right) = A_p \quad \text{and} \quad 20 \log_{10}(A) = A_s, \qquad (10.8)$$

which gives

$$\epsilon = \sqrt{10^{(0.1A_p)} - 1} \quad \text{and} \quad A = 10^{(0.05A_s)}. \qquad (10.9)$$

Example 10.1   Conversion of filter specifications

A lowpass digital filter is specified by the following relative specifications:

$$\omega_p = 0.3\pi, \; A_p = 0.5 \text{ dB}; \quad \omega_s = 0.5\pi, \; A_s = 40 \text{ dB}.$$

Then from (10.5) the absolute specifications for the filter are given by

$$A_p = 0.5 = 20 \log_{10}\left(\frac{1 + \delta_p}{1 - \delta_p}\right) \Rightarrow \delta_p = 0.0288,$$

$$A_s = 40 = 20 \log_{10}\left(\frac{1 + \delta_p}{\delta_s}\right) \Rightarrow \delta_s = 0.0103.$$

Similarly, using (10.9), the analog filter specifications are given by

$$\epsilon = \sqrt{10^{(0.1A_p)} - 1} = 0.3493 \quad \text{and} \quad A = 10^{(0.05A_s)} = 100.$$

Tutorial Problem 1 examines some more specification conversions.   ∎

Finally, we note that there are three additional classes of useful filter that cannot be specified by a tolerance scheme like the one shown in Figure 10.1. These classes are: differentiators, Hilbert transformers, and shaping lowpass filters (see Section 10.7).

10.1.2   Filter approximation

Consider an ideal filter with impulse response sequence $h_d[n]$ and frequency response function $H_d(e^{j\omega})$. We wish to find a *practical* filter $H(e^{j\omega})$ which approximates the *desired* filter $H_d(e^{j\omega})$ according to the design specifications (10.6). Since the practical filter should be causal, stable, and should have a finite-order rational system function

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}, \tag{10.10}$$

we will next discuss the implications of causality, stability, and rational form of system function on the filter approximation problem.

The stability and causality requirements have some crucial implications on the characteristics of $H(e^{j\omega})$, which follow from the following Paley–Wiener theorem:

Theorem 1 (Paley–Wiener): If $h[n]$ has finite energy and $h[n] = 0$ for $n < 0$, then

$$\int_{-\pi}^{\pi} \left| \ln \left| H(e^{j\omega}) \right| \right| d\omega < \infty. \tag{10.11}$$

Conversely, if $\left| H(e^{j\omega}) \right|$ is square integrable and the integral (10.11) is finite, then we can obtain a phase response $\angle H(e^{j\omega})$ so that the filter $H(e^{j\omega}) = \left| H(e^{j\omega}) \right| \times e^{j\angle H(e^{j\omega})}$ is causal; the solution $\angle H(e^{j\omega})$ is unique if $H(z)$ is minimum phase. A proof of this theorem and its implications are discussed in Papoulis (1977).

An important consequence of this theorem is that the frequency response of a stable and causal system cannot be zero over any finite band of frequencies because, in this case, the integral becomes infinite. Hence, *any stable ideal frequency-selective filter must be noncausal*.

As we discussed in Section 5.8, there are 2^{M+N} systems with a rational system function (10.10) which have the same magnitude response but different phase responses; the phase response can be uniquely determined from the magnitude response only if the system is minimum phase. The Paley–Wiener theorem generalizes this result to LTI systems with arbitrary (nonrational) system functions. Therefore, *given the magnitude response* $\left| H(e^{j\omega}) \right|$ *of a causal and stable system, we cannot assign its phase response arbitrarily*. There are two approaches to deal with this problem:

1. Impose constraints on the phase response, for example $\angle H(e^{j\omega}) = -\alpha\omega$, and obtain a filter whose magnitude response satisfies the design specifications.

2. Obtain a filter whose magnitude response satisfies the design specifications irrespective of the resulting phase response.

The interdependence between magnitude and phase response of causal systems should be expected given the relationship between the real and imaginary parts of their frequency response. For example, a real $h[n]$ can be decomposed into its even and odd parts as follows:

$$h[n] = h_{\mathrm{e}}[n] + h_{\mathrm{o}}[n], \tag{10.12}$$

where

$$h_{\mathrm{e}}[n] = \tfrac{1}{2}(h[n] + h[-n]), \tag{10.13a}$$

$$h_{\mathrm{o}}[n] = \tfrac{1}{2}(h[n] - h[-n]). \tag{10.13b}$$

If $h[n]$ is causal, it is uniquely defined by its even part

$$h[n] = 2h_{\mathrm{e}}[n]u[n] - h_{\mathrm{e}}[0]\delta[n]. \tag{10.14}$$

If $h[n]$ is absolutely summable, the DTFT of $h[n]$ exists, and it can be written as

$$H(\mathrm{e}^{\mathrm{j}\omega}) = H_{\mathrm{R}}(\mathrm{e}^{\mathrm{j}\omega}) + \mathrm{j}H_{\mathrm{I}}(\mathrm{e}^{\mathrm{j}\omega}), \tag{10.15}$$

where $H_{\mathrm{R}}(\mathrm{e}^{\mathrm{j}\omega})$ is the DTFT of $h_{\mathrm{e}}[n]$. Thus, if a filter is real, causal, and stable, its frequency response $H(\mathrm{e}^{\mathrm{j}\omega})$ is uniquely defined by its real part $H_{\mathrm{R}}(\mathrm{e}^{\mathrm{j}\omega})$. Indeed, we first obtain $h_{\mathrm{e}}[n]$ by inverting $H_{\mathrm{R}}(\mathrm{e}^{\mathrm{j}\omega})$, then we determine $h[n]$ from (10.14), and finally, we obtain $H(\mathrm{e}^{\mathrm{j}\omega})$ from $h[n]$. This implies a relationship between the real and imaginary parts of $H(\mathrm{e}^{\mathrm{j}\omega})$, which is formally given by the following *discrete Hilbert transform* expression:

$$H_{\mathrm{I}}(\mathrm{e}^{\mathrm{j}\omega}) = -\frac{1}{2\pi} \int_{-\pi}^{\pi} H_{\mathrm{R}}(\mathrm{e}^{\mathrm{j}\omega}) \cot\left(\frac{\omega - \theta}{2}\right) \mathrm{d}\theta. \tag{10.16}$$

A detailed treatment of discrete Hilbert transforms is given by Oppenheim and Schafer (2010) and by Papoulis (1977); see also Tutorial Problem 2.

10.1.3   Optimality criteria for filter design

In any specific application we want to find a rational system function $H(z)$ such that the magnitude response $\left|H(\mathrm{e}^{\mathrm{j}\omega})\right|$ or the phase response $\angle H(\mathrm{e}^{\mathrm{j}\omega})$ approximate the shape of some ideal filter responses. The coefficients of $H(z)$ obtained depend on the criterion used to determine what is considered an optimum approximation.

Mean-squared-error approximation A widely used criterion of the "goodness" of the approximation is the mean-square error over the frequency interval of interest $\mathcal{B}$, that is,

$$E_2 \triangleq \left[\frac{1}{2\pi} \int_{\mathcal{B}} \left|H_{\mathrm{d}}(\mathrm{e}^{\mathrm{j}\omega}) - H(\mathrm{e}^{\mathrm{j}\omega})\right|^2 \mathrm{d}\omega\right]^{1/2}. \tag{10.17}$$

If some frequencies are more important than others, we could use a properly selected *weighting* function $W_f(e^{j\omega}) \geq 0$ to weight the error in (10.17). The interval $\mathcal{B}$ is usually the union of all passbands and stopbands of the filter.

Minimax approximation The mean-square error measures the total or average error in the interval $\mathcal{B}$; however, a small mean-square error does not preclude the possibility of large errors at individual frequencies. Nevertheless, in some applications it is very important that the error be small at all frequencies of the interval $\mathcal{B}$; in such cases, we may want to minimize the maximum absolute deviation, that is, the error

$$E_\infty \triangleq \max_{\omega \in \mathcal{B}} \left| H_d(e^{j\omega}) - H(e^{j\omega}) \right|. \tag{10.18}$$

The maximum norm is the natural criterion to use in designing filters which have an assigned accuracy throughout the interval $\mathcal{B}$. The solution minimizing the error function (10.18) is called a *minimax* or *best uniform* or *Chebyshev approximation*.

Maximally-flat approximation A third approach is based on a truncated Taylor series expansions. For example, suppose that we want the function $A(\omega) \triangleq \left| H(e^{j\omega}) \right|^2$ to be very close to the function $A_d(\omega) \triangleq \left| H_d(e^{j\omega}) \right|^2$ at $\omega = \omega_0$. Expanding both functions in Taylor's series form about ω_0, we obtain

$$A_d(\omega) = A_d(\omega_0) + \frac{A_d^{(1)}(\omega_0)}{1!}(\omega - \omega_0) + \frac{A_d^{(2)}(\omega_0)}{2!}(\omega - \omega_0)^2 + \cdots, \tag{10.19a}$$

$$A(\omega) = A(\omega_0) + \frac{A^{(1)}(\omega_0)}{1!}(\omega - \omega_0) + \frac{A^{(2)}(\omega_0)}{2!}(\omega - \omega_0)^2 + \cdots \tag{10.19b}$$

The approximation will be very good at $\omega = \omega_0$ if $A(\omega_0) = A_d(\omega_0)$ and if as many derivatives as possible are equal. If the first $(m - 1)$ derivatives are equal, the error will start with the mth term, that is,

$$E(\omega) \triangleq A_d(\omega) - A(\omega) = \frac{A_d^{(m)}(\omega_0) - A^{(m)}(\omega_0)}{m!}(\omega - \omega_0)^m + \cdots \tag{10.20}$$

If all possible derivatives are equal, we say that the error function is completely flat at $\omega = \omega_0$ because the function cannot change in value from that at $\omega = \omega_0$. If m is finite, we say that the approximation is optimum according to the *maximally flat* criterion. Taylor approximations are very good about some point ω_0, but they get worse as we move away from that point. The well known *Butterworth approximation* is a special case of a Taylor approximation.

We should mention that there are some filter design techniques that use a combination of these criteria; for example, a Chebyshev approximation in the passband and a maximally flat approximation in the stopband. Also, there are some popular design approaches, for example, the windowing method, that do not directly use any criterion.

The main purpose of any filter design technique is to determine the coefficients of a system function or difference equation that approximates a desired frequency response or impulse response within specified tolerances. In this sense, the design of FIR filters

requires the solution of polynomial approximation problems, whereas the design of IIR filters involves approximation problems with rational functions. Hence, the techniques used for the design of FIR filters are different from the techniques used to design IIR filters. In this chapter, we discuss filter design techniques that are well-established, widely used, and have been implemented in many software packages. Thus, we focus more on fundamental concepts, principles of operation, and limitations of each approach and less on computational aspects.

10.2   FIR filters with linear phase

As we stated in Section 5.3, a filter with constant frequency response magnitude and linear phase over a certain frequency band passes a signal with a spectrum over the same frequency band undistorted. In this section, we investigate the implications of this requirement for the design of practical filters.

Without loss of generality, we focus on the ideal lowpass filter; however, the same results apply to all ideal filters. The frequency response of an ideal lowpass filter with linear phase is

$$H_{\text{lp}}(e^{j\omega}) = \begin{cases} e^{-j\alpha\omega}, & |\omega| < \omega_c \\ 0. & \omega_c < |\omega| \le \pi \end{cases} \tag{10.21}$$

The corresponding impulse response from (4.91) is given by

$$h_{\text{lp}}[n] = \frac{\sin \omega_c (n - \alpha)}{\pi (n - \alpha)}. \tag{10.22}$$

If $\omega_c = \pi$ and $\alpha = n_d$ (integer), we can show that $h_{\text{lp}}[n] = \delta[n - n_d]$; that is, the ideal lowpass filter corresponds to an ideal delay operation (see Problem 23).

The impulse response (10.22) can be obtained by sampling the impulse response $h_{\text{lp}}^c(t)$ of an ideal continuous-time lowpass filter with cutoff frequency $\Omega_c = \omega_c/T$ at $t = nT$. Indeed, we can easily see that

$$h_{\text{lp}}[n] = h_{\text{lp}}^c(t)|_{t=nT} = \left. \frac{\sin \Omega_c(t - \alpha T)}{\pi (t - \alpha T)} \right|_{t=nT}. \tag{10.23}$$

The function $h_{\text{lp}}^c(t)$ is symmetric about $t = \alpha T$ for any value of α. However, the sequence $h_{\text{lp}}[n] = h_{\text{lp}}^c(nT)$ may or may not be symmetric dependent on the value of delay α. There are three cases, which are illustrated in Figure 10.2(a).

1. The delay α is an integer n_d. In this case, the sequence $h_{\text{lp}}[n]$ is symmetric about its sample at $n = n_d$.
2. The quantity 2α is an integer or α is an integer plus one-half. In this case, the sequence is symmetric about the middle between the samples at $\alpha - 1/2$ and $\alpha + 1/2$. Note that the center of symmetry is not a sample of the sequence.
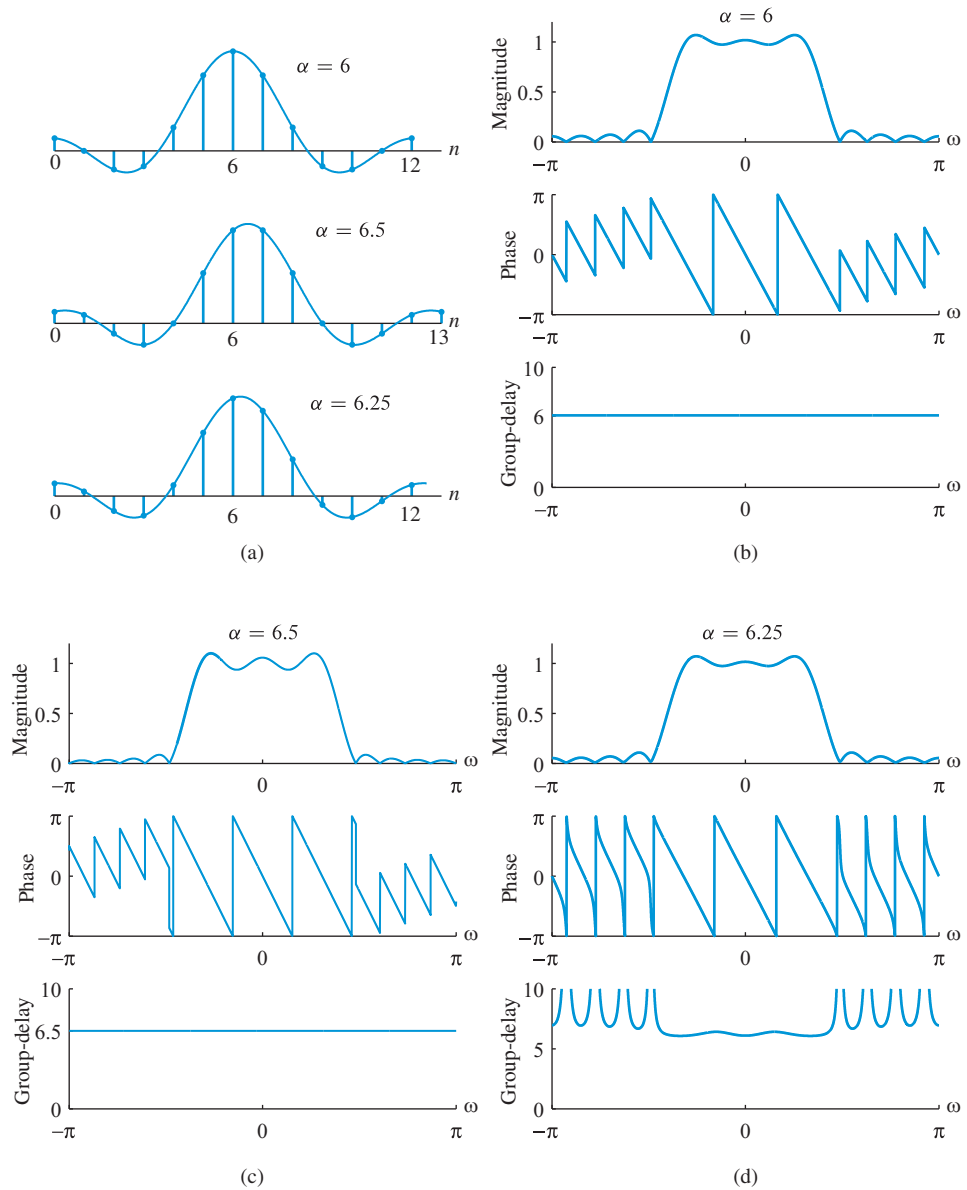
Figure 10.2 (a) Impulse responses of ideal analog lowpass filter with $\omega_c = 0.4\pi$ and delays of $\alpha = 6, 6.5$, and 6.25, respectively, along with samples of the truncated ideal lowpass FIR filter impulse response. Shown are magnitude response, phase response, and group delay of the truncated ideal lowpass FIR filter with (b) $\alpha = 6$ and $M = 12$, (c) $\alpha = 6.5$ and $M = 13$, and (d) $\alpha = 6.25$ and $M = 13$. Observe that if 2α is not an integer, then the phase response is nonlinear and the group delay is not a constant.

3. The quantity 2α is not an integer. In this case, there is *no* symmetry at all because of the misalignment between the continuous-time impulse response and the sampling grid.

We next create a causal FIR filter by setting $h[n] = h_{\text{lp}}[n]$ for $0 \le n \le M$ and zero elsewhere. We note that if $2\alpha = M = $ integer, the impulse $h[n]$ is symmetric about $\alpha = M/2$ and the resulting filters have linear phase. This is illustrated in Figure 10.2(b) for $M = 12$ (even) and Figure 10.2(c) when $M = 13$ (odd). The time delay $\alpha = M/2$ is an integer multiple of the sampling interval only when M is even. However, as is illustrated in Figure 10.2(d), if 2α is *not* an integer, symmetry is lost and the resulting discrete-time filter has a nonlinear phase response and a variable group delay.

  If for a given value of delay α, we choose a value of $M > 2\alpha$ the symmetry of $h[n]$ is lost and the resulting filter has a nonlinear phase response; thus, we cannot have causal IIR filters with linear phase ($M = \infty$). In fact, it has been shown by Clements and Pease (1989) that only FIR filters with linear phase are practically realizable; all causal IIR filters with a rational system function have a nonlinear phase response.

  We show next that depending on the type of symmetry (even or odd) and whether the filter order M is an even or odd integer, there are four types of FIR filter with linear phase. More specifically, if the frequency response is given by

$$H(e^{j\omega}) = \sum_{n=0}^{M} h[n]e^{-j\omega n}, \tag{10.24}$$

the four cases of interest are specified by the following conditions:

$$h[n] = \pm h[M - n]. \quad M = \text{ even or odd integer} \tag{10.25}$$

To avoid confusion, we emphasize that M is the order of the system function polynomial, whereas $L \triangleq M + 1$ is the length or duration of its impulse response.

10.2.1   Type-I FIR linear-phase filters

A type-I FIR system has a symmetric impulse response with even order M, that is,

$$h[n] = h[M - n]. \quad 0 \le n \le M \tag{10.26}$$

To understand the implications of (10.25) we consider the case $M = 4$. From the symmetry condition (10.26) we obtain $h[4] = h[0]$ and $h[3] = h[1]$. Thus, we have

$$
\begin{aligned}
H(e^{j\omega}) &= h[0] + h[1]e^{-j\omega} + h[2]e^{-j2\omega} + h[3]e^{-j3\omega} + h[4]e^{-j4\omega} \\
&= \left(h[0]e^{j2\omega} + h[1]e^{j\omega} + h[2] + h[3]e^{-j\omega} + h[4]e^{-j2\omega} \right) e^{-j2\omega} \\
&= \left(h[0]e^{j2\omega} + h[1]e^{j\omega} + h[2] + h[1]e^{-j\omega} + h[0]e^{-j2\omega} \right) e^{-j2\omega} \\
&= \left(h[2] + 2h[1]\cos\omega + 2h[0]\cos 2\omega \right) e^{-j2\omega} \\
&\triangleq \left(a[0] + a[1]\cos\omega + a[2]\cos 2\omega \right) e^{-j2\omega}.
\end{aligned}
$$

In general, the frequency response of a type-I FIR filter can be expressed as

$$
H(e^{j\omega}) = \left(\sum_{k=0}^{M/2} a[k] \cos \omega k \right) e^{-j\omega M/2} \triangleq A(e^{j\omega}) e^{-j\omega M/2}, \tag{10.27}
$$

where $A(e^{j\omega})$ is a real, even, and periodic function of ω with coefficients given by

$$
a[0] = h[M/2], \quad a[k] = 2h[(M/2) - k]. \quad k = 1, 2, \ldots, M/2 \tag{10.28}
$$

10.2.2   Type-II FIR linear-phase filters

A type-II FIR system has a symmetric impulse response (10.26) with odd order M. For $M = 5$, we can easily see that the frequency response can be expressed as follows

$$
H(e^{j\omega}) = h[0] + h[1]e^{-j\omega} + h[2]e^{-j2\omega} + h[2]e^{-j3\omega} + h[1]e^{-j4\omega} + h[0]e^{-j5\omega}
$$

$$
= \{2h[2] \cos(\omega/2) + 2h[1] \cos(3\omega/2) + 2h[0] \cos(5\omega/2)\} e^{-j(5/2)\omega}
$$

$$
\triangleq \{b[1] \cos(\omega/2) + b[2] \cos(3\omega/2) + b[3] \cos(5\omega/2)\} e^{-j(5/2)\omega}.
$$

The general expression for the frequency response for a type-II FIR filter is

$$
H(e^{j\omega}) = \left(\sum_{k=1}^{(M+1)/2} b[k] \cos\left[\omega \left(k - \frac{1}{2} \right) \right] \right) e^{-j\omega M/2} \triangleq A(e^{j\omega}) e^{-j\omega M/2}, \tag{10.29}
$$

where the delay $M/2$ is an integer plus one-half and the coefficients of $A(e^{j\omega})$ are

$$
b[k] = 2h[(M+1)/2 - k]. \quad k = 1, 2, \ldots, (M+1)/2 \tag{10.30}
$$

Using the identity $2 \cos \alpha \cos \beta = \cos(\alpha + \beta) + \cos(\alpha - \beta)$ we can express $A(e^{j\omega})$ in terms of $\cos \omega k$ as for type-I filters. Indeed, for $M = 5$ we can show that

$$
A(e^{j\omega}) = b[1] \cos(\omega/2) + b[2] \cos(3\omega/2) + b[3] \cos(5\omega/2)
$$

$$
= \cos\left(\frac{\omega}{2} \right) \{(b[1] - b[2] + b[3]) + 2(b[2] - b[3]) \cos \omega + 2b[3] \cos 2\omega\}.
$$

The general expression, which is derived in Tutorial Problem 3, is given by

$$
A(e^{j\omega}) = \cos\left(\frac{\omega}{2} \right) \sum_{k=0}^{(M-1)/2} \tilde{b}[k] \cos \omega k, \tag{10.31}
$$

where

$$
b[k] = \begin{cases} \frac{1}{2}(\tilde{b}[1] + 2\tilde{b}[0]), & k = 1 \\ \frac{1}{2}(\tilde{b}[k] + \tilde{b}[k-1]), & 2 \le k \le (M-1)/2 \\ \frac{1}{2}\tilde{b}[(M-1)/2]. & k = (M+1)/2 \end{cases}
\tag{10.32}
$$

The expression (10.32) is used for the design of type-II FIR filters using the minimax criterion. The design algorithm provides $\tilde{b}[k]$; the impulse response is obtained using (10.32) and (10.30). For this reason we have expressed $b[k]$ in terms of $\tilde{b}[k]$; however, it is straightforward to express $\tilde{b}[k]$ in terms of $b[k]$.

We note from (10.31) that at $\omega = \pi$, $A(\mathrm{e}^{\mathrm{j}\omega}) = 0$, independent of $\tilde{b}[k]$ or equivalently $h[k]$. This implies that filters with a frequency response that is nonzero at $\omega = \pi$, for example, a highpass filter, cannot be satisfactorily approximated with a type-II filter.

10.2.3   Type-III FIR linear-phase filters

A type-III FIR system has an antisymmetric impulse response with even order M

$$
h[n] = -h[M - n]. \quad 0 \le n \le M
\tag{10.33}
$$

In this case the delay $M/2$ is an integer and the frequency response is given by

$$
H(\mathrm{e}^{\mathrm{j}\omega}) = \left(\sum_{k=1}^{M/2} c[k]\sin\omega k\right)\mathrm{j}\mathrm{e}^{-\mathrm{j}\omega M/2} \triangleq \mathrm{j}A(\mathrm{e}^{\mathrm{j}\omega})\mathrm{e}^{-\mathrm{j}\omega M/2},
\tag{10.34}
$$

where

$$
c[k] = 2h[M/2 - k]. \quad k = 1, 2, \ldots, M/2
\tag{10.35}
$$

The antisymmetry condition (10.33) yields $h[M/2] = -h[M/2]$ for $k = M/2$; this requires that $h[M/2] = 0$ for every type-III filter.

Using the trigonometric identity $2\sin\alpha\cos\beta = \sin(\alpha + \beta) + \sin(\alpha - \beta)$ we can show that $A(\mathrm{e}^{\mathrm{j}\omega})$ can be expressed as a linear combination of cosines by

$$
A(\mathrm{e}^{\mathrm{j}\omega}) = \sin\omega \sum_{k=0}^{M/2} \tilde{c}[k]\cos\omega k,
\tag{10.36}
$$

where

$$
c[k] = \begin{cases} \frac{1}{2}(2\tilde{c}[0] - \tilde{c}[1]), & k = 1 \\ \frac{1}{2}(\tilde{c}[k-1] - \tilde{c}[k]), & 2 \le k \le (M/2) - 1 \\ \frac{1}{2}\tilde{c}[(M/2) - 1]. & k = M/2 \end{cases}
\tag{10.37}
$$

We note from (10.36) that at $\omega = 0$ and $\omega = \pi$, $A(e^{j\omega}) = 0$, independent of $\tilde{c}[k]$ or equivalently $h[k]$. In addition, the presence of the factor $j = e^{j\pi/2}$ in (10.34) shows that the frequency response is imaginary. Thus, type-III filters are most suitable for the design of differentiators and Hilbert transformers.

10.2.4   Type-IV FIR linear-phase filters

If the impulse response is antisymmetric (10.33) and M is odd, then we have

$$H(e^{j\omega}) = \left(\sum_{k=1}^{(M+1)/2} d[k] \sin\left[\omega\left(k - \frac{1}{2} \right) \right] \right) j e^{-j\omega M/2} \triangleq j A(e^{j\omega}) e^{-j\omega M/2}, \quad (10.38)$$

where

$$d[k] = 2h[(M+1)/2 - k]. \quad k = 1, 2, \ldots, (M+1)/2 \quad (10.39)$$

Similarly to type-III filters, $A(e^{j\omega})$ can be expressed as a sum of cosines by

$$A(e^{j\omega}) = \sin\left(\frac{\omega}{2} \right) \sum_{k=0}^{(M-1)/2} \tilde{d}[k] \cos \omega k, \quad (10.40)$$

where

$$d[k] = \begin{cases} \frac{1}{2}(2\tilde{d}[0] - \tilde{d}[1]), & k = 1 \\ \frac{1}{2}(\tilde{d}[k-1] - \tilde{d}[k]), & 2 \le k \le (M-1)/2 \\ \frac{1}{2}\tilde{d}[(M-1)/2]. & k = (M+1)/2 \end{cases} \quad (10.41)$$

For type-IV filters we have $A(e^{j\omega}) = 0$ at $\omega = 0$, independent of $\tilde{d}[k]$ or equivalently $h[k]$. As for type-III, this class of filters is most suitable for approximating differentiators and Hilbert transformers.

10.2.5   Amplitude response function of FIR filters with linear phase

Careful inspection of the frequency response functions of type-I–IV FIR filters with linear phase shows that they can all be expressed in the form (see Table 10.1)

$$H(e^{j\omega}) = \sum_{n=0}^{M} h[n] e^{-j\omega n} \triangleq A(e^{j\omega}) e^{j\Psi(e^{j\omega})}. \quad (10.42)$$

The real function $A(e^{j\omega})$, which may take positive or negative values, is called *amplitude response* to distinguish it from the magnitude response $|H(e^{j\omega})|$. The angle $\Psi(e^{j\omega})$ is a linear function of ω defined by

$$\Psi(e^{j\omega}) \triangleq -\alpha\omega + \beta. \quad (10.43)$$

Table 10.1 Properties of impulse response sequence $h[n]$ and frequency response function $H(e^{j\omega}) = A(e^{j\omega})e^{j\Psi(e^{j\omega})}$ of FIR filters with linear phase.

Type	$h[k]$	M	$A(e^{j\omega})$	$A(e^{j\omega})$	$\Psi(e^{j\omega})$
I	even	even	$\displaystyle\sum_{k=0}^{M/2} a[k]\cos\omega k$	even–no restriction	$-\dfrac{\omega M}{2}$
II	even	odd	$\displaystyle\sum_{k=1}^{\frac{M+1}{2}} b[k]\cos\left[\omega\left(k-\dfrac{1}{2}\right)\right]$	even $A(e^{j\pi})=0$	$-\dfrac{\omega M}{2}$
III	odd	even	$\displaystyle\sum_{k=1}^{M/2} c[k]\sin\omega k$	odd $A(e^{j0})=0$ $A(e^{j\pi})=0$	$\dfrac{\pi}{2}-\dfrac{\omega M}{2}$
IV	odd	odd	$\displaystyle\sum_{k=1}^{\frac{M+1}{2}} d[k]\sin\left[\omega\left(k-\dfrac{1}{2}\right)\right]$	odd $A(e^{j0})=0$	$\dfrac{\pi}{2}-\dfrac{\omega M}{2}$

The constant β results from the presence of factor j in the frequency response of type-III and IV systems. Thus $\beta = 0$ for type-I and II filters and $\beta = \pi/2$ for type-III and IV filters. We can restrict β in the range $0 \leq \beta \leq \pi$ because a minus sign can be incorporated into $A(e^{j\omega})$. The constant β does *not* affect the group delay

$$\tau_{\mathrm{gd}}(\omega) = -\frac{\mathrm{d}\Psi(e^{j\omega})}{\mathrm{d}\omega} = \alpha. \tag{10.44}$$

The amplitude function $A(e^{j\omega})$, in contrast to the magnitude $|H(e^{j\omega})|$, is analytic, that is, its derivative exists for all ω. This definition of $A(e^{j\omega})$, makes $\Psi(e^{j\omega})$ a continuous function of ω; in contrast, the phase $\angle H(e^{j\omega})$ is not continuous (see Section 5.3). These important properties are illustrated in Figure 10.3 using a moving-average filter (see Tutorial Problem 4). Consequently, the use of $A(e^{j\omega})$ and $\Psi(e^{j\omega})$ makes possible the design of FIR filters with linear phase using optimization techniques.

A filter with any magnitude response and linear phase can be expressed as

$$H(e^{j\omega}) = |H(e^{j\omega})|e^{-j\alpha\omega} = A(e^{j\omega})e^{-j\alpha\omega+j\beta}. \tag{10.45}$$

The first expression provides a polar representation of the complex function $H(e^{j\omega})$ because $|H(e^{j\omega})| \geq 0$. The magnitude response $|H(e^{j\omega})|$ is associated with the linear phase response $\angle H(e^{j\omega}) = -\alpha\omega$. Since the amplitude function $A(e^{j\omega})$ may be positive or negative, the second expression is not a polar representation of $H(e^{j\omega})$; therefore,
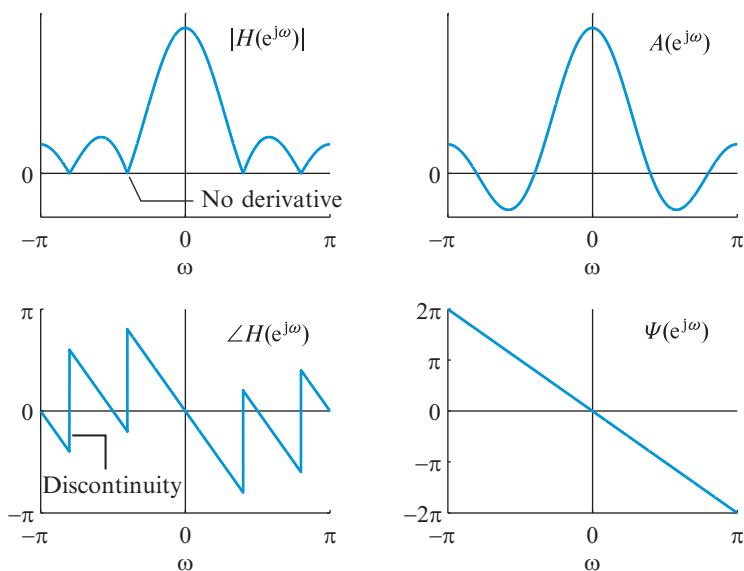
Figure 10.3 The differences between the magnitude and amplitude response representations of the frequency response function.

strictly speaking, the quantity $\Psi(e^{j\omega}) = -\alpha\omega + \beta$ is not a phase response function. Sometimes, the term *generalized linear phase* is used in the literature to emphasize the use of decomposition (10.45); to minimize confusion and ambiguity about the concept of phase response, we avoid this terminology. Since the system $H_{zp}(e^{j\omega}) \triangleq |H(e^{j\omega})|$ has zero-phase response, any linear-phase system can be represented by a zero-phase filter followed by a time-delay system.

To gain some additional insight, we recall the following Fourier transforms

$$h_e[n] \xrightarrow{\text{DTFT}} A_e(e^{j\omega}) \qquad \Rightarrow h_e[n-\alpha] \xrightarrow{\text{DTFT}} A_e(e^{j\omega})e^{-j\omega\alpha}, \qquad (10.46a)$$

$$h_o[n] \xrightarrow{\text{DTFT}} jA_o(e^{j\omega}) \qquad \Rightarrow h_o[n-\alpha] \xrightarrow{\text{DTFT}} A_o(e^{j\omega})e^{-j\omega\alpha + j\pi/2}, \qquad (10.46b)$$

where $h_e[n]$ and $A_e(e^{j\omega})$ are real and even, and $h_o[n]$ and $A_o(e^{j\omega})$ are real and odd. Thus, a system with frequency response given by (10.45), where $A(e^{j\omega})$ has even or odd symmetry, has constant group delay.

Computation of amplitude response $A(e^{j\omega})$ Using (10.27) and (10.28), (10.29) and (10.30), (10.34) and (10.35), and (10.38) and (10.39), it is straightforward to compute $A(e^{j\omega})$ for each of the four types of linear-phase filter given their impulse response $h[n]$. The book toolbox function `A=amplresp(h,w)` computes the amplitude response in array `A` at frequency locations provided in array `w`, given the impulse response values in `h`. It first determines the type of the linear-phase FIR filter and then uses the appropriate equations.

Table 10.2 Unified representation and uses of FIR filters with linear phase.

Type	M	$Q(e^{j\omega})$	$P(e^{j\omega})$	$H(e^{j\omega}) = 0$	Uses
I	even	1	$\displaystyle\sum_{k=0}^{M/2} \tilde{a}[k]\cos\omega k$		LP, HP, BP, BS, multiband filters
II	odd	$\cos(\omega/2)$	$\displaystyle\sum_{k=0}^{\frac{M-1}{2}} \tilde{b}[k]\cos\omega k$	$\omega = \pi$	LP, BP
III	even	$\sin\omega$	$\displaystyle\sum_{k=0}^{M/2} \tilde{c}[k]\cos\omega k$	$\omega = 0, \pi$	differentiators, Hilbert transformers
IV	odd	$\sin(\omega/2)$	$\displaystyle\sum_{k=0}^{\frac{M-1}{2}} \tilde{d}[k]\cos\omega k$	$\omega = 0$	differentiators, Hilbert transformers

Its use is shown in Tutorial Problem 6 and it will also be used extensively for plotting amplitude responses in several linear-phase FIR filter design examples.

Unified representation From (10.27), (10.31), (10.36), and (10.41) we conclude that the function $A(e^{j\omega})$ can be expressed as the product of two terms

$$A(e^{j\omega}) = Q(e^{j\omega})P(e^{j\omega}), \tag{10.47}$$

where $Q(e^{j\omega})$ is a fixed function of ω dependent on the type of the filter and $P(e^{j\omega})$ is a sum of cosines dependent on the filter coefficients. The different forms of this decomposition and their implications in filter design are summarized in Table 10.2. This representation unifies the design of the four types of FIR filter with linear phase using the minimax criterion. The design algorithm provides the coefficients of $P(e^{j\omega})$, which can be used to obtain the samples of the impulse response. The alternative representation, based on the coefficients $\{a[k], b[k], c[k], d[k]\}$, is better suited to the design of filters using least square optimization or linear programming techniques.

10.2.6   Zero locations of FIR filters with linear phase

The symmetry or antisymmetry of the impulse response of FIR systems and its length impose restrictions on the locations of the system function zeros. These restrictions upon the zero pattern provide an alternative explanation for the constraints upon the shape of frequency response summarized in Table 10.2.

To investigate the effects of the symmetry condition (10.26) on the zeros of type-I systems, we note that the system function can be written as

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n} = \sum_{n=0}^{M} h[M-n]z^{-n}$$

$$= \sum_{k=M}^{0} h[k]z^{k}z^{-M} = z^{-M}H(z^{-1}). \tag{10.48}$$

From (10.48) we conclude that if $z_0 = re^{j\theta}$ is a zero of $H(z)$, then $z_0^{-1} = r^{-1}e^{-j\theta}$ is also a zero of $H(z)$. If $h[n]$ is real, then its complex conjugate $z_0^* = re^{-j\theta}$ is also a zero of $H(z)$. Therefore, if $h[n]$ is real, each zero not on the unit circle will be part of a cluster of four conjugate reciprocal zeros of the form

$$(1 - re^{j\theta}z^{-1})(1 - re^{-j\theta}z^{-1})(1 - r^{-1}e^{j\theta}z^{-1})(1 - r^{-1}e^{-j\theta}z^{-1}). \tag{10.49}$$

Zeros on the unit circle ($r = 1$) and real zeros ($\theta = \pm k\pi$) appear in pairs as

$$(1 - e^{j\theta}z^{-1})(1 - e^{-j\theta}z^{-1}) \text{ or } (1 \pm rz^{-1})(1 \pm r^{-1}z^{-1}), \tag{10.50}$$

respectively. A real-coefficient polynomial $H(z)$ satisfying the condition (10.48) is called a *mirror-image polynomial* and its zeros exhibit *mirror-image symmetry* with respect to the unit circle. Finally, we note that if there is a zero at $z = \pm 1$ ($r = 1$ and $\theta = 0$ or $\theta = \pi$), $H(z)$ includes factors of the form $(1 \pm z^{-1})$.

The same considerations apply for type-II systems, with one important exception: type-II systems *always* have a zero at $z = -1$. Indeed, from (10.48) we have

$$H(-1) = (-1)^M H(-1). \tag{10.51}$$

If M is even, (10.51) is a simple identity; if M is odd, we have $H(-1) = -H(-1)$, which implies that $H(-1)$ must be zero.

For systems with an antisymmetric impulse response, see (10.33), we have

$$H(z) = -z^{-M}H(z^{-1}), \tag{10.52}$$

which shows that $H(z)$ is a mirror-image polynomial. For $z = 1$, we have $H(1) = -H(1)$; thus, $H(z)$ *must* have a zero at $z = 1$ for any M (even or odd). For $z = -1$, we obtain $H(-1) = -(-1)^M H(-1)$; if M is even, we have $H(-1) = -H(-1)$, which implies that $H(-1) = 0$. Thus, type-III linear-phase systems *must* have a zero at $z = -1$. The presence of a zeros at $z = -1$ implies that $H(e^{j\omega})$ has a zero at $\omega = \pi$; thus, a type-III filter cannot be used for the design of highpass filters. Similar considerations apply to type-II, III, and IV linear-phase filters.

Figures 10.4 and 10.5 illustrate the properties of type-I–IV FIR filters with linear phase by showing examples of the impulse response, pole-zero pattern, magnitude response, and amplitude response for each filter type. We note that $A(e^{j\omega})$ has period 2π for type-I and type-III filters and period 4π for type-II and type-IV filters. We also note that, since $|H(e^{j\omega})|$ is an even function, the function $A(e^{j\omega})$ can be either even or odd about $\omega = 0$ (see Tutorial Problem 7).

Type I: Symmetric Impulse Response, Even Order $M = 8$

Type II: Symmetric Impulse Response, Odd Order $M = 7$
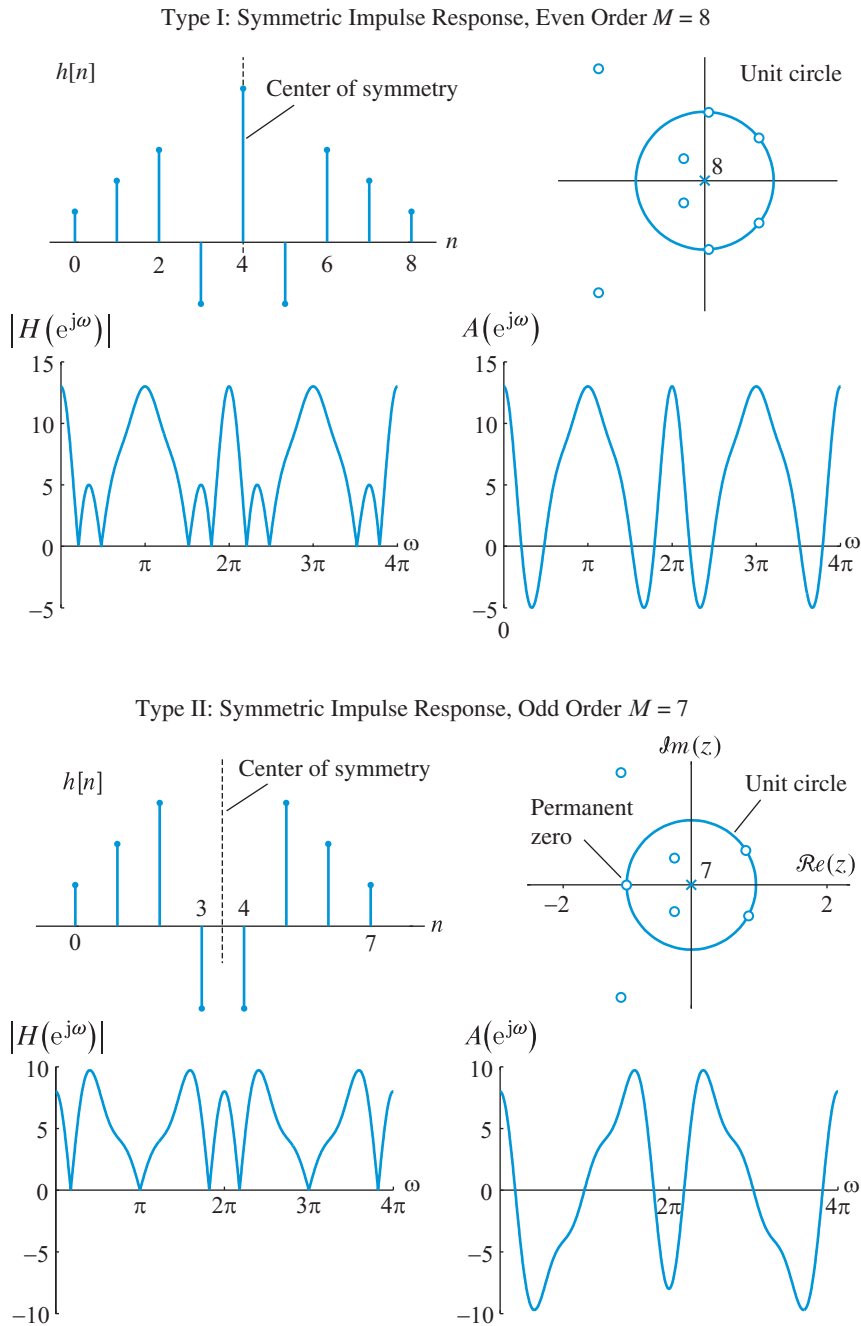
Figure 10.4 Impulse response, pole-zero pattern, magnitude response, and amplitude response for type-I and II FIR filters with linear phase.

Type III: Anti-Symmetric Impulse Response, Even Order $M = 8$

Type IV: Anti-symmetric Impulse Response, Odd Order $M = 7$

Figure 10.5 Impulse response, pole-zero pattern, magnitude response, and amplitude response for type-III and IV FIR filters with linear phase.
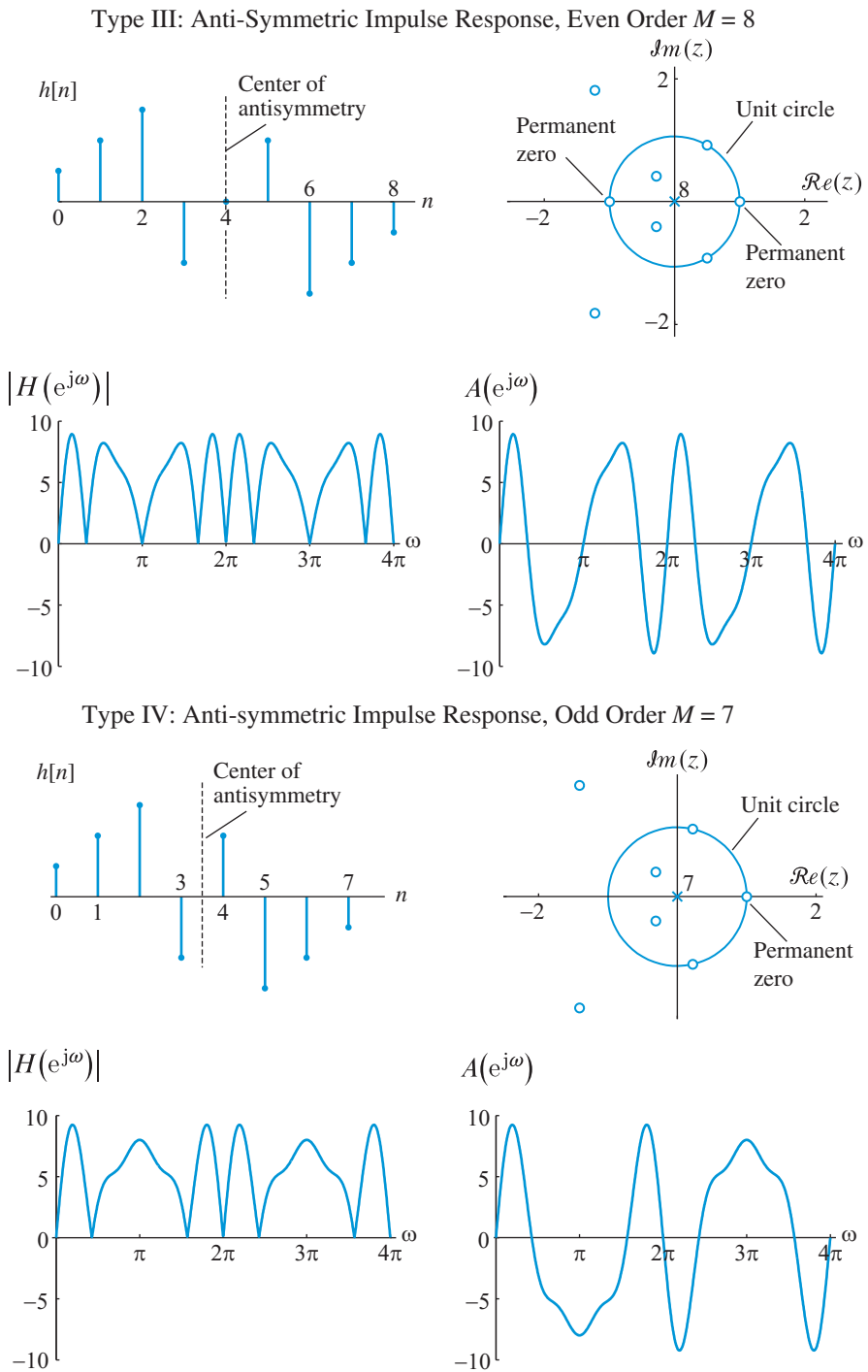
10.3 | Design of FIR filters by windowing

The easiest way to obtain an FIR filter is simply to truncate the impulse response of an IIR filter. Although this truncation minimizes the mean square error between the original and obtained frequency responses, the resulting filter has unacceptable size ripples. The windowing design approach reduces these ripples to a desired level by applying a window to the impulse response; however, the filter obtained is no longer optimum in the mean square error sense.

10.3.1 | Direct truncation of an ideal impulse response

Suppose that we wish to approximate a desired ideal frequency response function

$$H_d(e^{j\omega}) = \sum_{n=-\infty}^{\infty} h_d[n]e^{-j\omega n}, \tag{10.53}$$

with an FIR filter $h[n]$, $0 \le n \le M$, by minimizing the mean-square error

$$\varepsilon^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H_d(e^{j\omega}) - H(e^{j\omega}) \right|^2 d\omega. \tag{10.54}$$

Using Parseval's identity (4.94) we can express ε^2 in the time-domain as

$$\varepsilon^2 = \sum_{n=0}^{M}(h_d[n] - h[n])^2 + \sum_{n=-\infty}^{-1} h_d^2[n] + \sum_{n=M+1}^{\infty} h_d^2[n]. \tag{10.55}$$

The last two terms on the right hand side of (10.55) depend only on $H_d(e^{j\omega})$ and are not affected by the choice of $h[n]$. Since the first term is nonnegative, the mean-square error is minimized if and only if this term is zero. Thus, the optimum solution is

$$h[n] = \begin{cases} h_d[n], & 0 \le n \le M \\ 0. & \text{otherwise} \end{cases} \tag{10.56}$$

We note that the best, in the mean-square error sense, FIR approximation to the ideal IIR impulse response $h_d[n]$ is obtained by truncation. The ideal impulse response $h_d[n]$ is obtained from $H_d(e^{j\omega})$ using the inverse DTFT.

Frequency domain effects of truncation To understand the effects of truncation on the frequency response function, we express the truncated impulse response as the product of the desired IIR impulse response and the finite *rectangular window* sequence:

$$w[n] = \begin{cases} 1, & 0 \le n \le M \\ 0. & \text{otherwise} \end{cases} \tag{10.57}$$

Thus, we can express (10.56) as follows:

$$h[n] = h_d[n]w[n]. \tag{10.58}$$

Applying the windowing theorem (4.156) of the DTFT, we obtain

$$H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta. \tag{10.59}$$

Thus, the approximation frequency response $H(e^{j\omega})$ is the periodic convolution of the desired ideal frequency response with the Fourier transform of the window

$$W(e^{j\omega}) = \sum_{n=0}^{M} e^{-j\omega n} = \frac{1 - e^{-j\omega(M+1)}}{1 - e^{-j\omega}} = \frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)} e^{-j\omega M/2}. \tag{10.60}$$

The effects of this convolution process are illustrated in Figure 10.6 using the amplitude response $A_d(e^{j\omega})$ of the ideal lowpass filter (10.21) for $\alpha = M/2$ and the amplitude function $A_w(e^{j\omega})$ of the rectangular window, given by

$$A_d(e^{j\omega}) \triangleq \begin{cases} 1, & |\omega| \le \omega_c \\ 0, & \omega_c < |\omega| \le \pi \end{cases}, \quad A_w(e^{j\omega}) \triangleq \frac{\sin[\omega(M+1)/2]}{\sin(\omega/2)}. \tag{10.61}$$

The value of $A(e^{j\omega})$ at $\omega = \omega_0$ is equal to the integral of the blue area, which changes as $A_w(e^{j(\omega-\theta)})$ slides along the frequency axis. When the mainlobe is inside the passband or
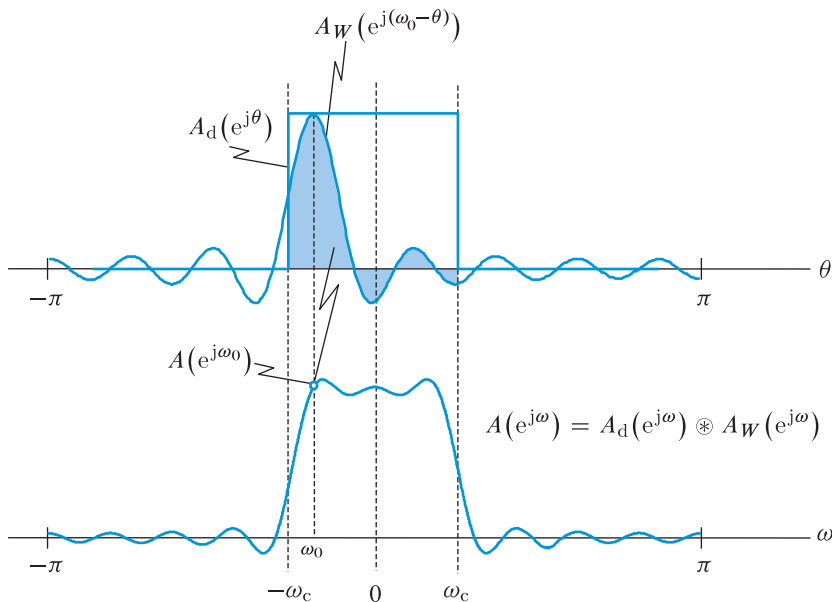
Figure 10.6 The effect of windowing in the frequency domain.

stopband the integral changes slowly because of variations in the side lobes. Essentially the sidelobes insert their ripples into the "flat" passband and stopband regions of the ideal frequency response. However, when the large-size mainlobe slides through the sharp discontinuity ("brick wall") of the ideal response the integral decreases quickly. The result is two gradual transition bands, which resemble the left and right sides of the mainlobe. The transition regions are approximately symmetric about the point of discontinuity. An alternative explanation of this operation is provided in Figure 7.25. This behavior is a result of the Gibbs phenomenon, which was discussed in Section 4.2.1.

Computation of ripples and transition bandwidth To determine how the truncation of the ideal impulse response dictates the passband ripple, stopband attenuation, and width of transition band of the ideal lowpass filter, we use (10.59) to evaluate the amplitude response of the designed filter

$$A\big(e^{j\omega}\big) = \frac{1}{2\pi} \int_{-\pi}^{\pi} A_d\big(e^{j\theta}\big) A_w\big(e^{j(\omega-\theta)}\big) d\theta = \frac{1}{2\pi} \int_{-\omega_c}^{\omega_c} A_w\big(e^{j(\omega-\theta)}\big) d\theta. \tag{10.62}$$

The window amplitude function $A_w\big(e^{j\omega}\big)$ in (10.61) has the first zero crossing at $\omega_1 = \pm 2\pi/L$, where $L = M + 1$ is the length of the window; thus, the width of the mainlobe is about $4\pi/L$. The negative peak of the first sidelobe occurs approximately at $\omega = 3\pi/L$; therefore, we have $|A_w\big(e^{j3\pi/L}\big)/A_w\big(e^{j0}\big)| = |\sin(3\pi/2)/\sin(3\pi/2L)|/L$. For large values of L, where we can use the approximation $\sin\theta \approx \theta$, this ratio is about $2/(3\pi)$; hence, the peak sidelobe magnitude is 13 dB below the peak of the mainlobe (see Figure 10.9).

To investigate the properties of $A\big(e^{j\omega}\big)$ at the vicinity of the cutoff frequency $\omega = \omega_c$, we first consider the case for $\omega < \omega_c$ and break (10.62) into two parts as follows, (see Porat (1997)):

$$A\big(e^{j\omega}\big) = \frac{1}{2\pi} \int_{-\omega_c}^{\omega} A_w\big(e^{j(\omega-\theta)}\big) d\theta + \frac{1}{2\pi} \int_{\omega}^{\omega_c} A_w\big(e^{j(\omega-\theta)}\big) d\theta. \tag{10.63}$$

Using the change of variables $\phi = \omega - \theta$ in the first integral, $\phi = (\omega - \theta)L/2$ in the second integral, and the symmetry of $A_w\big(e^{j\omega}\big)$, we obtain

$$A\big(e^{j\omega}\big) = \frac{1}{2\pi} \int_{0}^{\omega+\omega_c} A_w\big(e^{j\phi}\big) d\phi + \frac{1}{L\pi} \int_{0}^{(\omega_c-\omega)L/2} A_w\big(e^{j(2\phi/L)}\big) d\phi. \tag{10.64}$$

From Figure 10.6 we see that for ω close to ω_c, the first integral can be approximated by the integral of $A_w\big(e^{j\omega}\big)$ from 0 to π, which is equal to π. Also, for large L we have $A_w\big(e^{j(2\phi/L)}\big) = \sin\phi/(\sin\phi/L) \approx L\sin\phi/\phi$. Therefore, we obtain

$$A\big(e^{j\omega}\big) \approx \frac{1}{2} + \frac{1}{\pi} \int_{0}^{(\omega_c-\omega)L/2} \frac{\sin\phi}{\phi} d\phi = \frac{1}{2} + \frac{1}{\pi} \text{Si}\big[(\omega_c - \omega)L/2\big], \tag{10.65}$$

where the *sine integral function* is defined in Papoulis (1977) by

$$\text{Si}(\xi) \triangleq \int_{0}^{\xi} \frac{\sin\phi}{\phi} d\phi. \tag{10.66}$$
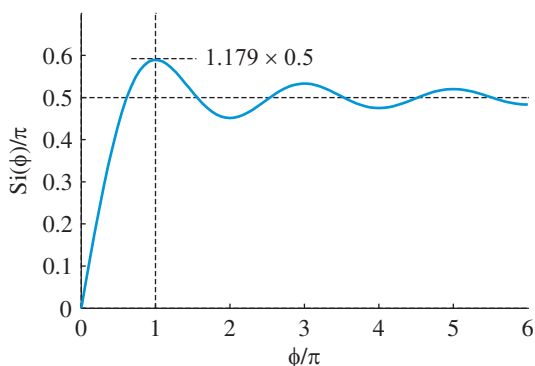
Figure 10.7 The sine integral function.

For $\omega > \omega_c$, we can show in a similar manner that (see Problem 24).

$$A\left(e^{j\omega}\right) \approx \frac{1}{2} - \frac{1}{\pi} \text{Si}\left[(\omega - \omega_c)L/2\right]. \tag{10.67}$$

The sine integral function, as shown in Figure 10.7, is linear for small values of ξ, that is, $\text{Si}(\xi) \approx \xi$, has a global maximum of value $1.179 \times 0.5\pi$ at $\xi = \pi$, and reaches an asymptotic value of 0.5π. Using these properties, (10.65), and (10.67) we obtain

$$\omega = \omega_c \qquad\qquad A\left(e^{j\omega}\right) = 0.5, \tag{10.68a}$$

$$\omega = \omega_c - 2\pi/L \qquad A\left(e^{j\omega}\right) = 0.5 + \text{Si}(\pi)/\pi \approx 1.0895, \tag{10.68b}$$

$$\omega = \omega_c + 2\pi/L \qquad A\left(e^{j\omega}\right) = 0.5 - \text{Si}(\pi)/\pi \approx 0.0895. \tag{10.68c}$$

Careful inspection of Figure 10.8, which shows the amplitude function in the vicinity of ω_c, and equation (10.68) leads to the following important observations:

1. Both the passband ripple δ_p and stopband attenuation δ_s are approximately equal, that is,

$$\delta_p \approx \delta_s \approx 0.0895, \tag{10.69}$$

irrespective of the order M, even if the design specifications require $\delta_p \neq \delta_s$.

2. Truncating the ideal impulse response yields FIR filters with passband ripple about $20 \log_{10}(1.0895) = 0.75$ dB and minimum stopband attenuation of about $20 \log_{10}(1/0.0895) = 21$ dB, irrespective of filter length; both are *not* sufficient for practical applications.

3. The width of the transition band, which is upper bounded by the width of the mainlobe of $A_w(e^{j\omega})$, is given by

$$\Delta\omega \triangleq \omega_s - \omega_p \approx \frac{1.8\pi}{M} \leq \frac{4\pi}{M+1}, \tag{10.70}$$

because, as illustrated in Figure 10.8, the transition band begins at $A(e^{j\omega}) = 1 - 0.0895$ and ends at $A(e^{j\omega}) = +0.0895$ which is about half the distance between the highest peak and the deepest valley of $A(e^{j\omega})$.
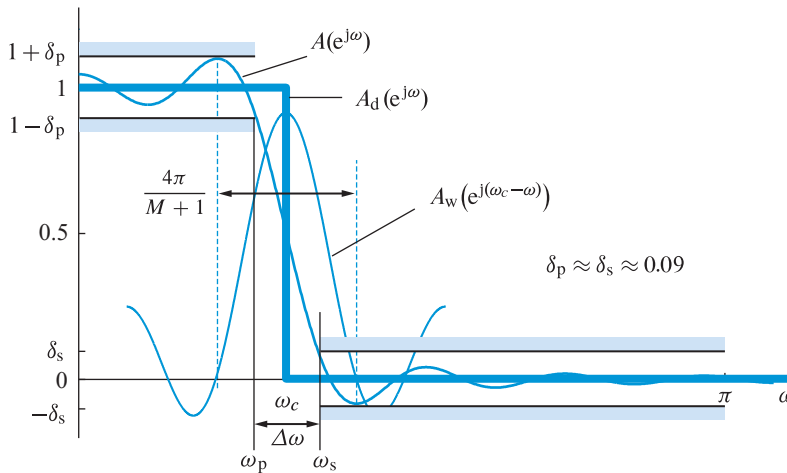
Figure 10.8 Implications of Gibbs phenomenon FIR filter design.

We notice that the passband and stopband ripples of the designed filter are determined by the integral of the window amplitude function $A_w(e^{j\omega})$. A simple way to determine these quantities numerically is from the plot of the *accumulated window amplitude function* (see Tutorial Problem 8):

$$A_{ac}(e^{j\omega}) \triangleq \int_{-\pi}^{\omega} A_w(e^{j\theta})d\theta. \tag{10.71}$$

Figure 10.9 shows plots of $A_w(e^{j\omega})$ and $A_{ac}(e^{j\omega})$ for rectangular windows with $M = 20$ and $M = 40$. We note that as we increase the order M, the level of the highest sidelobe remains constant at 13 dB and the stopband attenuation is fixed at about 21 dB. The widths of main lobe and therefore the transition band decrease as the order M increases. However, the only way to improve the stopband attenuation is to reduce the sidelobes of the window, which can be done *only* by changing the shape of the window. As we show in the rest of this section, there are nonrectangular windows that make possible the design of useful practical filters.

10.3.2   Smoothing the frequency response using fixed windows

Using nonrectangular windows to obtain a less abrupt truncation of the impulse response reduces the height of the ripples at the expense of a wider transition band. The most commonly used windows are defined by the following equations:

Rectangular

$$w[n] = \begin{cases} 1, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases} \tag{10.72}$$
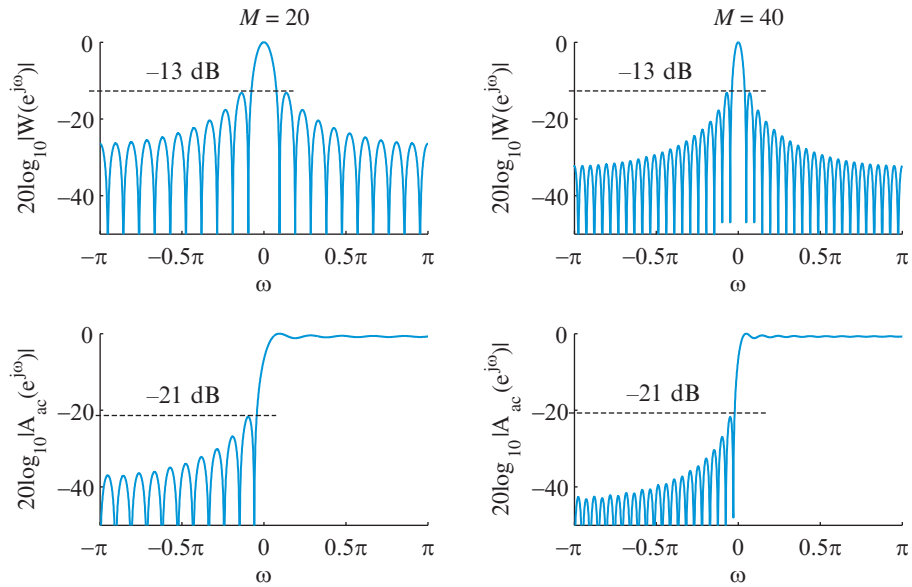
Figure 10.9 Fourier transform $W(e^{j\omega})$ of a rectangular window and its accumulated amplitude response $A_{ac}(e^{j\omega})$ for $M = 20$ and $M = 40$.

Bartlett (triangular)

$$w[n] = \begin{cases} 2n/M, & 0 \leq n \leq M/2, \quad M \text{ even} \\ 2 - 2n/M, & M/2 < n \leq M \\ 0, & \text{otherwise} \end{cases} \quad (10.73)$$

Hann

$$w[n] = \begin{cases} 0.5 - 0.5\cos(2\pi n/M), & 0 \leq n \leq M \\ 0, & \text{otherwise} \end{cases} \quad (10.74)$$

Hamming

$$w[n] = \begin{cases} 0.54 - 0.46\cos(2\pi n/M), & 0 \leq n \leq M \\ 0, & \text{otherwise} \end{cases} \quad (10.75)$$

Blackman

$$w[n] = \begin{cases} 0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M), & 0 \leq n \leq M \\ 0. & \text{otherwise} \end{cases} \quad (10.76)$$

Figure 10.10 shows the waveform shape and the magnitude of its Fourier transform (in dB) for these common windows for $M = 50$. We notice that, as expected, the nonrectangular
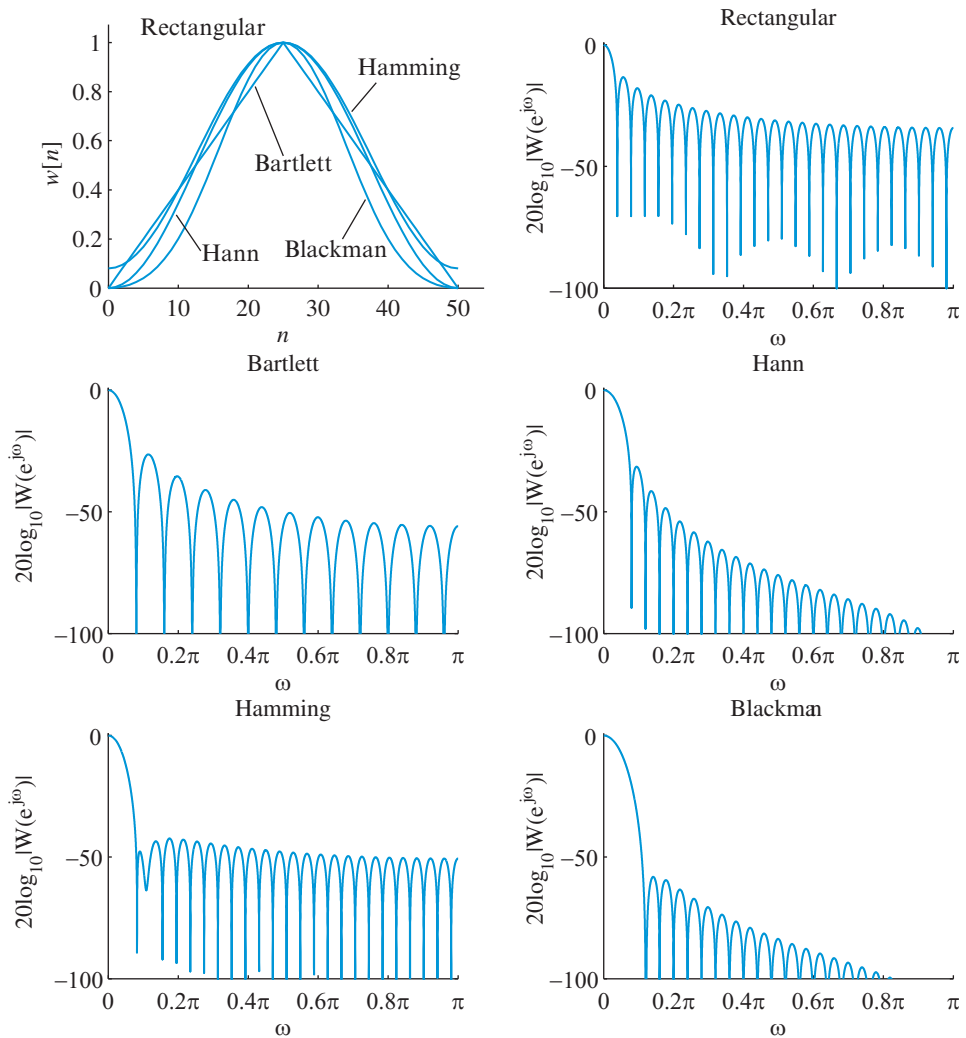
Figure 10.10 Time-domain and frequency-domain characteristics of some commonly used windows.

windows have wider mainlobes and lower sidelobes than the rectangular window. The Fourier transforms of the nonrectangular windows can be expressed in terms of the Fourier transform of rectangular windows (see Problems 25 and 43).

Windows are always symmetric, that is, they satisfy the condition

$$w[n] = \begin{cases} w[M - n], & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases} \tag{10.77}$$

for M even or odd. Therefore, the windowed impulse response $h[n] = w[n]h_d[n]$ has the same symmetry with the ideal impulse response $h_d[n]$; in other words, windowing an FIR filter with linear phase does *not* change its type.

Table 10.3 Properties of commonly used windows ($L = M + 1$).

Window name	Side lobe level (dB)	Approx. $\Delta\omega$	Exact $\Delta\omega$	$\delta_p \approx \delta_s$	A_p (dB)	A_s (dB)
Rectangular	−13	$4\pi/L$	$1.8\pi/L$	0.09	0.75	21
Bartlett	−25	$8\pi/L$	$6.1\pi/L$	0.05	0.45	26
Hann	−31	$8\pi/L$	$6.2\pi/L$	0.0063	0.055	44
Hamming	−41	$8\pi/L$	$6.6\pi/L$	0.0022	0.019	53
Blackman	−57	$12\pi/L$	$11\pi/L$	0.0002	0.0017	74

The effect of the window on the amplitude response can be analyzed following the approach used for the rectangular window. If we assume that $A_d(e^{j\omega})$ has a discontinuity at ω_c such that $A_d(e^{j\omega_c^-}) = 1$ and $A_d(e^{j\omega_c^+}) = 0$, we can show that in the vicinity of ω_c we have (see Problem 44)

$$A(e^{j\omega}) \approx \begin{cases} 0.5 + \frac{1}{\pi}\Lambda_w[0.5(\omega_c - \omega)L], & \omega < \omega_c \\ 0.5 - \frac{1}{\pi}\Lambda_w[0.5(\omega - \omega_c)L], & \omega > \omega_c \end{cases} \tag{10.78}$$

where $\Lambda_w(\phi)$ is the window "amplitude function integral" defined by

$$\Lambda_w(\phi) = \frac{1}{L}\int_0^\phi A_w(e^{j2\theta/L})d\theta. \tag{10.79}$$

Therefore, the passband and stopband ripples and the width of the transition band are determined by the running-integral of the amplitude window function $A_w(e^{j\omega})$. Furthermore, to a very good approximation, the passband ripple δ_p and stopband attenuation δ_s are equal and independent of M; they can be changed only by changing the shape of the window. The width of the transition band, which is controlled by the width of the mainlobe of $A_w(e^{j\omega})$, can be reduced by increasing the order M of the filter. Table 10.3 summarizes the properties of the windows defined in (10.72) through (10.76). These windows thus are termed "fixed" windows since their stopband attenuation is independent of window length. A careful inspection of the table shows that the Hamming window provides the best choice for filter design since it gives the best compromise between a narrow transition width and a high stopband attenuation.

Figure 10.11 shows the magnitude response of an FIR filter with $\omega_c = \pi/4$ and $M = 40$ designed using a rectangular, Hann, Hamming, and Blackman window. We note that the windows with smaller sidelobes and narrower mainlobe provide a better approximation of the ideal response around a discontinuity. A similar discussion of windows from the point of view of spectral analysis was given in Section 7.6.
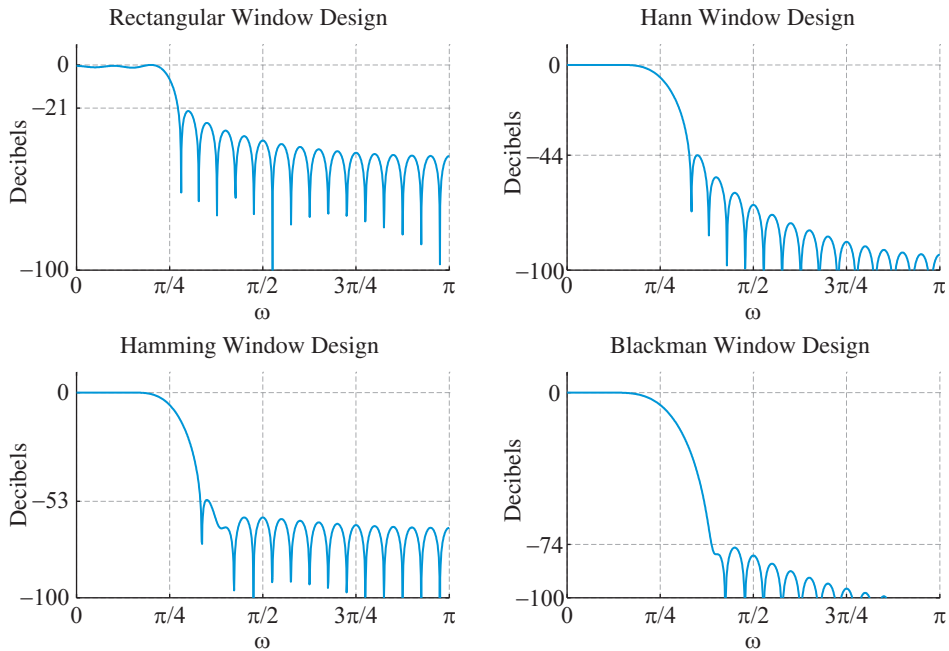
Figure 10.11 Magnitude responses in dB for 40th-order FIR lowpass filters designed using rectangular, Hann, Hamming, and Blackman windows with cutoff frequency $\omega_c = \pi/4$.

FIR filter design using fixed windows Using window functions and their parameters given in Table 10.3 we can now provide a systematic procedure for design of FIR filters. For lowpass filters these design steps are:

1. Given the design specifications $\{\omega_p, \omega_s, A_p, A_s\}$, determine the ripples δ_p and δ_s and set $\delta = \min\{\delta_p, \delta_s\}$.
2. Since the transition band is symmetric about ω_c (see Figure 10.8), determine the cutoff frequency of the ideal lowpass prototype by $\omega_c = (\omega_p + \omega_s)/2$.
3. Determine the design parameters $A = -20\log_{10}\delta$ and $\Delta\omega = \omega_s - \omega_p$.
4. From Table 10.3, choose the window function that provides the smallest stopband attenuation greater than A. For this window function, determine the required value of $M = L - 1$ by selecting the corresponding value of $\Delta\omega$ from the column labeled "exact $\Delta\omega$". If M is odd, we may increase it by one to have a flexible type-I filter.
5. Determine the impulse response of the ideal lowpass filter by

$$h_d[n] = \frac{\sin[\omega_c(n - M/2)]}{\pi(n - M/2)}. \tag{10.80}$$

6. Compute the impulse response $h[n] = h_d[n]w[n]$ using the chosen window.
7. Check whether the designed filter satisfies the prescribed specifications; if not, increase the order M and go back to step 5.

MATLAB provides several window functions for use in filter design. The functions that correspond to the windows given in Table 10.3 are: `rectwin(L)`, `bartlett(L)`, `hann(L)`, `hamming(L)`, and `blackman(L)`, respectively. The book toolbox function `ideallp(omc,M)` computes the ideal impulse given in (10.80).

Example 10.2

Let us design a lowpass linear-phase FIR filter to satisfy the following specifications:

$$\omega_p = 0.25\pi, \quad \omega_s = 0.35\pi, \quad A_p = 0.1 \text{ dB}, \quad A_s = 50 \text{ dB}. \qquad (10.81)$$

First, using (10.5), we obtain $\delta_p = 0.0058$ and $\delta_s = 0.0032$. Hence δ is set to 0.0032 or $A = A_p = 50$ dB. We then set the ideal lowpass filter cutoff frequency $\omega_c = (\omega_p+\omega_s)/2 = 0.3\pi$ and compute transition bandwidth $\Delta\omega = \omega_s - \omega_p = 0.1\pi$. From Table 10.3 we choose a Hamming window since it provides at least 53 dB attenuation which is greater than $A = 50$ dB. For this window, using the transition bandwidth $\Delta\omega \approx 6.6\pi/L$, the minimum window length is $L = 66$. We will choose $L = 67$ or $M = 66$ to obtain a type-I filter. Now that the necessary design parameters are chosen, we finally compute the impulse response $h[n]$ using (10.80) and (10.75). These steps can easily be implemented in MATLAB using the script:
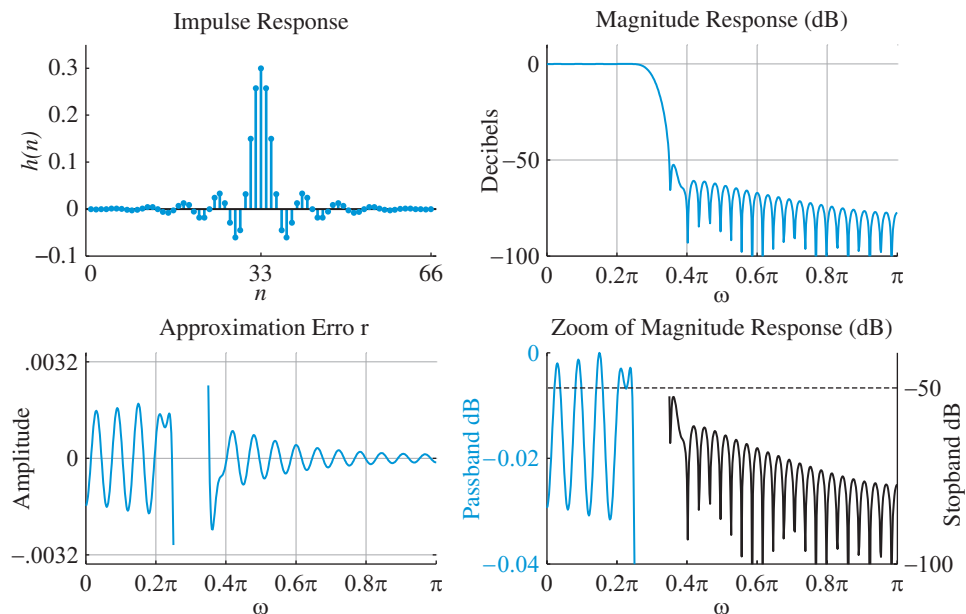
Figure 10.12 Impulse, approximation error, and magnitude response plots of the filter designed in Example 10.2 using a Hamming window to satisfy specifications: $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB.

```
>> wp = 0.25*pi; ws = 0.35*pi; Ap = 0.1; As = 50;
>> deltap = (10^(Ap/20)-1)/(10^(Ap/20)+1);
>> deltas = (1+deltap)/(10^(As/20));
>> delta = min(deltap,deltas); A = -20*log10(delta);
>> Deltaw = ws-wp; omegac = (ws+wp)/2;
>> L = ceil(6.6*pi/Deltaw)+1; M=L-1; % Window length and order
>> n = 0:M; hd = ideallp(omegac,M);
>> h = hd.*hamming(L)';
```

The plots of the impulse, approximation error, and magnitude responses of the designed filter are shown in Figure 10.12. From the approximation error and the zoomed magnitude response in dB we conclude that the designed filter satisfies the specifications given in (10.81). ∎

Since the transition band $\Delta\omega$ is proportional to $1/M$, we can control its width through the length of the window. However, since the size of the ripples depends on the shape of the window, there is no systematic way to control the passband ripple and stopband attenuation with fixed shape windows. The only way to address this problem is by using windows whose shape could be controlled with adjustable parameters. A practical solution to this problem, which was proposed by Kaiser (1974), is discussed in the next section.

10.3.3 Filter design using the adjustable Kaiser window

A window useful for filter design should have a Fourier transform that has a narrow main-lobe, a small relative peak sidelobe, and good side lobe roll-off. To a good approximation, the width of the mainlobe determines the width of the transition band, while the relative height of the sidelobes controls the size of ripples in the amplitude response. According to the uncertainty principle (see Section 7.6.3), there is a trade-off between main-lobe width and sidelobe height; in other words, we cannot reduce both quantities at the same time.

A class of optimum windows with adjustable parameters can be designed by maximizing the ratio of the energy in a frequency band about $\omega = 0$ over the total energy. For continuous-time windows this problem has been solved by Slepian (1978) in closed-form in terms of a complicated class of functions called *prolate spheroidal wave functions*. The solution of this problem in discrete-time leads to a numerically ill-conditioned eigenvalue problem. Kaiser (1974) avoided the solution of this problem by obtaining a simple, but sufficiently accurate, approximation to the prolate spheroidal window. Another approach seeks to minimize the relative peak sidelobe level; the solution is the Dolph–Chebyshev window, whose sidelobes all have the same level. A combination of these two approaches has been developed by Saramaki (1993). The Kaiser window, which is the standard for filter design, is defined by

$$w[n] = \begin{cases} \dfrac{I_0\left[\beta\sqrt{1 - [(n-\alpha)/\alpha]^2}\right]}{I_0(\beta)}, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases} \qquad (10.82)$$

```
function y = Izero(x,K)
% Computes y = I0(x) function using a summation of K terms
% y = Izero(x,K)
K = round(K); m = 1:K;
y = 1+ sum(((x/2).^m./factorial(m)).^2);
```

Figure 10.13 MATLAB function that computes $I_0(x)$.

where $\alpha = M/2$, and $I_0(\cdot)$ denotes the zeroth-order modified Bessel function of the first kind

$$I_0(x) = 1 + \sum_{m=1}^{\infty} \left[\frac{(x/2)^m}{m!} \right]. \tag{10.83}$$

This expression for $I_0(x)$ can be evaluated (to within specified accuracy) by the MATLAB function shown in Figure 10.13, using an algorithm due to Kaiser. The number of terms needed in (10.83) for practical convergence as well as the design of a MATLAB function called `kaiser0` to compute a Kaiser window function are investigated in Problem 40. MATLAB also provides the function `kaiser(L,beta)` to compute a length-L Kaiser window function given β.

As shown in Figure 10.14(a), the shape of the Kaiser window is controlled by the shape parameter β. Indeed, the Kaiser window can be used to approximate most of the fixed windows given in Table 10.3 (see Problem 41). The case $\beta = 0$ yields the rectangular window; however, as β increases, the tapering at the ends increases leading to a more concentrated window. The plots in Figure 10.14(b) show that as β increases, the mainlobe becomes wider, while the sidelobes become smaller. Figure 10.14(c) shows that increasing M while keeping β constant causes a decrease in the width of the mainlobe without affecting the peak amplitude of the sidelobes.

Since, to a good approximation, the stopband attenuation A_s does not depend on M, we can start the design process by finding the value of β required to obtain a prescribed value A for A_s. Through extensive numerical experimentation, Kaiser (1974) derived the following empirical relation:

$$\beta = \begin{cases} 0, & A < 21 \\ 0.5842(A - 21)^{0.4} + 0.07886(A - 21), & 21 \le A \le 50 \\ 0.1102(A - 8.7), & A > 50 \end{cases} \tag{10.84}$$

where $A = -20 \log_{10} \delta$ is the ripple height in dB. Since the windowing method produces filters with $\delta_p \approx \delta_s$, irrespective of the design specifications, we set $\delta = \max\{\delta_p, \delta_s\}$. Figure 10.14(d) shows β as a function of stopband attenuation; we note that the range $0 \le \beta \le 8$ provides useful windows. The case $\beta = 0$ corresponds to the rectangular window for which, as we have seen, the stopband attenuation is approximately $A_s = 21$ dB. By properly choosing β we can approximate other fixed windows. The order M required to achieve prescribed values of A and $\Delta\omega$ is estimated using the formula

$$M = \frac{A - 8}{2.285\Delta\omega}. \tag{10.85}$$
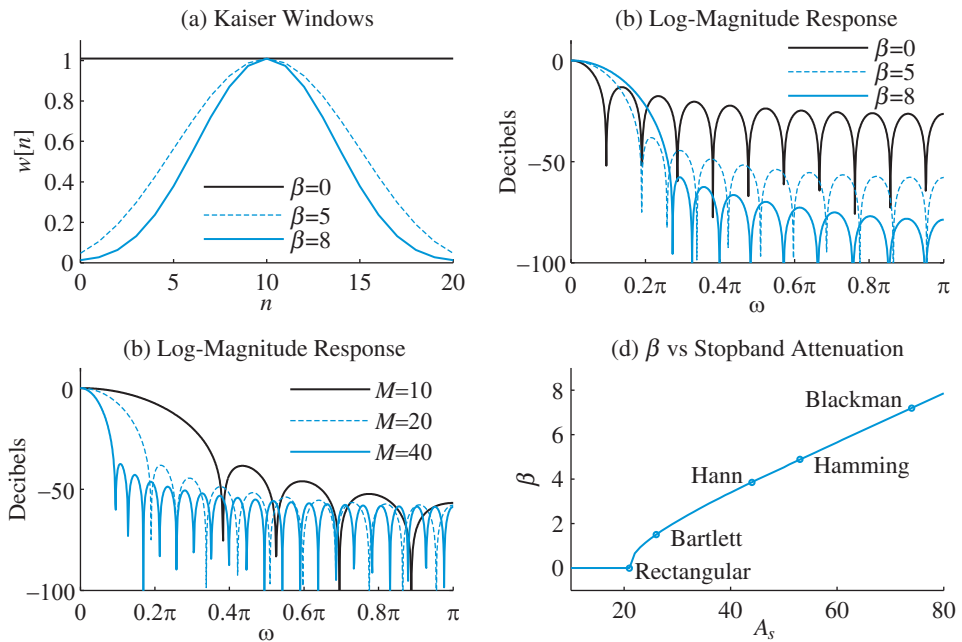
Figure 10.14 (a) Kaiser window function for $M = 20$ and $\beta = 0$, 5, and 8, (b) Log-magnitude responses for windows in (a), (c) Log-magnitude responses in dB for $\beta = 5$ and $M = 10$, 20, and 40, and (d) Stopband attenuation versus β function in which β parameters of Kaiser windows for equivalent fixed windows are identified.

Using these formulas we can develop a systematic procedure for design of FIR filters using the Kaiser window. For lowpass filters, this procedure has the following steps:

1. Given the design specifications $\{\omega_p, \omega_s, A_p, A_s\}$, determine the ripples δ_p and δ_s and set $\delta = \max\{\delta_p, \delta_s\}$.
2. Because the transition band is symmetric about ω_c, determine the cutoff frequency of the ideal lowpass prototype by $\omega_c = (\omega_p + \omega_s)/2$.
3. Determine the design parameters $A = -20\log_{10}\delta$ and $\Delta\omega = \omega_s - \omega_p$.
4. Determine the required values of β and M from (10.84) and (10.85), respectively. If M is odd, we may increase it by one to have a flexible type-I filter.
5. Determine the impulse response of the ideal lowpass filter using (10.80).
6. Compute the impulse response $h[n] = h_d[n]w[n]$ using the Kaiser window.
7. Check whether the designed filter satisfies the prescribed specifications; if not, increase the order M and go back to step 5.

Example 10.3

Consider the lowpass linear-phase FIR filter specifications given in (10.81): $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB. We will design the filter using Kaiser window. From Example 10.2 we have $A = 50$ dB and $\omega_c = (\omega_s + \omega_p)/2 = 0.3\pi$. Using (10.84) and (10.85) we obtain the required values of the Kaiser window parameters as
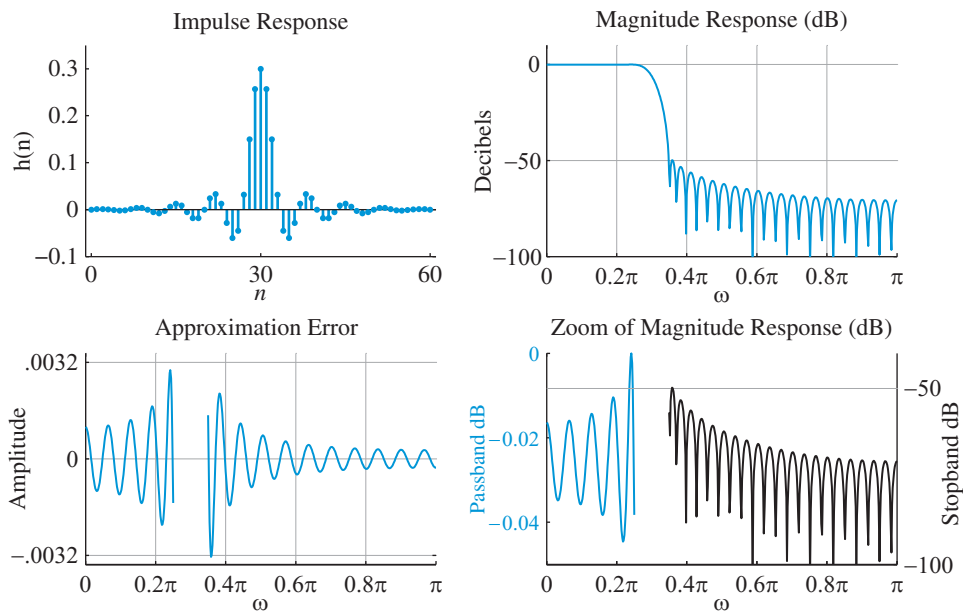
Figure 10.15 Impulse, approximation error, and magnitude response plots of the filter designed in Example 10.3 using a Kaiser window to satisfy specifications: $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB.

$\beta = 4.528$ and $M = 59$. We choose $M = 60$ for a type-I filter. We finally compute the impulse response $h[n]$ using (10.80) and (10.82). The MATLAB script relevant to design and implement the Kaiser window is given below:

```
>> beta = 0.5842*(A-21)^(0.4)+0.07886*(A-21); % Kaiser beta
>> M = ceil((A-8)/(2.285*Deltaw))+1; L = M+1; % Window length
>> alpha = M/2; n = 0:M;
>> hd = wc*sinc(wc*(n-alpha)); h = hd.*kaiser(L,beta)';
```

The plots of the impulse, approximation error, and magnitude responses of the designed filter are shown in Figure 10.15. From the approximation error and the zoomed magnitude response in dB we conclude that the designed filter satisfies the specifications given in (10.81). Furthermore, the Kaiser window design satisfies specifications using a length $L = 61$ window which is smaller than $L = 67$ for the Hamming window. ∎

The above approach can be generalized to the design of highpass, bandpass, bandstop, and other multiband filters, as long as we can derive an analytical expression for the required ideal impulse response. For example, consider an ideal bandpass filter with linear phase and unit passband $0 \leq \omega_{c_1} < |\omega| < \omega_{c_2} \leq \pi$. Since this filter is equivalent to the difference of two lowpass filters with cutoff frequencies ω_{c_2} and ω_{c_1}, its impulse response is given by

$$h_{bp}[n] = \frac{\sin[\omega_{c_2}(n - M/2)]}{\pi(n - M/2)} - \frac{\sin[\omega_{c_1}(n - M/2)]}{\pi(n - M/2)}. \tag{10.86}$$
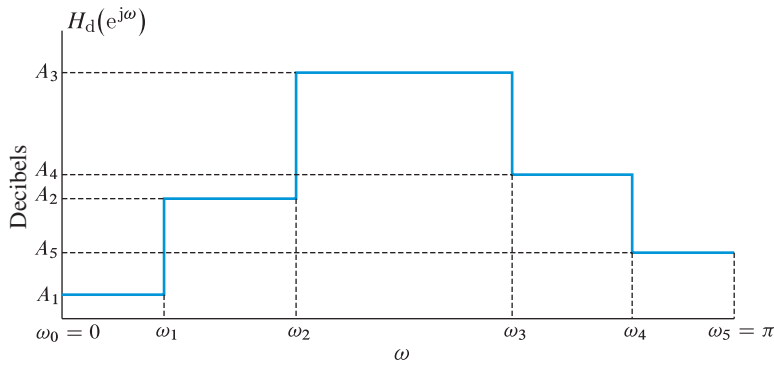
Figure 10.16 An example of the magnitude response of a 5-band ideal multiband filter.

The case $\omega_{c_2} = \pi$ corresponds to a highpass filter. In general, the impulse response of a multiband filter (see Figure 10.16 for an example) is given by

$$h_{mb}[n] = \sum_{k=1}^{K} (A_k - A_{k+1}) \frac{\sin[\omega_k(n - M/2)]}{\pi(n - M/2)}, \tag{10.87}$$

where $\left| H_{mb}(e^{j\omega}) \right| = A_k$ for $\omega_{k-1} \leq \omega \leq \omega_k$, $k = 1, 2, \ldots, K$, $A_{K+1} = 0$, and K is the number of bands. The behavior at each transition band is similar to that shown in Figure 10.8 and can be accurately characterized by formulas (10.78) and (10.79), as long as the frequencies ω_k are far enough apart and the ripples at each band are properly scaled by the corresponding gain A_k (see Tutorial Problem 9 for details).

Example 10.4

In this example we will design a bandpass filter using a Kaiser window to satisfy the following specifications:

$$\left| H(e^{j\omega}) \right| \leq 0.01, \qquad\qquad |\omega| \leq 0.2\pi$$

$$0.99 \leq \left| H(e^{j\omega}) \right| \leq 1.01, \quad 0.3\pi \leq |\omega| \leq 0.7\pi \tag{10.88}$$

$$\left| H(e^{j\omega}) \right| \leq 0.01. \quad 0.78\pi \leq |\omega| \leq \pi$$

In multiband filters, there are more than two band ripple parameters and more than one transition band. From the discussion on the creation of a transition band as well as passband/stopband ripples due to window response function, it should be clear that the window design approach can satisfy only one ripple parameter and one transition bandwidth. Hence we design for the minimum of the desired values.

In this example, ripple parameter is $\delta = 0.01$ or $A \approx 40$ dB. There are two transition bands: $0.2\pi \leq \omega \leq 0.3\pi$ and $0.7\pi \leq \omega \leq 0.78\pi$ for which the cutoff frequencies
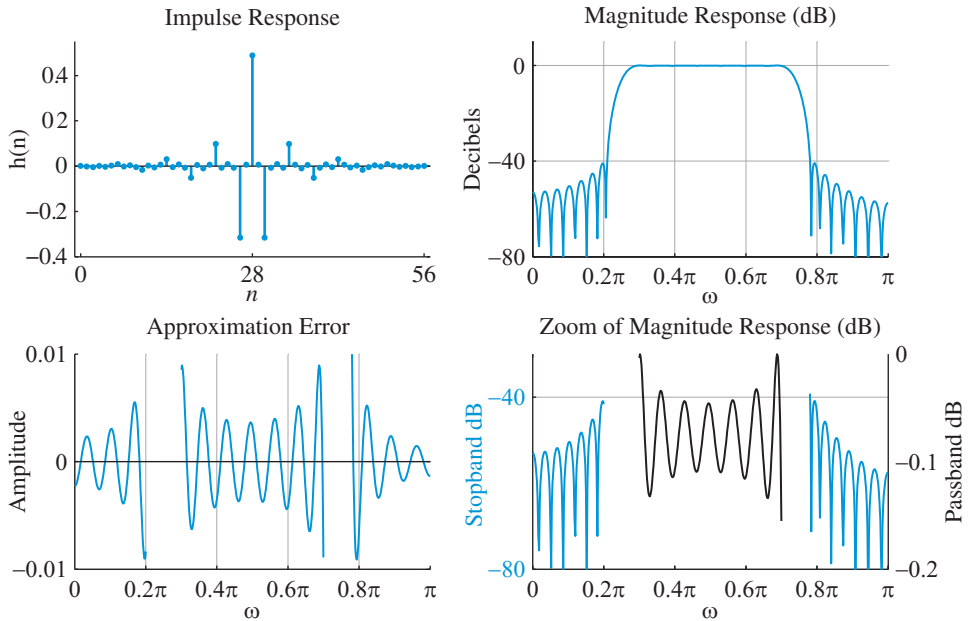
Figure 10.17 Impulse and frequency response plots of the bandpass filter designed in Example 10.4 using a Kaiser window to satisfy specifications: $\omega_{s_1} = 0.2\pi$, $\omega_{p_1} = 0.3\pi$, $\omega_{p_2} = 0.7\pi$, $\omega_{s_2} = 0.78\pi$, $\delta_s = \delta_p = 0.01$.

are $\omega_{c_1} = 0.25\pi$ and $\omega_{c_2} = 0.74\pi$, respectively. Similarly the transition bandwidths are $\Delta\omega_1 = 0.3\pi - 0.2\pi = 0.1\pi$ and $\Delta\omega_2 = 0.78\pi - 0.7\pi = 0.08\pi$. We set $\Delta\omega = 0.08\pi$. Using (10.84) and (10.85) the required values of the Kaiser window parameters are $\beta = 3.3953$ and $M = 56$ which we choose for a type-I filter. The impulse response $h[n]$ is now computed using (10.80) and (10.82).

The plots of the impulse, approximation error, and magnitude responses of the designed filter are shown in Figure 10.17. From the approximation error and the zoomed magnitude response in dB we conclude that the designed filter satisfies the specifications given in (10.88). ∎

MATLAB functions for window design In addition to the window functions given in Section 10.3.2, MATLAB provides the function `fir1` to design standard multiband (lowpass, bandpass, etc.) FIR filters using a window design approach in which a Hamming window is the default window. For example, the fragment `h=fir1(66,0.3)` designs the 66th-order lowpass filter given in Example 10.2 with cutoff frequency 0.3π. We still need to determine the filter order and cutoff frequency given the filter specifications. Similarly, the fragment `h=fir1(60,0.3, kaiser(61,4.528)')` designs the length 61 lowpass filter via the Kaiser window given in Example 10.3 with cutoff frequency 0.3π and $\beta = 4.528$. For Kaiser window design the `kaiserord` function provides the order M, β, cutoff frequency ω_c and the type of frequency selective filter.
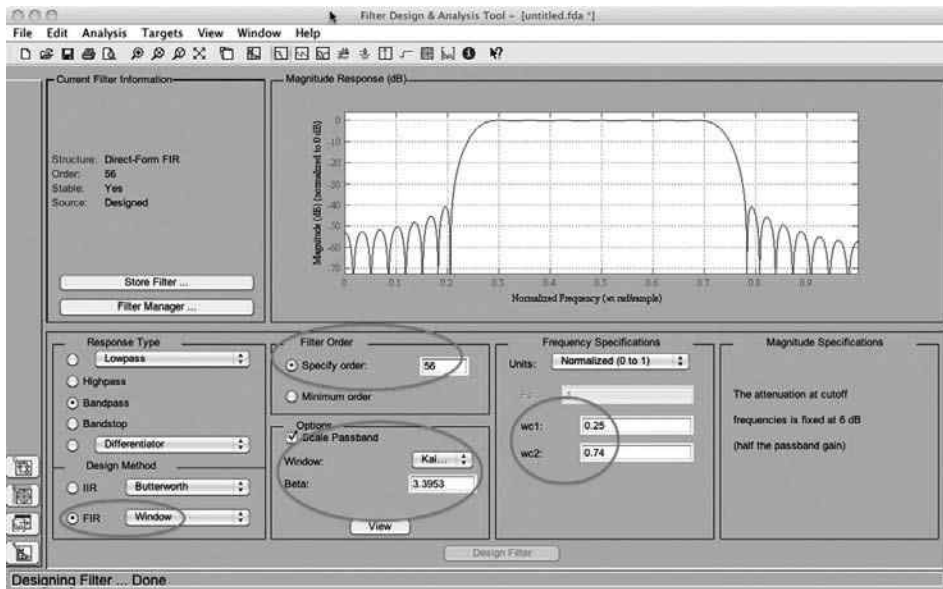
Figure 10.18 Graphical user interface of the FDATool for designing FIR filters via window design method.

For example, the following script is an alternate approach to design the lowpass filter of Example 10.3:

```
>> [M,wc,beta,ftype] = kaiserord([0.25,0.35],[1,0],...
      [deltap,deltas]);
>> h = fir1(M,wc,ftype,kaiser(M+1,beta));
```

The `fir1` is a versatile function that can be used to design multiband filters and MATLAB's SP toolbox manual should be consulted for its numerous invocations.

FDATool for window design The SP toolbox in MATLAB contains a GUI-based tool for designing FIR and IIR digital filters that makes designing them a convenient and straightforward task. It is invoked by `fdatool` and the resulting interface displays several sub-panels that provide user interactive areas for entering needed parameters and choices. Figure 10.18 shows the graphical user interface for designing the bandpass filter of Example 10.4 using the Kaiser window. In the Design Method panel the Window option is selected after clicking the FIR radio button. The Options panel is used in selecting the Kaiser window and its β parameter while the filter order is selected in the Filter Order panel. Finally the cutoff frequencies are specified in the Frequency Specifications panel in the normalized fashion. Thus this graphical user interface eliminates command window based design steps, however, it does not eliminate our need for computations of critical parameters like cutoff frequencies or β, etc. from the given specifications.

The windowing method is widely used because it is simple to understand, easy to use, and provides practically useful filters; however, it also has some important limitations. First, filters designed by windowing have approximately equal size ripples in each band.

Thus, we cannot weight the passband and stopband ripples differently, even if most applications require smaller ripples in the stopband. Second, because of the frequency convolution operation, we cannot specify the edges and maximum ripple size of each band precisely. Third, it is difficult to find analytical expressions for the ideal impulse response for arbitrary desired frequency responses. Finally, the windowing method is sub-optimal because the designed filters do not satisfy any clear optimality criterion; only the use of a rectangular window minimizes the mean square error.

10.4   Design of FIR filters by frequency sampling

The windowing method requires an analytical expression for the desired frequency response $H_\mathrm{d}(e^{j\omega})$, and the impulse response $h_\mathrm{d}[n]$ is obtained from the inverse Fourier transform (4.91). However, in certain applications, the desired filter is specified by samples of its frequency response function without necessarily knowing an analytical expression for $H_\mathrm{d}(e^{j\omega})$. In this section, we discuss filter design techniques based upon sampling the desired frequency response function.

Basic design approach Suppose that we are given samples of a desired frequency response at L equally spaced points on the unit circle

$$H_\mathrm{d}[k] \triangleq H_\mathrm{d}(e^{j2\pi k/L}). \quad k = 0, 1, \ldots, L-1 \tag{10.89}$$

The desired impulse response $h_\mathrm{d}[n]$, which is not available, may have finite or infinite duration. The inverse DFT of $H_\mathrm{d}[k]$ is related to $h_\mathrm{d}[n]$ by the aliasing relation (see Section 7.3 regarding the following discussion)

$$\tilde{h}[n] \triangleq \frac{1}{L} \sum_{k=0}^{L-1} H_\mathrm{d}[k]W_N^{-kn} = \sum_{m=-\infty}^{\infty} h_\mathrm{d}[n-mL], \tag{10.90}$$

which is a periodic sequence with fundamental period L. We can design an FIR filter by multiplying $\tilde{h}[n]$ with a window of length L, that is

$$h[n] = \tilde{h}[n]w[n]. \tag{10.91}$$

Since $\tilde{h}[n]$ is periodic, the frequency response of the designed filter is obtained using (7.61)

$$H(e^{j\omega}) = \frac{1}{L} \sum_{k=0}^{L-1} H_\mathrm{d}[k]W(e^{j(\omega-2\pi k/L)}), \tag{10.92}$$

where $W(e^{j\omega})$ is the Fourier transform of the window. Thus, the frequency response of the designed filter is obtained by interpolating between the samples $H_\mathrm{d}[k]$ using $W(e^{j\omega})$ as an interpolation function. The interpolation process (10.92) acts similarly to the frequency-domain convolution (10.59) in the windowing method. The windowing method creates

an FIR filter by truncating $h_d[n]$; the frequency sampling method creates an FIR filter by aliasing or folding $h_d[n]$.

If $w[n]$ is an L-point rectangular window then $h[n]$ is the primary period of $\tilde{h}[n]$. In this case the approximation error is zero at the sampling frequencies, and finite between them because $W(e^{j\omega})$ is a Dirichlet (periodic sinc) function. For nonrectangular windows, the approximation error is not exactly zero at the sampling frequencies but is very close to zero, however, they have advantages in reducing passband and stopband ripples. In practice, to minimize the time-domain aliasing distortion, we start with a large value for L in (10.89), and then we choose a much smaller value for the length of the window in (10.91) (see Tutorial Problem 11).

Linear-phase FIR filter design In general, the approach of (10.89)–(10.91) results in FIR filters with arbitrary phase. To design FIR filters with linear phase we should incorporate the appropriate constraints into the design equations. To assure that the sequence $h[n]$ is real and satisfies the linear phase constraints

$$h[n] = \pm h[L-1-n] \tag{10.93}$$

we should form the DFT coefficients $H_d[k]$ from the samples of the desired response very carefully.

This is done using the following formulas (see Table 10.1):

$$H_d[k] = A_d[k]e^{j\Psi_d[k]}, \tag{10.94a}$$

$$A_d[k] = \begin{cases} A_d(e^{j0}), & k = 0 \\ A_d(e^{j2\pi(L-k)/L}), & k = 1, 2, \dots, L \end{cases} \tag{10.94b}$$

$$\Psi_d[k] = \begin{cases} \pm\dfrac{\pi}{2}q - \dfrac{L-1}{2}\dfrac{2\pi}{L}k, & k = 0, 1, \dots, Q \\ \mp\dfrac{\pi}{2}q + \dfrac{L-1}{2}\dfrac{2\pi}{L}(L-k), & k = Q+1, \dots, L-1 \end{cases} \tag{10.94c}$$

where $Q = \lfloor (L-1)/2 \rfloor$. The parameter $q = 0$ for type I–II FIR filters, and $q = 1$ for type III–IV FIR filters. The impulse response $h[n]$ is obtained using (10.90) and (10.91) with a rectangular window. The frequency-domain samples $H_d[k]$ of a typical ideal lowpass filter and the resulting interpolated frequency response $H(e^{j\omega})$ are illustrated in Figure 10.19, in which the cutoff frequency is 0.45π. Observe that $H(e^{j\omega})$ is zero at the sampling frequencies and the approximation error is larger near the sharp transition and is smaller away from it. This is because of the Gibbs phenomenon (due to a rectangular window) created by the sharp transition between passband and stopband. The maximum approximation error thus depends on the shape of the ideal frequency response; it is smaller for a smoother transition and vice versa.

Better design approaches Since the transition band is typically unspecified, we can improve the quality of the approximation by enforcing a smoother transition band. There are several approaches; one is optimal in the sense of obtaining the smallest ripple for the given length, while other approaches are sub-optimal yet produce superior results.

Sharp Transition: $M = 19$
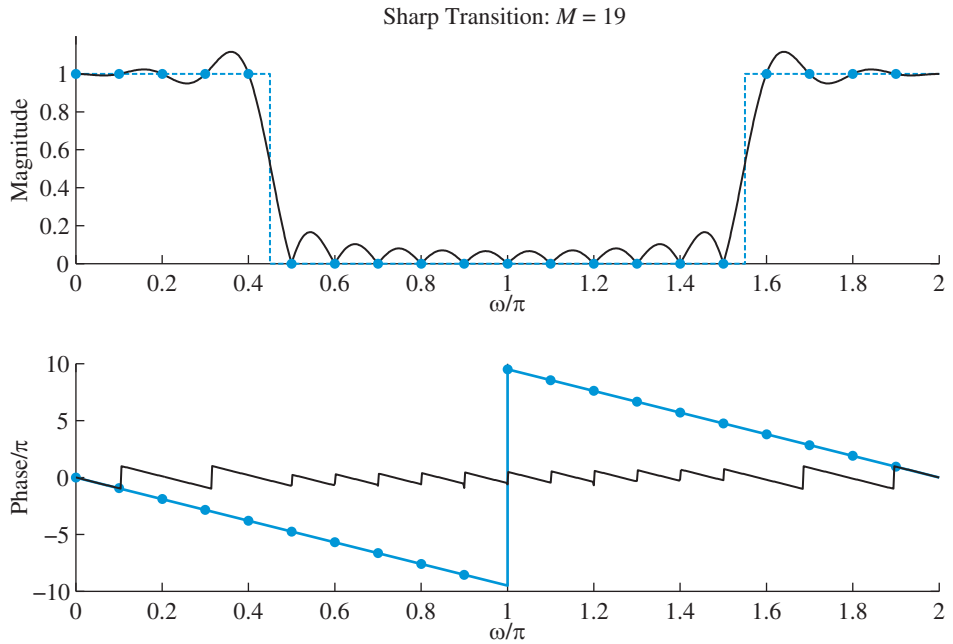
Figure 10.19 Frequency sampling FIR filter design technique. Note that the samples of the frequency response transition sharply from passband to stopband.

Optimal design approach   One approach is to make a small number of frequency samples in the transition band unconstrained variables. Since $H(e^{j\omega})$ is a linear combination of the samples $H_d[k]$, Rabiner *et al.* (1970) used linear programming optimization to determine the values of unconstrained samples to minimize the peak approximation error. This paper provides tables to choose transition band samples for a select combination of filter length L, passband width ω_p, and stopband attenuation A_s. The use of this optimal design approach is explained in Tutorial Problem 12.

Smooth transition band approach   In this approach, discussed in Burrus *et al.* (1992), a smooth transition band is introduced that makes $A_d(e^{j\omega})$ continuous; this eliminates the Gibbs phenomenon and causes $h[n]$ to decay asymptotically faster than $h_d[n]$. Spline functions (Unser (1999)) provide a very attractive choice for transition functions. However, a straight-line roll-off (first-order spline)

$$A_d(e^{j\omega}) = (\omega_s - \omega)/(\omega_s - \omega_p), \quad \omega_p < \omega < \omega_s \tag{10.95}$$

or a raised-cosine roll-off

$$A_d(e^{j\omega}) = 0.5 + 0.5\cos[\pi(\omega_s - \omega)/(\omega_s - \omega_p)], \quad \omega_p < \omega < \omega_s \tag{10.96}$$

is sufficient for many applications.

(a) Linear Transition: *M* = 19

(b) Raised-Cosine Transition: *M* = 19
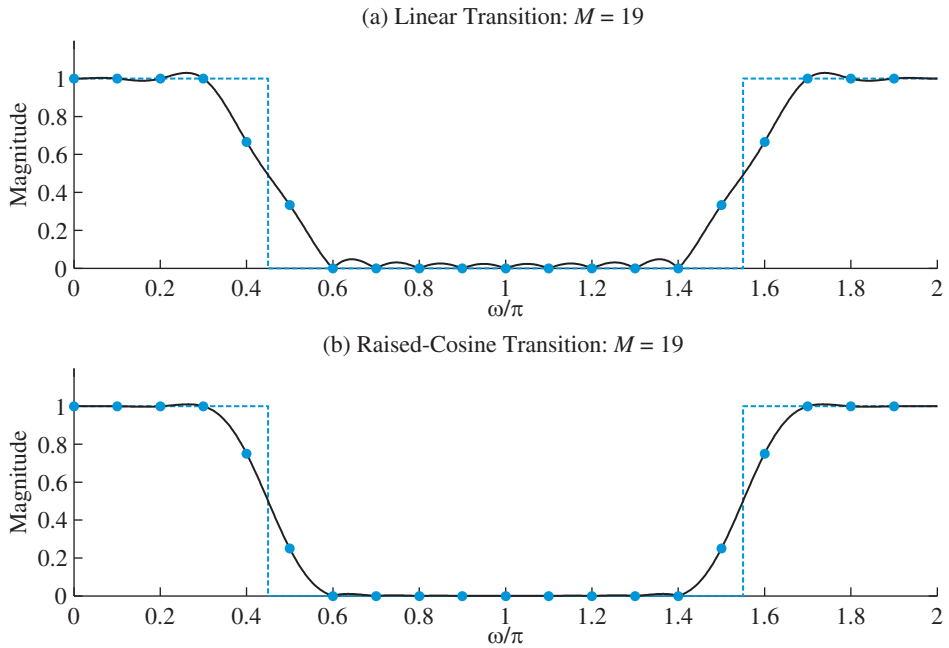
Figure 10.20 Frequency sampling FIR filter design technique in which the transition of the frequency response samples is (a) linear, and (b) raised-cosine.

Figure 10.20 shows an example of a frequency-sampling design approach using (10.95) and (10.96). One sample each on each side of the sharp ideal transition in Figure 10.19 is adjusted according to (10.95) in Figure 10.20(a) to produce a linear transition band, while in Figure 10.20(b) these transition-band samples are adjusted according to (10.96) to produce a raised-cosine shape. Notice a substantial reduction in passband and stopband ripples over the design in Figure 10.19. The ripples obtained using the raised-cosine approach are almost nonnoticeable. The main drawback of this smooth transition-band approach is to create a wider transition bandwidth in the resulting design which can be reduced by increasing the order *M*.

Nonrectangular window design approach As stated above in the basic design approach, we also can obtain the FIR filter impulse response $h[n]$ by multiplying the periodic $\tilde{h}[n]$ by a nonrectangular window $w[n]$. Although any window function discussed in Section 10.3 can be used, the most popular are Hamming and Kaiser windows. Figure 10.21(a) shows the effect of a Hamming window on the basic frequency-sampling design technique. The resulting frequency response is similar to the one obtained using the raised-cosine smooth transition in Figure 10.20(b) along with the wider transition bandwidth. Figure 10.21(b) shows the equivalent result using a Kaiser window with $\beta = 4$. Note that in both cases the resulting frequency response does not go through samples of the desired frequency response near the transition band, while the approximation is very close for samples away from the transition. For many applications this design approach is acceptable and can be improved using a higher *M* value and an appropriate window function $w[n]$.
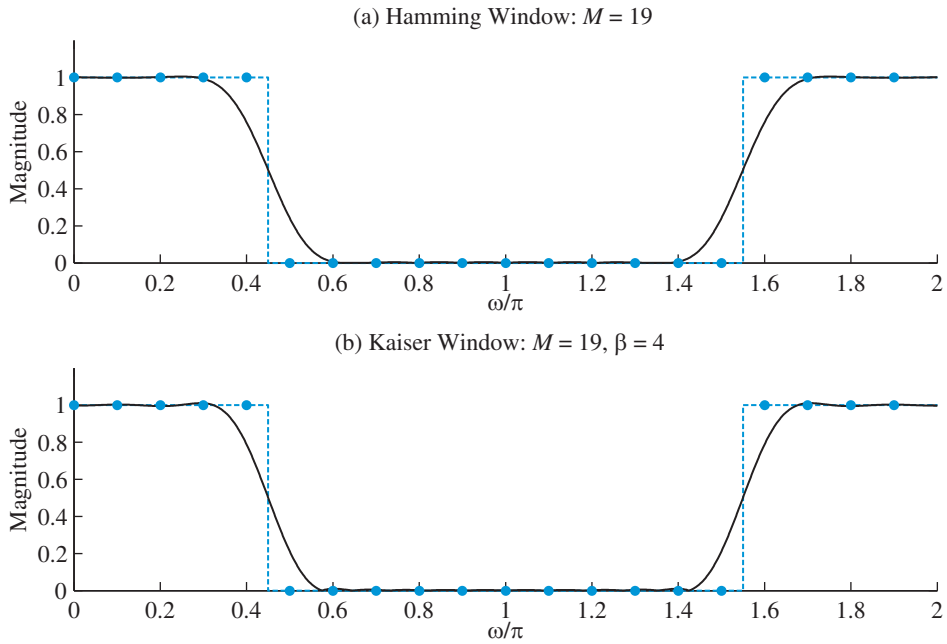
Figure 10.21 Frequency sampling FIR filter design technique using windows: (a) Hamming window, and (b) Kaiser window ($\beta = 4$).

Design procedure The frequency sampling design procedure is straightforward for arbitrary but well-defined desired frequency responses as explained above. For frequency selective filters with undefined transition bands, the design procedure can be iterative if the optimal design approach is not used. For example, to design a lowpass filter with specifications ω_p, ω_s, A_p, and A_s, the following steps can be used:

1. Choose the order of the filter M by placing at least two samples in the transition band.
2. For a window design approach obtain samples of the desired frequency response $H_d[k]$ using (10.94). For a smooth transition band approach, use (10.95) or (10.96) for transition band samples in addition to (10.94) for remaining samples.
3. Compute the $(M+1)$-point IDFT of $H_d[k]$ to obtain $h[n]$. For a window design approach multiply $h[n]$ by the appropriate window function.
4. Compute log-magnitude response $H_d(e^{j\omega})$ and verify the design over passband and stopband.
5. If the specifications are not met, increase M and go back to step 1.

Example 10.5 Lowpass filter design

Consider the following specifications of a lowpass filter:

$$\omega_p = 0.25\pi, \quad \omega_s = 0.35\pi, \quad A_p = 0.1 \text{ dB}, \quad A_s = 40 \text{ dB}.$$

From Figure 10.20, it should be obvious that we need two or more samples in the transition band to substantially reduce ripples in the passband and stopband. Since the transition bandwidth is 0.1π, we will need more than 40 samples of the ideal frequency response. Choosing $M = 44$ and using the raised-cosine transition band samples, the design steps in MATLAB are:

```
M = 44; L = M+1; % Impulse response length
alpha = M/2; Q = floor(alpha); % phase delay parameters
om = linspace(0,2*pi,1001); % Frequency array
k = 0:M; % Frequency sample index
psid = -alpha*2*pi/L*[(0:Q),-(L-(Q+1:M))]; % Desired Phase
Dw = 2*pi/L; % Width between frequency samples
% Design
k1 = floor(wp/Dw); % Index for sample nearest to PB edge
k2 = ceil(ws/Dw);   % Index for sample nearest to SB edge
w = ((k2-1):-1:(k1+1))*Dw; % Freq in the transition band
A = 0.5+0.5*cos(pi*(ws-w)/(ws-wp)); % Transition band samples
B = fliplr(A); % Transition band samples for omega >pi
Ad = [ones(1,k1+1),A,zeros(1,L-2*k2+1),...
    B,ones(1,k1)]; % Desired Amplitude
Hd = Ad.*exp(1j*psid); % Desired Freq Resp Samples
hd = real(ifft(Hd)); % Desired Impulse response
h = hd.*rectwin(L)'; % Actual Impulse response
H = freqz(h,1,om); % Frequency response of the actual filter
```

The stopband attenuation in decibels of the designed filter was measured using the script:

```
maxmag = max(abs(H)); dw = 2*pi/1000;
Asd = min(-20*log10(abs(H(ceil(ws/dw):501))/maxmag))
Asd =
    33.1507
```

The measured value of the stopband attenuation for this design was 33.2 dB, which did not satisfy the given specifications. Clearly, we need to increase M or use another approach. Since a Hamming window provides more than 50 dB of attenuation, we used it in the following script:

```
M = 44; L = M+1; % Impulse response length
alpha = M/2; Q = floor(alpha); % phase delay parameters
om = linspace(0,2*pi,1001); % Frequency array
k = 0:M; % Frequency sample index
psid = -alpha*2*pi/L*[(0:Q),-(L-(Q+1:M))]; % Desired Phase
Dw = 2*pi/L; % Width between frequency samples
% Design
omc = (wp+ws)/2; % Cutoff frequency
k1 = floor(omc/Dw); % Left sample index nearest to cutoff edge
k2 = ceil(omc/Dw);   % Right sample index nearest to cutoff edge
```
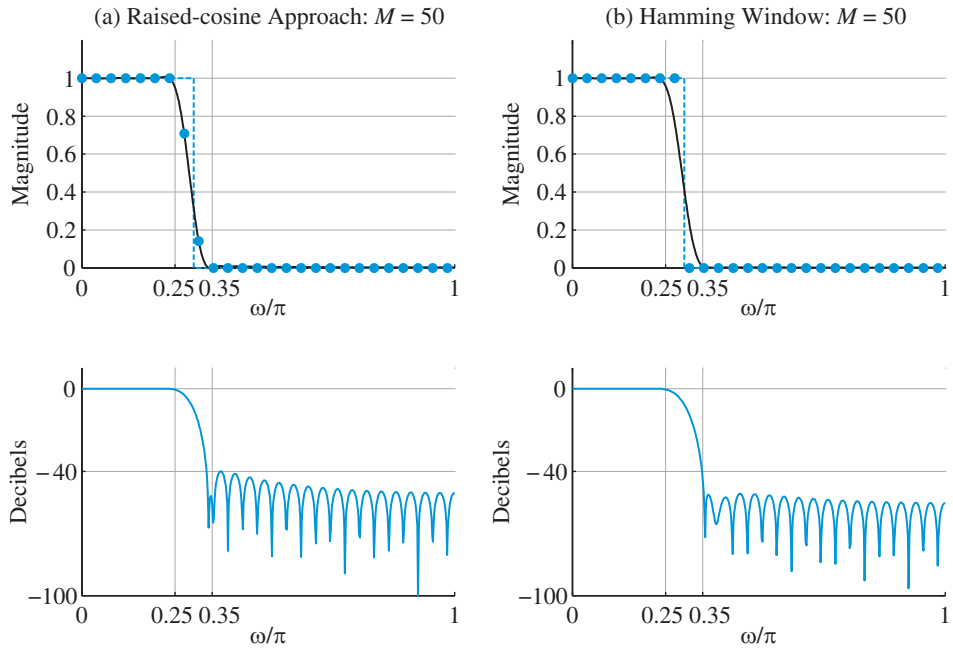
Figure 10.22 Magnitude and log-magnitude responses of the lowpass FIR filter design in Example 10.5: (a) raised-cosine approach, and (b) Hamming window approach.

```
Ad = [ones(1,k1+1),zeros(1,L-2*k2+1),...
    ones(1,k1)]; % Desired Amplitude
Hd = Ad.*exp(1j*psid); % Desired Freq Resp Samples
hd = real(ifft(Hd)); % Desired Impulse response
h = hd.*hamming(L)'; % Kaiser window Impulse response
H = freqz(h,1,om); % Frequency response of the actual filter
```

The measured value of the stopband attenuation was 37.4 dB, which was better than the previous approach but still was not sufficient. By increasing the value of M and checking for the resulting stopband attenuation, we obtained $M = 50$ for both approaches that satisfied the given specifications. The measured values were $A_s = 40$ dB for the raised-cosine window design and $A_s = 40.4$ dB for the Hamming window design. Figure 10.22 shows the magnitude response plots of the designed filters, samples of the desired frequency responses and the log-magnitude responses in dB.   ■

Example 10.5 shows that the frequency sampling technique is not well suited to standard frequency-selective filter design, such as lowpass, bandpass, etc., because it is difficult, if not impossible, to determine a priori the number of samples needed for the correct design. The following example illustrates an application where the frequency sampling technique is most appropriate.

Example 10.6   DAC equalization

In Section 6.5.2 we discussed the operation of digital-to-analog conversion to obtain ana-
log signals. It uses a sample-and-hold system whose frequency response is a sinc function
shown in Figure 6.26 and given in (6.66). It replaces the ideal lowpass filter but its fre-
quency response is not flat and attenuates the analog signal at higher frequencies. For
example, at 80% of the folding frequency $F_s/2$ the frequency response is attenuated by
$-20\log_{10}(0.8/2) = 2.42$ dB. That loss is unacceptable for many broadband applications
requiring a flat frequency response.

One approach is to use an analog lowpass post-filter given in (6.67) which provides
an inverse sinc compensation over the band of interest. Another approach is to perform
this compensation, known as DAC equalization, using a digital filter prior to the DAC
operation. Since the desired frequency response of the digital filter is completely known
and because its inverse Fourier transform does not have a closed-form expression, the
frequency sampling design is an appropriate technique to design a DAC equalizer. From
(6.67) the frequency response of the fullband equalizer is given by

$$H_d(e^{j\omega}) = \frac{\omega/2}{\sin(\omega/2)}. \qquad -\pi \le \omega \le \pi \qquad (10.97)$$

Since the filter response at $\omega = \pm\pi$ is not zero, the designed FIR filter should have an even
order. The following MATLAB script uses $M = 40$ and designs a linear-phase FIR filter
using the basic approach:

```
% Desired DAC Equalizer response
M = 40; L = M+1; % Impulse response length
k = -M/2:M/2; Ad = 1./sinc(k/L); % Hd over -pi < omega <= pi
Ad = ifftshift(Ad); % Hd over 0 <= omega < 2*pi
alpha = M/2; Q = floor(alpha); % phase delay parameters
om = linspace(0,2*pi,1001); % Frequency array
k = 0:M; % Frequency sample index
psid = -alpha*2*pi/L*[(0:Q),-(L-(Q+1:M))]; % Desired Phase
Hd = Ad.*exp(1j*psid); % Desired Freq Resp Samples
hd = real(ifft(Hd)); % Desired Impulse response
h = hd.*rectwin(L)'; % Actual Impulse response
H = freqz(h,1,om); % Frequency response of the actual filter
```

Figure 10.23(a) shows the magnitude response of the DCA equalizer using a basic
approach, while 10.23(b) shows the similar design using a Hamming window. In both cases
the design is satisfactory. Such equalizers eliminate the need for the analog compensator
and simplify design of the analog post-filter.   ■

MATLAB functions for frequency sampling design MATLAB provides the function
`fir2` that designs linear-phase FIR filters of arbitrary frequency response using a win-
dow design approach in which the Hamming window is the default window. For a basic
approach, the window function `rectwin(L)` should be used.
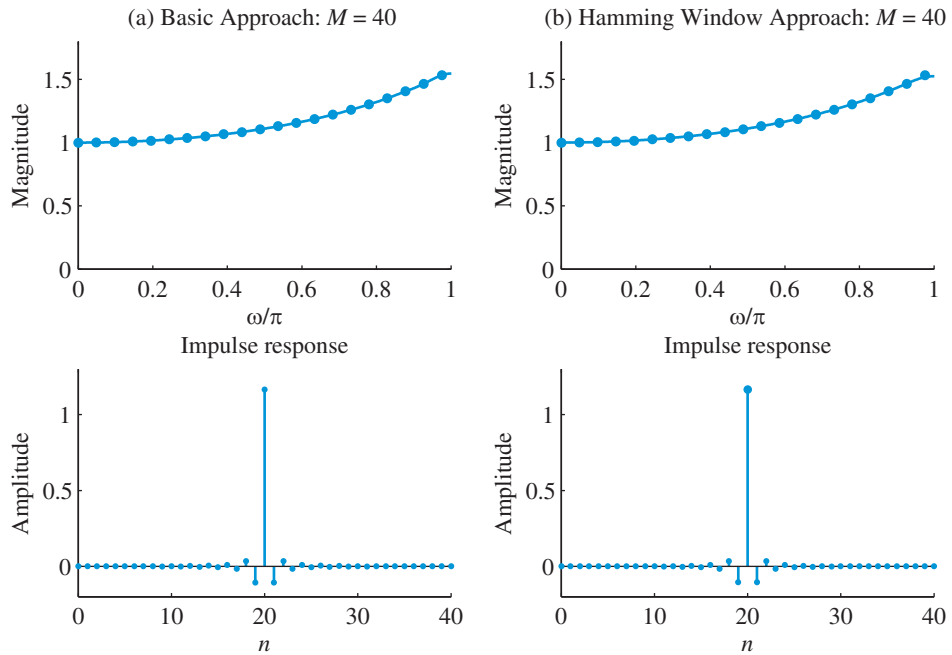
Figure 10.23 Magnitude and impulse responses of the FIR DAC equalizer design in Example 10.6: (a) basic approach, and (b) Hamming window approach.

The minimal invocation `h=fir2(M,f,A)` designs an `M`th order linear-phase FIR digital filter with the frequency response specified by arrays `f` and `A`, and returns the filter coefficients in length $M + 1$ array `h`. The array `f` is a normalized frequency array where `f=1` corresponds to π radians (or half the sampling frequency). The first and last elements of `f` must equal 0 and 1, respectively. The array `A` contains frequency response magnitude values specified at locations in `f`. For example, the lowpass filter in Example 10.5 can be designed using the command

```
>> h = fir2(62,[0,0.25,0.35,1],[1,1,0,0]);
```

which uses the Hamming window function. Similarly, the command

```
>> h = fir2(40,[0:20]/20,Ad(1:21),rectwin(41));
```

designs the DAC equalizer of Example 10.6 using a basic approach. MATLAB's SP toolbox manual should be consulted for other invocations of `fir2`.

Filters designed using frequency sampling methods are not optimum in the sense of any well-defined criterion. However, in applications where minimization of the mean squared error is a meaningful criterion, it is possible to design FIR filters by minimizing

$$E = \frac{1}{L}\sum_{k=0}^{L-1}\left|H\left(e^{j2\pi k/L}\right) - H_d\left(e^{j2\pi k/L}\right)\right|^2 = \sum_{k=0}^{L-1}\left|h[n] - h_d[n]\right|^2 \qquad (10.98)$$

in the time domain. Several design techniques using this approach, which have been introduced by Parks and Burrus (1987) and Selesnick et al. (1996), are implemented by MATLAB functions `firls`, `fircls`, and `fircls1`.

10.5   Chebyshev polynomials and minimax approximation

Design of FIR filters by minimizing the mean square error does not preclude the possibility of large errors at individual frequencies. However, in most applications it is important that the error be small at all frequencies within the range of interest. This can be achieved by minimizing the maximum absolute error (see Section 10.1.3). The solution to this mini-max optimization problem has an interesting and fundamental connection with Chebyshev polynomials.

10.5.1   Definition and properties

The basis for the development of Chebyshev polynomials is summarized in Figure 10.24. We note that for each $x \in [-1, 1]$, there is a complex number w on the unit circle, say $w = e^{j\theta}$, such that

$$x = \mathcal{R}e(w) = \frac{1}{2}\left(w + w^{-1}\right) = \cos\theta \in [-1, 1]. \qquad (10.99)$$

The mth-order *Chebyshev polynomial*, denoted by $T_m(x)$, is defined by

$$T_m(x) \triangleq \mathcal{R}e\left(w^m\right) = \frac{1}{2}\left(w^m + w^{-m}\right) = \cos(\theta m) = \cos\left[m \cos^{-1}(x)\right]. \qquad (10.100)$$

We note that the variables x and w are uniquely related for $0 \le \theta \le \pi$. To see that $T_m(x)$ is indeed a polynomial, we recall the trigonometric identity

$$\cos[(m+1)\theta] = 2\cos(\theta)\cos(m\theta) - \cos[(m-1)\theta], \quad m \ge 1 \qquad (10.101)$$
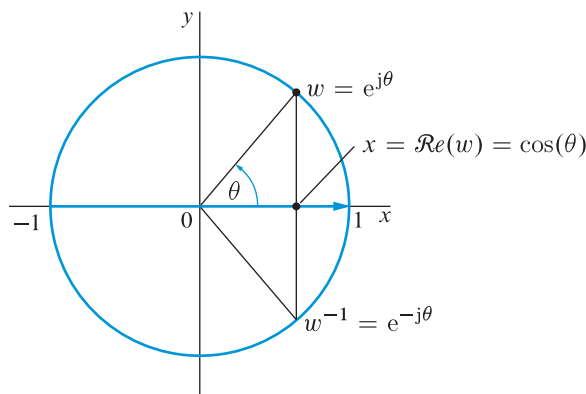
Figure 10.24 Variables used for the definition of Chebyshev polynomials.

Table 10.4 Low order Chebyshev polynomials.

Order	Polynomial $T_m(x) = \cos[m\cos^{-1}(x)]$
0	$T_0(x) = 1$
1	$T_1(x) = x$
2	$T_2(x) = 2x^2 - 1$
3	$T_3(x) = 4x^3 - 3x$
4	$T_4(x) = 8x^4 - 8x^2 + 1$
5	$T_5(x) = 16x^5 - 20x^3 + 5x$
6	$T_6(x) = 32x^6 - 48x^4 + 18x^2 - 1$

which, using (10.100) and (10.99), leads to the following recursive formula

$$T_{m+1}(x) = 2xT_m(x) - T_{m-1}(x). \quad m \geq 1 \tag{10.102}$$

With $T_0(x) = 1$ and $T_1(x) = x$, obtained from (10.100), we can generate any Chebyshev polynomial using (10.102). The first few polynomials are shown in Table 10.4. In general, we can show that $T_m(x)$ is an mth degree polynomial with leading coefficient 2^{m-1} for $m \geq 1$ and 1 for $m = 0$. Note that from (10.100) it follows that $|T_m(x)| \leq 1$ for $x \in [-1, 1]$, even though its leading coefficient is as large as 2^{m-1}. All Chebyshev polynomials have integer coefficients and satisfy the property

$$T_m(-x) = (-1)^m T_m(x). \tag{10.103}$$

To see how Chebyshev polynomials look, consider Figure 10.25, which shows how a graph of $T_m(x)$ in the variable x compares with a graph in the variable $\theta = \cos^{-1}(x)$. We note that the shape of $T_5(x)$ on $[-1, 1]$ is very similar to that of $V_5(\theta) \triangleq \cos(5\theta)$ on $[0, \pi]$, and in particular both oscillate between six extrema of equal magnitudes (unity) and alternating signs. However, there are three key differences: (a) the polynomial $T_5(x)$ corresponds to $V_5(\theta)$ reversed (that is, starting with a value of -1 and finishing with a value of $+1$),
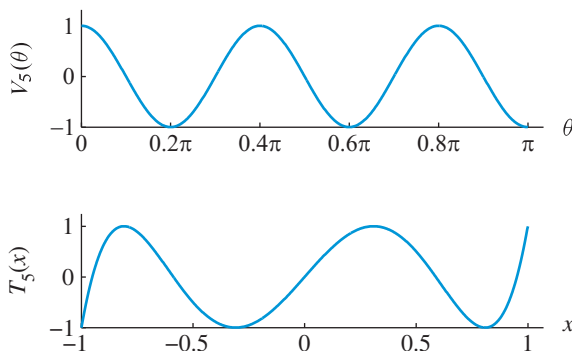
Figure 10.25 Plots of $V_m(\theta) = \cos(m\theta)$ and $T_m(x)$ for $m = 5$.

(b) the extrema of $T_5(x)$ at the end points $x = \pm 1$ do not correspond to zero gradients, as they do for $V_5(\theta)$, and (c) the zeros and extrema of $T_5(x)$ are clustered towards the end points $x = \pm 1$, whereas the zeros and extrema of $V_5(\theta)$ are equally spaced. Hamming (1973), who provides a lucid introduction to Chebyshev polynomials, refers to $T_m(x)$ as a "cosine in disguise."

Since $T_m(x) = \cos(\theta m) = 0$ for $\theta = (2k - 1)\pi/(2m)$, the mth-order polynomial $T_m(x)$ has m zeros in $[-1, 1]$, which are given by

$$x_k = \cos\left(\frac{2k - 1}{m}\frac{\pi}{2}\right). \quad k = 1, 2, \ldots, m \tag{10.104}$$

Also, because $\cos(m\theta) = \pm 1$ for $\theta = k\pi/m$, the $(m + 1)$ extrema of $T_m(x)$ are

$$\xi_k = \cos(k\pi/m), \quad T_m(\xi_k) = (-1)^k. \quad k = 0, 1, \ldots, m \tag{10.105}$$

Note that these formulas do not give the zero points and extremum points ordered as $x_k \leq x_{k+1}$ and $\xi_k \leq \xi_{k+1}$. Since the alternate maxima and minima (extremes) are of the same magnitude, we say that Chebyshev polynomials have an *equal-ripple* or *equiripple* *property*.

Note that any frequency response function that has the form of a finite trigonometric series in $\cos(\omega m)$ can be expressed in terms of Chebyshev polynomials. For example, using the definition (10.100) and Table 10.4, we can easily show that

$$P(\omega) \triangleq 2 + \cos(\omega) + \cos(2\omega) + \cos(3\omega)$$
$$= [2T_0(x) + T_1(x) + T_2(x) + T_3(x)]|_{x=\cos(\omega)}$$
$$= 1 - 2x + 2x^2 + 4x^3|_{x=\cos(\omega)}. \tag{10.106}$$

In general, any trigonometric cosine series can be expressed as a polynomial in $\cos(\omega)$ as follows:

$$P(e^{j\omega}) = \sum_{k=0}^{R} \tilde{p}[k] \cos \omega k = \sum_{k=0}^{R} \tilde{p}[k]T_k(x) = \left.\sum_{k=0}^{R} p_k x^k\right|_{x=\cos \omega}. \tag{10.107}$$

This allows us to express the real and even amplitude response of FIR filters with linear phase in terms of Chebyshev polynomials. This relationship establishes the connection between Chebyshev polynomial approximation and filter design.

10.5.2 Minimax approximation optimality

As shown by the following theorem, see Hamming (1973), the equiripple property connects Chebyshev polynomials with the minimax criterion (minimization of the maximum deviation):

Chebyshev's theorem Of all polynomials of degree m with coefficient of x^m equal to 1, the Chebyshev polynomial $T_m(x)$ multiplied by $2^{-(m-1)}$ has the least maximum amplitude on the interval $[-1, 1]$.

The proof of this elegant theorem is quite simple. We first note that the polynomial $Q_m(x) \triangleq T_m(x)/2^{m-1}$ has a leading coefficient 1; with this normalization all polynomials are scaled alike. Assume now that there exists an mth-degree polynomial $P_m(x)$ with leading coefficient 1 which has smaller extreme values than $Q_m(x)$, that is,

$$\max_{-1 \le x \le 1} |P_m(x)| < \max_{-1 \le x \le 1} |Q_m(x)|. \tag{10.108}$$

If $\xi_0, \xi_1, \ldots, \xi_m$ denote the extremal points of $Q_m(x)$, defined by (10.105), we have

$$P_m(\xi_k) \le Q_m(\xi_k), \quad \text{if } Q_k(\xi_k) > 0 \tag{10.109a}$$

$$P_m(\xi_k) \ge Q_m(\xi_k). \quad \text{if } Q_k(\xi_k) < 0 \tag{10.109b}$$

The difference polynomial $D(x) \triangleq Q_m(x) - P_m(x)$ has degree $(m-1)$ because the term x^m is canceled. From (10.105) we conclude that $D(x)$ changes sign in each of the m intervals (ξ_k, ξ_{k+1}), $k = 0, 1, \ldots, m-1$; therefore, it has m real zeros. However, this is impossible, because $D(x)$ can have at most $(m-1)$ zeros. This contradiction proves that the normalized Chebyshev polynomial $Q_m(x)$ has the minimax property.

This theorem shows the significance of Chebyshev polynomials to minimax polynomial approximation problems: *If we can express the error as a single Chebyshev polynomial, then we have the best minimax polynomial approximation.* The values $x = \xi_k$ are called the *extremal nodes* or points of equi-oscillation of the minimax approximation.

The theory of minimax approximation has been studied thoroughly and there exist algorithms for finding the best solution. The methods are based on the following characterization of the best minimax approximation, see Cheney (1966) or Powell (1981).

Alternation theorem Suppose that $f(x)$ is a continuous function. Then $P_m(x)$ is the best minimax approximating polynomial to $f(x)$ if and only if the error $e(x) = f(x) - P_m(x)$ has an $(m+2)$-point equiripple property.

As an illustration of the alternation theorem, suppose that the function $f(x) = x^2$ is approximated by the polynomial $P_1(x) = x - 0.125$ in the interval $0 \le x \le 1$. The error $e(x) = f(x) - P_1(x) = x^2 - x + 0.125$ has $m + 2 = 3$ extrema with the same magnitude and alternating signs: $x(0) = 0.125$, $x(0.5) = -0.125$, and $x(1) = 0.125$ as depicted in Figure 10.26. Hence, the polynomial $P_1(x)$ is the unique minimax approximation to the function $f(x) = x^2$.

The alternation theorem provides a unique characterization of minimax approximations, but does not suggest a procedure to find one. The most effective method for finding polynomial minimax approximations is the *Remez exchange algorithm*, see Cheney (1966), Powell (1981). Starting with a "guessed" set of $(m+2)$ nodes, we find a polynomial $P_m(x)$ such that

$$e(\xi_k) = f(\xi_k) - P_m(\xi_k) = (-1)^k \delta, \quad k = 0, 1, \ldots, m+1 \tag{10.110}$$
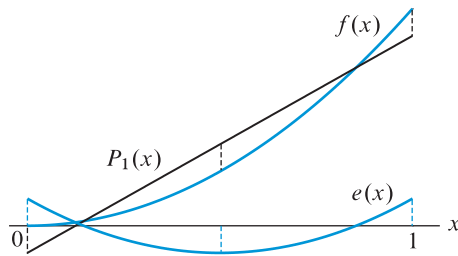
Figure 10.26 Minimax approximation of $f(x) = x^2$.

where δ is fixed but unknown. This involves solving a system of $(m + 2)$ linear equations for the $(m + 1)$ coefficients of $P_m(x)$ and δ. The $(m + 2)$ points at which $e(x)$ involves local extreme values alternating in sign and $|e(x)| > |\delta|$, are then determined. If these $(m + 2)$ values differ in magnitude more than some prescribed tolerance, the process is repeated by exchanging the old extrema points with the new. The algorithm stops when a set of $(m + 2)$ "nearly equiripple" points is obtained. This process is illustrated in Tutorial Problem 14.

10.6   Equiripple optimum Chebyshev FIR filter design

In this section we show how the linear-phase FIR filters design problem can be formulated as a Chebyshev approximation problem. This formulation, combined with the Remez exchange algorithm, leads to a powerful, flexible, and fast filter design method, which is known as the *Parks–McClellan algorithm*; the discovery of this algorithm is discussed by McClellan and Parks (2005). This method is the leading technique for the design of optimum Chebyshev FIR filters.

10.6.1   Problem formulation

Formulation of the optimum linear phase FIR filter design problem as a Chebyshev approximation problem starts with specification of a desired amplitude response $A_d(e^{j\omega})$ and a nonnegative weighting function $W(\omega)$. If the actual filter has amplitude response $A(e^{j\omega})$, we define the approximation error by

$$E(\omega) \triangleq W(\omega)\left[A_d(e^{j\omega}) - A(e^{j\omega})\right]. \tag{10.111}$$

The function $W(\omega)$ enables us to control the relative size of the error in different frequency bands. The design objective is to find the coefficients of a type I–IV FIR filter that minimize the weighted Chebyshev error, defined by

$$\|E(\omega)\|_\infty = \max_{\omega \in \mathcal{B}} |E(\omega)|, \tag{10.112}$$

where $\mathcal{B}$ is a union of disjoint closed subsets (corresponding to frequency bands) of $0 \leq \omega \leq \pi$. The approximation function $A(e^{j\omega})$ is not constrained in the transition bands; thus, it can take any shape required to best approximate $A_d(e^{j\omega})$ in $\mathcal{B}$. Since we focus on FIR filters with linear phase, the amplitude response function $A(e^{j\omega})$ can be written as (see Table 10.2)

$$A(e^{j\omega}) = Q(e^{j\omega})P(e^{j\omega}), \tag{10.113}$$

where

$$Q(e^{j\omega}) = \begin{cases} 1, & \text{Type I} \\ \cos(\omega/2), & \text{Type II} \\ \sin(\omega), & \text{Type III} \\ \sin(\omega/2), & \text{Type IV} \end{cases}, \quad P(e^{j\omega}) = \sum_{k=0}^{R} p[k] \cos(\omega k), \tag{10.114}$$

and $R = M/2$, if M is even and $R = (M-1)/2$, if M is odd. Since $Q(e^{j\omega})$ does not depend on the filter coefficients, the error (10.111) can be rewritten as

$$E(\omega) = W(\omega)Q(e^{j\omega}) \left[\frac{A_d(e^{j\omega})}{Q(e^{j\omega})} - P(e^{j\omega}) \right], \tag{10.115a}$$

$$\triangleq \bar{W}(\omega) \left[\bar{A}_d(e^{j\omega}) - P(e^{j\omega}) \right], \tag{10.115b}$$

where

$$\bar{W}(\omega) \triangleq W(\omega)Q(e^{j\omega}), \ \bar{A}_d(e^{j\omega}) \triangleq \frac{A_d(e^{j\omega})}{Q(e^{j\omega})}. \tag{10.116}$$

We note that $E(\omega)$ is not defined at the end points where $Q(e^{j\omega}) = 0$. This formulation, introduced by McClellan and Parks (1973), provides a unified framework for the design of FIR filters with linear phase.

The Chebyshev approximation problem may now be stated:

Given the filter order M, determine the coefficients of $P(e^{j\omega})$ that minimize the maximum absolute value of $E(\omega)$ over the frequency bands of interest, that is, choose $P(e^{j\omega})$ so that

$$\|E(\omega)\|_\infty = \max_{\omega \in \mathcal{B}} |\bar{W}(\omega)[\bar{A}_d(e^{j\omega}) - P(e^{j\omega})]| \tag{10.117}$$

is minimum.

Since $P(e^{j\omega})$ can be expressed as a linear combination of cosine functions, which in turn is equivalent to a polynomial in $x = \cos\omega$ according to (10.107), the linear-phase FIR filter design problem is reduced to a Chebyshev polynomial approximation problem. However, as shown by Powell (1981), the alternation theorem can be generalized for approximations that satisfy the "Haar conditions," of which polynomials are a special case. This allows us

to look for best approximations in trigonometric polynomials, which are the ones used in FIR filter design problems.

10.6.2   Specifying the optimum Chebyshev approximation

The solution to the Chebyshev or minimax approximation problem (10.117) is characterized by the alternation theorem, which is most often stated for approximating a continuous function over an interval (see Section 10.5.2). However, for the filter design problem, we need a more general version which uses trigonometric polynomials to approximate continuous functions on a union of closed disjoint intervals and discrete points; see Cheney (1966). This version of the alternation theorem is stated as follows:

Alternation theorem for FIR filters   If $P(e^{j\omega})$ is given by (10.114), then a necessary and sufficient condition that $P(e^{j\omega})$ be the unique solution of (10.117) is that the weighted error function $E(\omega)$ exhibit at least $R + 2$ alternations in $\mathcal{B}$. That is, there must exist $R + 2$ extremal frequencies $\omega_1 < \omega_2 < \cdots < \omega_{R+2}$ such that for every $k = 1, 2, \ldots, R + 2$

$$E(\omega_k) = -E(\omega_{k+1}), \quad |E(\omega_k)| = \max_{\omega \in \mathcal{B}} E(\omega) \triangleq \delta. \tag{10.118}$$

The alternation theorem implies that the best Chebyshev approximation must have an equiripple error function. It also states that, for a given $\mathcal{B}$, filter order M, and weighting function $W(\omega)$, there is a unique best approximation. Careful examination of the alternation theorem shows that, as part of the design process, we have control of the following quantities:

1. Filter order M: this is related to the order R of $P(e^{j\omega})$ by $R = M/2$ (even M) and $R = (M - 1)/2$ (odd M).
2. Edge frequencies of passbands and stopbands: each band b is determined by a closed interval $[\omega_k^b, \omega_{k+1}^b]$. The compact set $\mathcal{B}$, which is part of the alternation theorem, is the union of these intervals.
3. Desired amplitude response $A_d(e^{j\omega})$: this function should be defined and be continuous only within the compact set $\mathcal{B}$. Transition bands are left unspecified to provide additional approximation power within the interval $\mathcal{B}$.
4. Weighting function $W(\omega)$: this function, which is defined only within $\mathcal{B}$, is used to include the approximation error parameters in the design process.

According to the alternation theorem, these design requirements *uniquely* specify a best Chebyshev filter, having minimax error δ given by (10.118). If the error δ does not meet the requirements, we should increase M until the requirements are met.

To understand the key aspects and the application of alternation theorem and its limitations, we consider the design of an Mth-order type I lowpass filter. The desired amplitude response function is defined by

$$A_d(e^{j\omega}) = \begin{cases} 1, & 0 \le \omega \le \omega_p \\ 0. & \omega_s \le \omega \le \pi \end{cases} \tag{10.119}$$
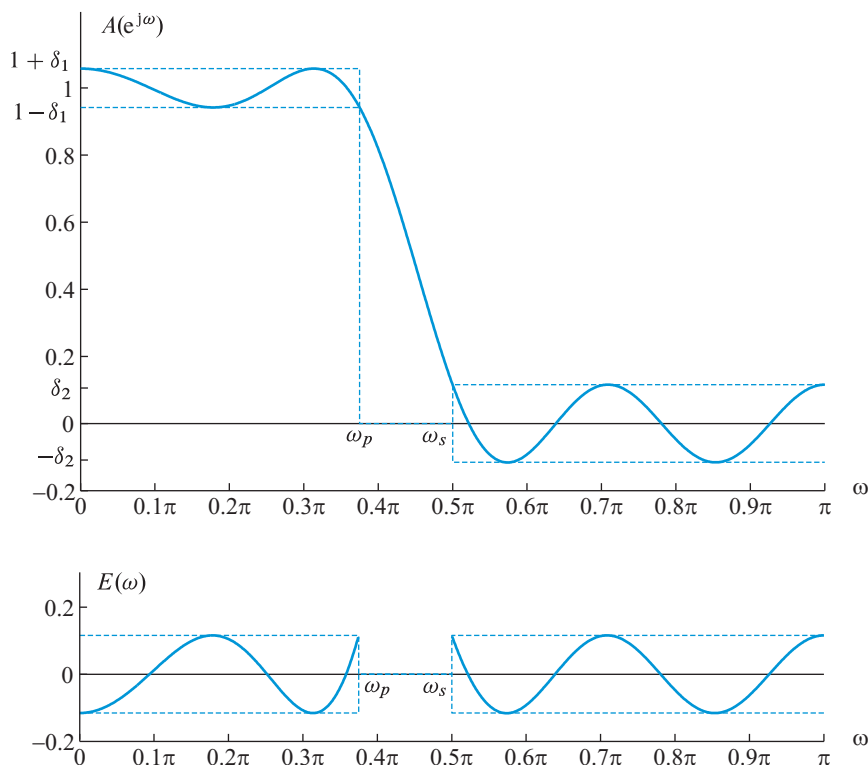
Figure 10.27 Specifications and typical amplitude response for a type-I FIR filter that is optimal according to the Chebyshev approximation criterion.

We note that the interval $\mathcal{B}$ is the union of the passband $[0, \omega_p]$ and the stopband $[\omega_s, \pi]$. To allow different weighting of the approximation error in the passband and stopband, we define the weighting function as

$$W(\omega) = \begin{cases} 1, & 0 \le \omega \le \omega_p \\ K, & \omega_s \le \omega \le \pi \end{cases} \tag{10.120}$$

where $K \triangleq \delta_p / \delta_s$. The amplitude response of a type I FIR filter is given by

$$P(e^{j\omega}) = A(e^{j\omega}) = \sum_{k=0}^{R} a[k] \cos(\omega k) = \sum_{k=0}^{R} a_k \{\cos(\omega)\}^k, \tag{10.121}$$

where $R = M/2$, $h[R] = a[0]$, and $h[k] = a[R-k]/2$, $k = 1, 2, \ldots, R$. Since $Q(e^{j\omega}) = 1$, this is the simplest case of the Chebyshev approximation problem.

Figure 10.27 shows the amplitude response of a type-I FIR filter which is optimum according to the alternation theorem for $R = M/2 = 7$. The design algorithm uses the weighted error function (10.120) with $K = \delta_p / \delta_s = 1/2$. We note that, as required by the alternation theorem, $E(\omega)$ has $R + 2 = 9$ alternations in $\mathcal{B}$ at the extremal frequencies

$\omega_1, \omega_2 \ldots, \omega_9$. Thus, as expected, the *weighted* error $E(\omega)$ of the optimum approximation is equiripple. The algorithm minimizes the maximum weighted error, which here is equal to δ_p.

The alternation theorem does not impose any limit on the number of disjoint intervals in $\mathcal{B}$; thus, the minimum number of alternations is $(R + 2)$, irrespective of the number of bands. However, the theorem does not say anything about the maximum number of alternations. Taking the derivative of (10.121) and setting it to zero, we have

$$\frac{\mathrm{d}}{\mathrm{d}\omega} A(\mathrm{e}^{\mathrm{j}\omega}) = -\sin(\omega) \sum_{k=1}^{R} k a_k \{\cos(\omega)\}^{k-1} = 0. \qquad (10.122)$$

The factor $\sin(\omega)$ implies solutions at $\omega = 0$ or $\omega = \pi$, while the remaining polynomial has order $(R-1)$, and thus has at most $(R-1)$ distinct roots. Hence, there are $(R-1)+2 = R+1$ possible extrema of zero slope. Furthermore, $E(\omega)$ has two extrema at the band edges ω_p and ω_s. Therefore, for a lowpass filter, $E(\omega)$ may have, at most, $(R + 3)$ extremal frequencies. The case of $R + 3$ alternations is known as the *extraripple* case. Thus, in general, type-I lowpass FIR filters have the following properties:

- The maximum possible number of alternations of the error is $(R + 3)$.
- Alternations always occur at the band edges ω_p and ω_s.
- The filter will be equiripple except possibly at $\omega = 0$ or $\omega = \pi$.

The alternation theorem has more implications for the properties of the designed filters; several such properties for type I–IV FIR filters are discussed in Rabiner and Gold (1975) and Rabiner et al. (1975).

10.6.3 Finding the optimum Chebyshev approximation

The alternation theorem characterizes the optimum minimax or Chebyshev solution so that one can be recognized, but it does not state explicitly how to obtain the optimum filter coefficients. However, the conditions characterizing the optimum filter can be used to develop an efficient algorithm for obtaining filter coefficients. Parks and McClellan (1972) solved this problem by using the Remez exchange algorithm. The result is the well-known *Parks–McClellan algorithm*, which is the dominant method for design of FIR filters with linear phase. We next present the basic principles of the Parks–McClellan algorithm. Because the actual implementation is quite complicated, we suggest use of the MATLAB function `firpm` or other similar professional software. A detailed description of the algorithm and its software implementation are provided in McClellan *et al.* (1973).

The Remez exchange algorithm, see Cheney (1966), exploits the characterization of the optimum error by the alternation theorem. If the extremal frequencies $\omega_i, i = 1, 2, \ldots, R+2$ for the optimum filter were known, we could find the coefficients $a[k], k = 0, 1, \ldots, R$ and the corresponding error δ by solving the set of linear equations

$$E(\omega_i) = W(\omega_i)\left[A_{\mathrm{d}}\left(\mathrm{e}^{\mathrm{j}\omega_i}\right) - A(\mathrm{e}^{\mathrm{j}\omega_i})\right] = (-1)^{i+1}\delta, \ 1 \le i \le R + 2, \qquad (10.123)$$
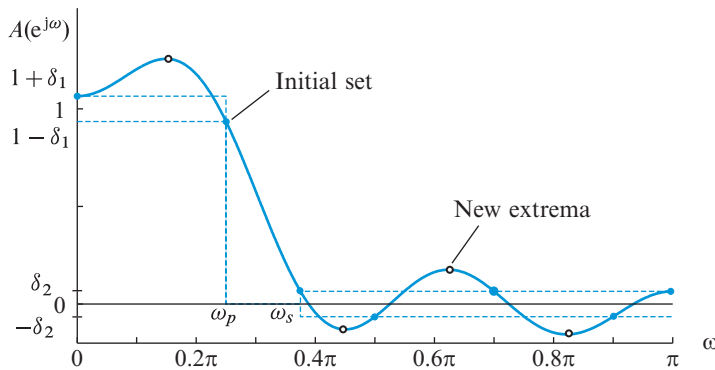
Figure 10.28 Initialization step of Remez exchange algorithm.

or equivalently

$$
\begin{bmatrix}
1 & \cos(\omega_1) & \dots & \cos(R\omega_1) & \frac{1}{W(\omega_1)} \\
1 & \cos(\omega_2) & \dots & \cos(R\omega_2) & \frac{-1}{W(\omega_2)} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
1 & \cos(\omega_{R+1}) & \dots & \cos(R\omega_{R+1}) & \frac{(-1)^{R+2}}{W(\omega_{R+1})} \\
1 & \cos(\omega_{R+2}) & \dots & \cos(R\omega_{R+2}) & \frac{(-1)^{R+3}}{W(\omega_{R+2})}
\end{bmatrix}
\begin{bmatrix}
a[0] \\
a[1] \\
\vdots \\
a[R] \\
\delta
\end{bmatrix}
=
\begin{bmatrix}
A_d(e^{j\omega_1}) \\
A_d(e^{j\omega_2}) \\
\vdots \\
A_d(e^{j\omega_{R+1}}) \\
A_d(e^{j\omega_{R+2}})
\end{bmatrix}.
\quad (10.124)
$$

Therefore, finding the optimum coefficients $a[k]$, $0 \le k \le R$ has been reduced to finding the optimum extremal frequencies ω_i, $1 \le i \le R + 2$. The Remez exchange algorithm solves the Chebyshev approximation problem by searching for the extremal frequencies of the best approximation.

The procedure begins by guessing a set of alternation frequencies ω_i, $i = 1, 2, \dots$, $(R + 2)$. This set, in which frequencies are usually uniformly choosen, should include the frequencies $\omega = 0, \omega_p, \omega_s$, and π. Note that, if $\omega_m = \omega_p$, then $\omega_{m+1} = \omega_s$. Given these frequencies, we could solve the linear system (10.124) to obtain $\{a[k], 0 \le k \le R\}$ and δ. The resulting filter is optimal but only for the guessed frequencies; the objective is optimality over the entire $\mathcal{B}$. Figure 10.28 illustrates the initialization step for a type-I lowpass filter with $M = 10$, $\omega_p = 0.25\pi$, $\omega_s = 0.375\pi$, and $K = 1$. Since $R = M/2 = 5$, we choose $R + 2 = 7$, initial frequency set $\{0, \omega_p, \omega_s, 0.5\pi, 0.7\pi, 0.9\pi, \pi\}$, shown by blue filled circles. Then, we solve the linear system (10.124) to obtain $a[0], a[1] \dots, a[5]$, and δ (see Tutorial Problem 15). The amplitude response $A(e^{j\omega})$ is evaluated using (10.121). Apparently, the value $\delta = 0.065$ is too small for the chosen extremal frequencies. There are $(R - 1) = 4$ local extrema of $E(\omega)$ in the open intervals $0 < \omega < \omega_p$ and $\omega_s < \omega < \pi$, because $A(e^{j\omega})$ has R roots in this interval. These points, which are denoted by open black circles, should be part of the new set of extremal frequencies. The frequencies ω_p and ω_s should always be part of the extremal set. The remaining extremal frequency can be at either $\omega = 0$ or $\omega = \pi$; we usually pick the one with the larger error. Including the new extremal set frequencies where $|E(\omega)| > |\delta|$, while preserving the alternating condition, assures convergence of the Remez algorithm. The monotonic increase of $|\delta|$ is
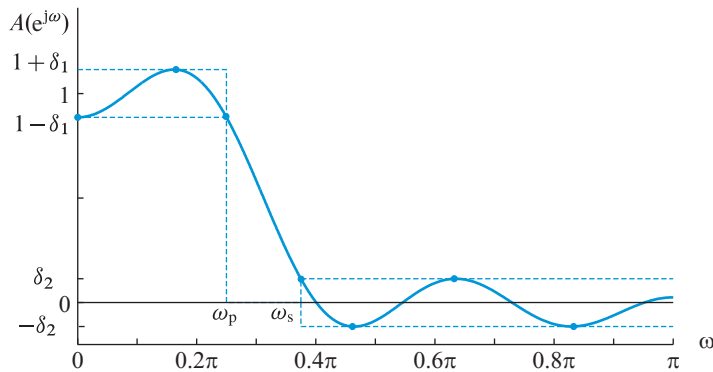
Figure 10.29 Optimum equiripple Chebyshev filter obtained when the Remez exchange algorithm, discussed in Figure 10.28, converges.

an indication that the algorithm converges to the optimum solution (see Powell (1981), Theorem 9.3). The process is repeated until δ does not change from its previous value by a significant amount. This value of δ is the minimum maximum approximation error for the optimum equiripple filter obtained when the Remez algorithm converges (see Figure 10.29).

Practical considerations At each iteration, the Remez algorithm requires (a) the value of error δ, (b) the values of $E(\omega)$ on a dense grid of frequencies, and (c) a search algorithm to find the local extrema of $E(\omega)$. Computation of δ and $E(\omega)$ using (10.119), (10.121), and (10.124) requires solution of a linear system, which is a difficult and slow process for large matrices. Since the coefficients $a[k]$ are not needed until the last iteration, we can avoid the solution of (10.124) using a more efficient approach proposed by Parks and McClellan (1972). The first step is to calculate δ analytically using the formulas

$$\delta = \frac{\sum_{k=1}^{R+2} b_k A_d(e^{j\omega_k})}{\sum_{k=1}^{R+2} \frac{b_k(-1)^{k+1}}{W(\omega_k)}}, \qquad b_k = \prod_{\substack{i=1 \\ i \neq k}}^{R+2} \frac{1}{x_k - x_i}, \tag{10.125}$$

where $x_i = \cos(\omega_i)$. Having determined δ, we can use (10.123) to determine the values

$$A(e^{j\omega_k}) = A_d(e^{j\omega_k}) - \frac{(-1)^{k+1}\delta}{W(\omega_k)}. \quad k = 1, 2, \ldots, R+1 \tag{10.126}$$

Since $A(e^{j\omega})$ is an Rth-degree trigonometric polynomial, we can fit this polynomial to the $R + 1$ points $(\omega_k, A(e^{j\omega_k}))$, and then use the fitted polynomial to interpolate at the required grid of frequencies. Parks and McClellan used the *barycentric* form of Lagrange interpolation formula, which is defined by

$$A(e^{j\omega}) = \frac{\sum_{k=1}^{R+1} A(e^{j\omega_k}) \frac{d_k}{(x - x_k)}}{\sum_{k=1}^{R+1} \frac{d_k}{(x - x_k)}},$$ (10.127)

and

$$d_k = \prod_{\substack{i=0 \\ i \neq k}}^{R+1} \frac{1}{(x_k - x_i)} = b_k(x_k - x_{R+2}),$$ (10.128)

where $x = \cos(\omega)$ and $x_k = \cos(\omega_k)$. The barycentric formulation is a stable and fully effective numerical algorithm for the interpolation of high degree polynomials and has contributed to the good numerical properties of the Parks–McClellan algorithm; see Berrut and Trefethen (2004) and Pachon and Trefethen (2009). Since $A(e^{j\omega})$ is available at any desired frequency, we can evaluate $E(\omega)$ at any frequency because $A_d(e^{j\omega})$ and $W(\omega)$ are defined analytically.

The search for the local extrema of $E(\omega)$ is an algorithmically complicated process. Finding the correct extremal frequencies is critical to the success of the Remez algorithm. However, according to Rabiner *et al.* (1974a), the outcome of this task becomes increasingly less reliable as the number of bands is increased beyond two. Thus, sometimes the algorithm may converge extremely slowly or it may generate filters with very large undesirable fluctuations in one or more transition bands. Various practical aspects and improvements of the Parks–McClellan algorithm are discussed by Shpak and Antoniou (1990) and Antoniou (2006).

Figure 10.30 shows the flow-chart for the Parks–McClellan algorithm described by McClellan *et al.* (1973). The best approximation returned by the algorithm is specified by a set of extremal frequencies ω_k and the corresponding values $A(e^{j\omega_k})$. The coefficients $a[k]$ can be determined by solving (10.124) or an interpolation problem (see Problem 56). The impulse response can be computed using the results in Tables 10.1 and 10.2.

To use this algorithm effectively, we should be able to determine the filter order M from the design parameters ω_p, ω_s, δ_p, and δ_s. Although an exact relationship is not known, we use the following empirical formula introduced by Kaiser (1974):

$$M = \frac{-20 \log_{10} \sqrt{\delta_p \delta_s} - 13}{2.324 \Delta\omega},$$ (10.129)

where $\Delta\omega = \omega_s - \omega_p$ is the transition bandwidth. This formula and the one in (10.85) can be used to compare optimum equiripple Parks–McClellan filters to filters designed using the Kaiser window. If $\delta_p = \delta_s$, for the same values of M and $\Delta\omega$, the attenuation of the equiripple filter is approximately 5 dB smaller compared to a Kaiser filter. An additional advantage of equiripple filters is that δ_p does *not* have to be equal to δ_s, as with the window design methods.
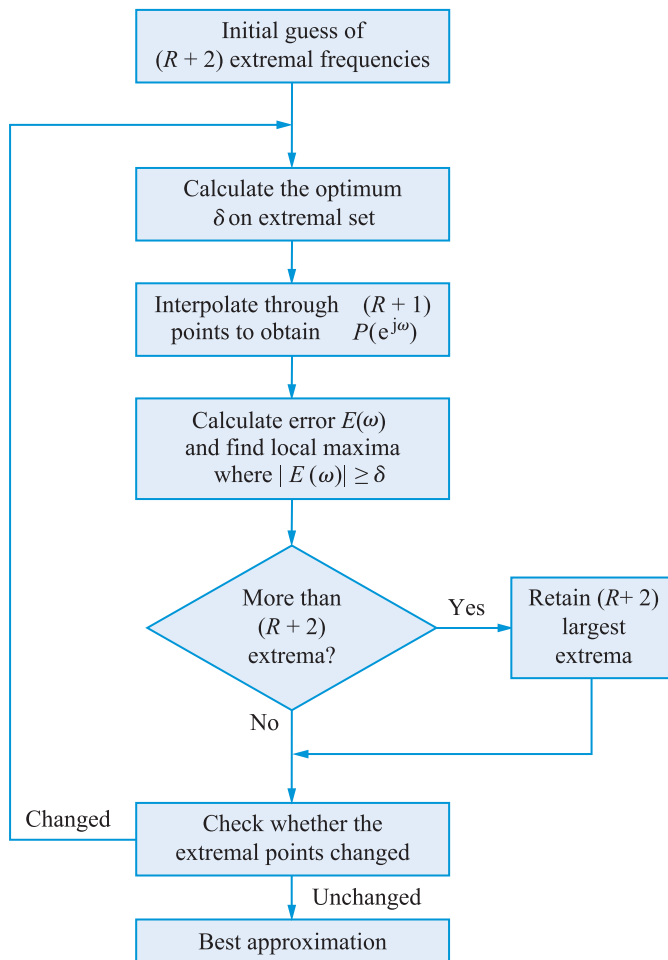
Figure 10.30 Flow-chart of Parks–McClellan algorithm.

10.6.4   Design examples using MATLAB

The optimum equiripple FIR filter design requires computer-aided techniques and MAT-LAB provides two useful functions. The function `firpmord` determines the approximate filter order M using (10.129) and the function `firpm` implements the flowchart of Figure 10.30 to obtain the impulse response of the optimum equiripple FIR filter for the computed M. The basic syntax of `firpmord` is

`[M,fo,ao,W]=firpmord(f,a,dev).`

The input parameters to the function are: an array `f` containing normalized band edges, an array `a` containing desired amplitudes in a band defined by `f`, and an array `dev` containing tolerances (not in dB) in each respective band. The array `f` should not contain

a 0 or 1. The `firpmord` function then computes filter order `M`, the normalized frequency band edges in `fo`, amplitude response in `ao`, and the band weights in array `W`. These parameters are then used by the `firpm` to determine the impulse response $h[n]$ of the equiripple filter. Its basic syntax is

```
[h,delta]=firpm(M,fo,ao,W),
```

where `delta` returns the maximum ripple height δ after convergence to the optimum solution for the given order `M`. This δ should then be compared with the tolerance in the first band since the weighing function normalizes the error function with respect to the first band tolerance in the `firpmord` function, for example, δ_p for a lowpass design. If δ is larger (smaller), then increase (decrease) M until δ is less than or equal to the given tolerance. We illustrate the use of these functions in the following design examples.

Example 10.7   Lowpass filter

Consider the lowpass linear-phase FIR filter specifications (10.81) given in Example 10.2:

$$\omega_p = 0.25\pi, \quad \omega_s = 0.35\pi, \quad A_p = 0.1 \text{ dB}, \quad A_s = 50 \text{ dB}.$$

We need passband and stopband tolerance values which were computed as $\delta_p = 0.0058$ and $\delta_s = 0.0032$ using (10.5) in Example 10.2. We first determine the approximate filter order M using `firpmord` and then the optimum filter along with the maximum weighted error using `firpm`:

```
>> % Given Specifications
>> wp = 0.25*pi; ws = 0.35*pi; Ap = 0.1; As = 50;
>> % Passband and Stopband Ripple Calulations
>> deltap = (10^(Ap/20)-1)/(10^(Ap/20)+1);
>> deltas = (1+deltap)/(10^(As/20));
>> % Estimated Filter order using FIRPMORD function
>> [M,fo,ao,W] = firpmord([wp,ws]/pi,[1,0],[deltap,deltas]);
>> M
M =
    48
>> % Filter Design using FIRPM function
>> [h,delta] = firpm(M,fo,ao,W); err, deltap
delta =
    0.0071
deltap =
    0.0058
```

It is obvious that the filter order $M = 48$ is not sufficient since δ is more than δ_p. Hence we increase M until the maximum error is less than δ_p. The optimum value was found to be $M = 50$ for which $\delta = 0.0055$. In the window design approaches, we obtained $M = 66$ for
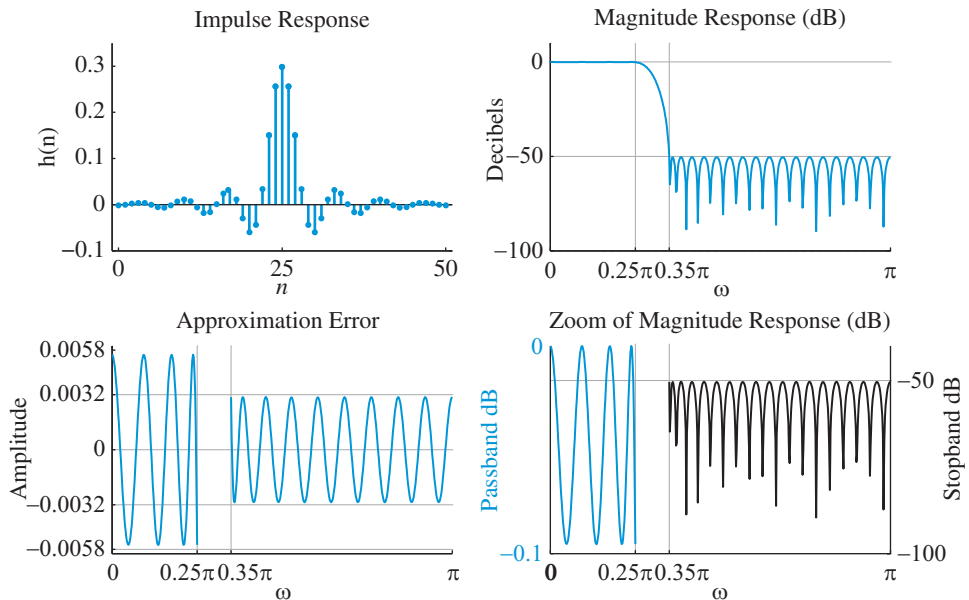
Figure 10.31 Impulse and frequency response plots of the lowpass filter designed in Example 10.7 using the Parks–McClellan algorithm to satisfy specifications: $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB.

the Hamming window and $M = 60$ for the Kaiser window. Clearly, the equiripple design approach provides the minimum order for the filter.

Figure 10.31 shows the impulse and frequency response plots of the designed filter. The magnitude (dB) of response plots show that the filter met its specifications in both bands and the response is equiripple. The approximation error plot shows that there are 27 extremal frequencies in the union set of passband and stopband, which is equal to $M/2 + 2$ as expected. ∎

Example 10.8   Bandpass filter

Consider the specifications (10.88) of the bandpass filter given in Example 10.4:

$$\left|H(e^{j\omega})\right| \le 0.01, \qquad\qquad |\omega| \le 0.2\pi$$

$$0.99 \le \left|H(e^{j\omega})\right| \le 1.01, \qquad 0.3\pi \le |\omega| \le 0.7\pi$$

$$\left|H(e^{j\omega})\right| \le 0.01. \qquad 0.78\pi \le |\omega| \le \pi$$

Since tolerances in absolute values are given, it is easy to assemble parameters for the `firpmord` function. The initial design is obtained using the MATLAB script:

```
% Given Specifications
>> ws1 = 0.2*pi; deltas1 = 0.01; % Lower stopband:
>> wp1 = 0.3*pi; wp2 = 0.7*pi; deltap = 0.01; % Passband
>> ws2 = 0.78*pi; deltas2 = 0.01; % Upper stopband
>> % Estimated Filter order using FIRPMORD function
>> f = [ws1,wp1,wp2,ws2]/pi; % Band-edge array
>> a = [0,1,0]; % Band-edge desired gain
>> dev = [deltas1,deltap,deltas2]; % Band tolerances
>> [M,fo,ao,W] = firpmord(f,a,dev); M
M =
    49
>> % Filter Design using FIRPM function
>> [h,delta] = firpm(M,fo,ao,W);
>> delta, deltas1
delta =
    0.0108
deltas1 =
    0.0100
```

The first band is a stopband, so we compare δ with δ_{s_1} and since $\delta > \delta_{s_1}$ we increase $M =$ 49 to 50. The resulting $\delta = 0.0093$ which satisfies the given requirements. This design again results in a smaller order than the one obtained using a Kaiser window ($M = 56$) in Example 10.4. Figure 10.32 shows the impulse and frequency response plots of
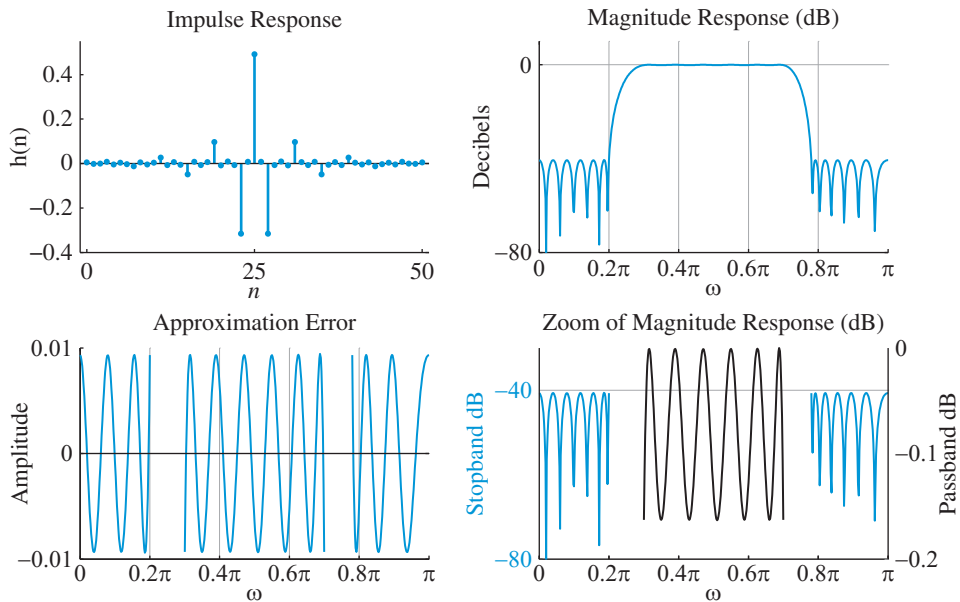
Figure 10.32 Impulse and frequency response plots of the bandpass filter designed in Example 10.7 using the Parks–McClellan algorithm to satisfy specifications: $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB.

the designed filter. Filter specifications are met in all bands as shown by the magnitude (dB) response plots and the response is equiripple. The approximation error plot shows that there are again $M/2 + 2 = 27$ extremal frequencies in the union set of passband and stopbands.   ■

The following example illustrates the use of the `pmfir` function in designing an arbitrary shaped frequency response. Additional equiripple FIR filter designs are explored in Tutorial Problems 16 and 17.

Example 10.9   DAC compensation

In Example 10.6, we designed a fullband DAC equalizer using a frequency sampling approach. Its frequency response is given by (10.97) and is repeated below

$$H_d(e^{j\omega}) = \frac{\omega/2}{\sin(\omega/2)}. \qquad -\pi \le \omega \le \pi$$

We consider a wideband DAC equalizer with bandwidth of $\omega_p = 0.55\pi$ and stopband starting at $\omega_s = 0.65\pi$. Let $\delta_p = \delta_s = 0.01$. First, note that the `firpmord` function is not suitable in this design. Second, we have to specify the frequency response of (10.97) over the given passband in the `firpm` function. This is done using a finer grid of bands so that the band edge responses linearly approximate the given $(\omega/2)/(\sin(\omega/2))$ response. We choose a grid based on 0.05π frequencies. Thus there are now six mini-bands interspersed with five transition bands, each of width 0.05π. Using trial and error we obtained the required design using $M = 41$. The following MATLAB script provides design details:

```
>> % DAC Specifications
>> fdac = 0:0.05:0.55;          % DAC band frequencies
>> adac = 1./sinc(fdac/2);      % DAC band responses
>> deltap = 0.01;                % DAC band ripple
>> ws = 0.65*pi; deltas = 0.01; % stopband
>> % Filter Design using FIRPM function
>> fo = [fdac,ws/pi,1]; % Band-edge array
>> ao = [adac,0,0]; % Band-edge desired gain
>> W = [deltap*ones(1,6),deltas]/deltas; % Band weights
>> M = 41; [h,delta] = firpm(M,fo,ao,W); delta, deltap,
delta =
    0.0099
deltap =
    0.0100
```

Figure 10.33 shows the impulse and frequency response plots of the designed DAC equalizer. Filter specifications are met in all bands as shown by the magnitude (dB)
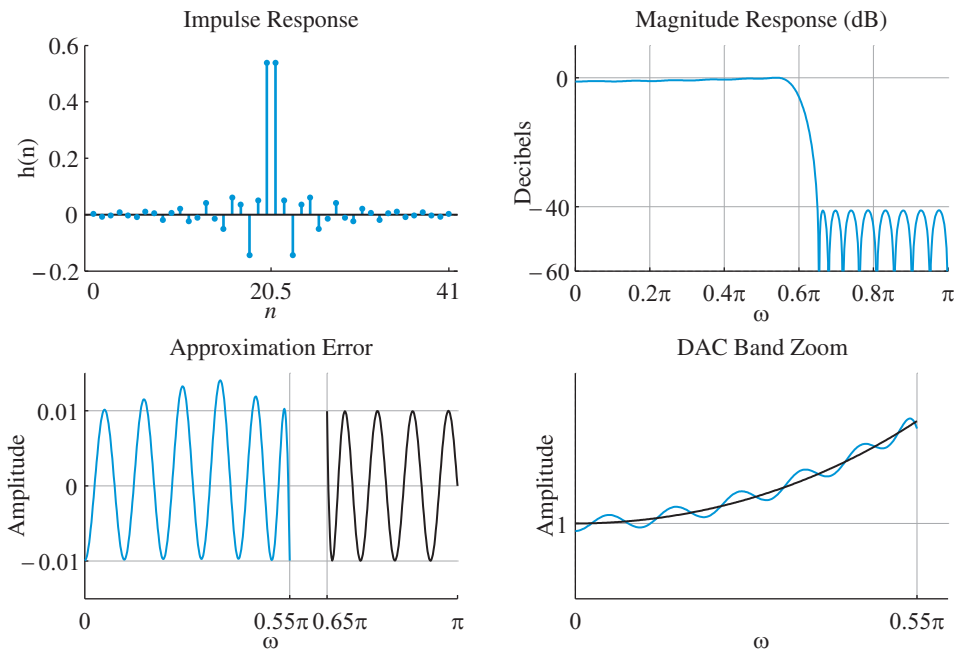
Figure 10.33 Impulse and frequency response plots of the wideband DAC equalizer designed in Example 10.9 using the Parks–McClellan algorithm to satisfy specifications: $\omega_p = 0.55\pi$, $\omega_s = 0.65\pi$, $\delta_p = 0.01$, and $\delta_s = 0.01$.

response plots and the response is equiripple. The peaks that are higher than δ are in the artificial transition bands created for the purpose of providing the arbitrary frequency response specifications to the `firpm` function. ■

FDATool for equiripple filter design The design panels displayed by `fdatool` for equiripple design are given in Figure 10.34, which shows parameters needed for the lowpass filter design in Example 10.7. To determine the initial filter order, we select the Minimum order radio button in the Filter Order panel, provide A_p and A_s in dB, and then design the filter. The resulting order is displayed as 48 in the Current Filter Specification panel. Since the achieved stopband attenuation shown in Figure 10.34(a) is more than 50dB, we select the Specify order radio button in the Filter Order panel, increase order to 50, provide the required weighting functions in the Magnitude Specifications panel, and again design the filter. This leads to the optimum filter response as shown in Figure 10.34(b) and as achieved in Example 10.7.

As a concluding comment before we close this section, we emphasize that good designs using the Parks–McClellan algorithm require experience. In particular, if the transition bands are narrow and/or the number of bands is more than three then ripples and band-edges should be properly chosen. Design plots for many such cases are given in Rabiner *et al.* (1974b).
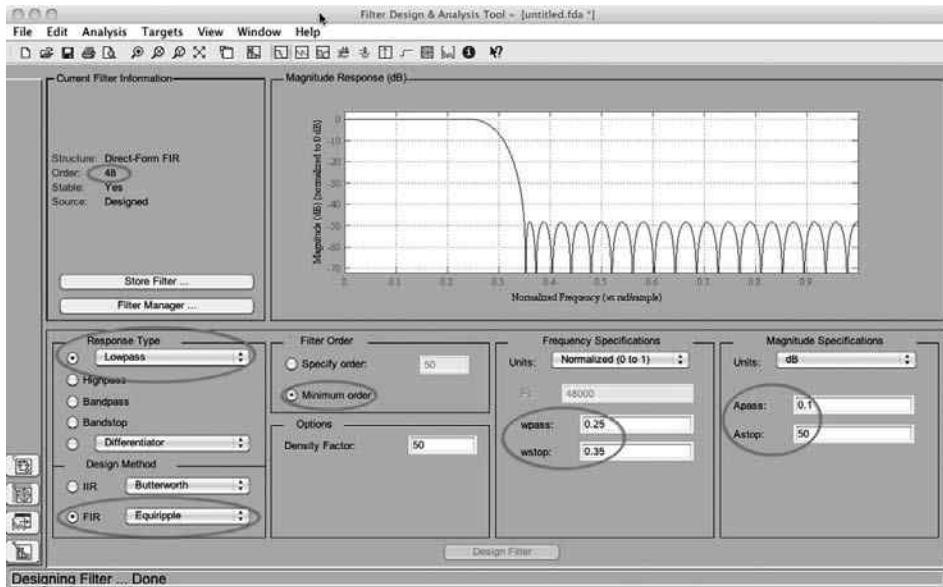
(a) Setup for obtaining initial filter order
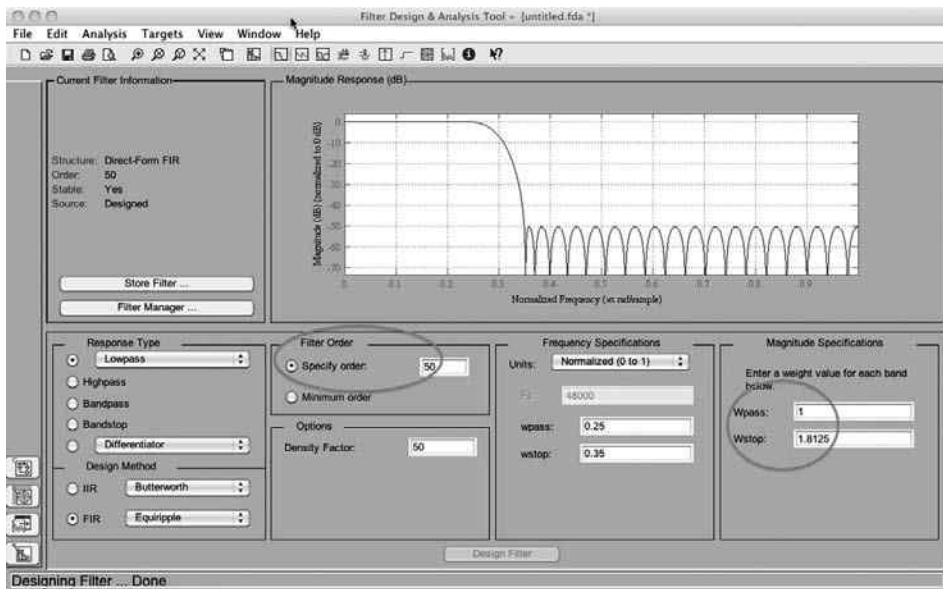
(b) Setup for obtaining optimum filter order

Figure 10.34 Graphical user interface of the FDATool for designing FIR filters via the Parks–McClellan algorithm.

10.7 Design of some special FIR filters

In this section we consider FIR filters that do not fall under the standard frequency selective category but nevertheless are important and useful in practice and can be designed using techniques discussed so far. These include differentiators, Hilbert transformers, and raised-cosine filters. Other special filters such as minimum-phase and Nyquist filters are discussed in Chapter 12.

10.7.1 Discrete-time differentiators

In continuous-time, the output of an ideal differentiator is the derivative of the input

$$y_c(t) = \frac{dx_c(t)}{dt}. \tag{10.130}$$

Taking the Fourier transform of both sides yields $Y_c(j\Omega) = j\Omega X_c(j\Omega)$. Thus, the frequency response function is

$$H_c(j\omega) = j\Omega. \tag{10.131}$$

The ideal continuous-time bandlimited differentiator is defined by

$$H_c(j\Omega) = \begin{cases} j\Omega, & |\Omega| < \pi/T \\ 0, & |\Omega| \geq \pi/T \end{cases} \tag{10.132}$$

because if $X_c(j\Omega) = 0$ for $|\Omega| > \pi/T$ the output is still the derivative of the input.

Clearly, the concept of derivative has no meaning in discrete-time. However, if we consider the system in Figure 6.18, an effective frequency response equal to (10.132) can be achieved by using a discrete-time system defined by (see Eq. (6.48))

$$H(e^{j\omega}) = H_c(j\omega/T) = \frac{j\omega}{T}. \quad |\omega| < \pi \tag{10.133}$$

Thus, we define the ideal causal *discrete-time differentiator* with linear phase by

$$H(e^{j\omega}) = (j\omega)e^{-j\omega\alpha}, \quad |\omega| < \pi \tag{10.134}$$

where we omit the factor $1/T$ for convenience. The impulse response is given by

$$h[n] = \frac{\cos[\pi(n-\alpha)]}{(n-\alpha)} - \frac{\sin[\pi(n-\alpha)]}{\pi(n-\alpha)^2}, \quad -\infty < n < \infty \tag{10.135}$$

which corresponds to a noncausal and unstable system. Comparing (10.134) to (10.34) or (10.38) we should use a type III or IV linear-phase system. A type III system has a zero at π and hence is unsuitable as a fullband differentiator. However, it provides an integer sample delay and can be used as a differentiator for signals up to 0.8π bandwidth. On the other hand, a type IV system provides a much better approximation to the amplitude response over the full bandwidth but has a noninteger delay which means that the derivative values are not obtained at signal sample locations, which may not be an issue in many applications.
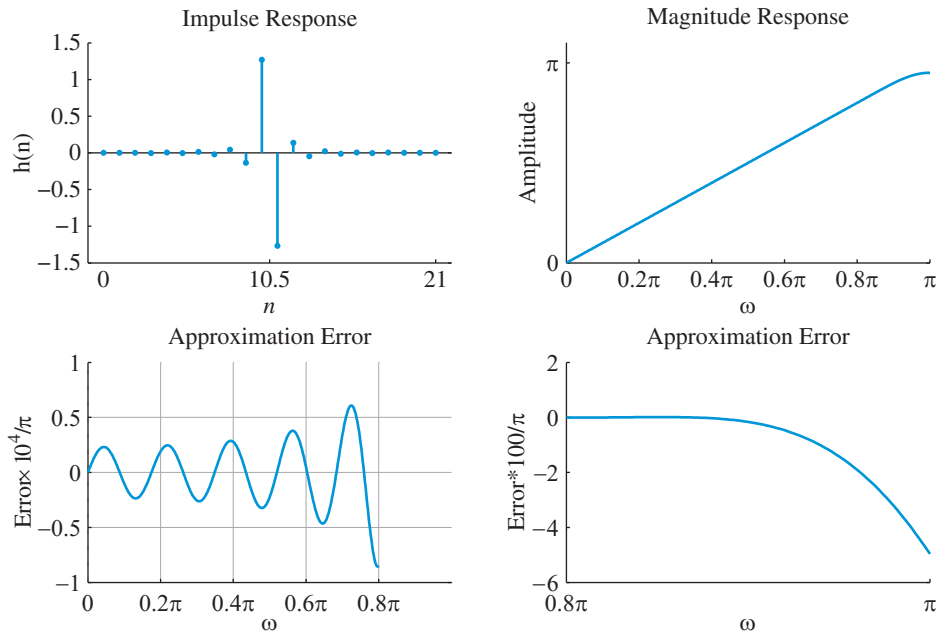
Figure 10.35 Filter response plots for the order $M = 21$ fullband differentiator designed in Example 10.10 using the Kaiser window method with $\beta = 4.5$.

Example 10.10

We design a type IV differentiator of order $M = 21$ that approximates (10.134) using the Kaiser window method and the Parks–McClellan algorithm. The Kaiser window design formula (10.84) is not appropriate for this design so we choose $\beta = 4.5$, which is approximately in the middle of the range in Figure 10.14. The following MATLAB script provides details of the design:

```
>> % Differentiator Specifications
>> M = 21; L = M+1; n = 0:M;
>> alpha = M/2; na = (n-alpha);
>> hd = cos(pi*na)./na - sin(pi*na)./(pi*na.^2);
>> % Differentiator design using Kaiser window
>> beta = 4.5; h = hd.*kaiser(L,beta)';
```

Figure 10.35 shows impulse and frequency response plots for the designed differentiator. Since $M = 21$, the delay in the output response is 10.5 as shown in the impulse response plot. The approximation error is much larger near π frequency compared to those in the band up to 0.8π. To obtain a better design we consider the Parks–McClellan algorithm. The details of its implementation in MATLAB are:

```
>> % Differentiator Design using FIRPM function
>> fo = [0,1]; % Band-edge array
>> ao = [0,pi]; % Band-edge desired slope
>> [h,delta] = firpm(M,fo,ao,'differentiator');
```
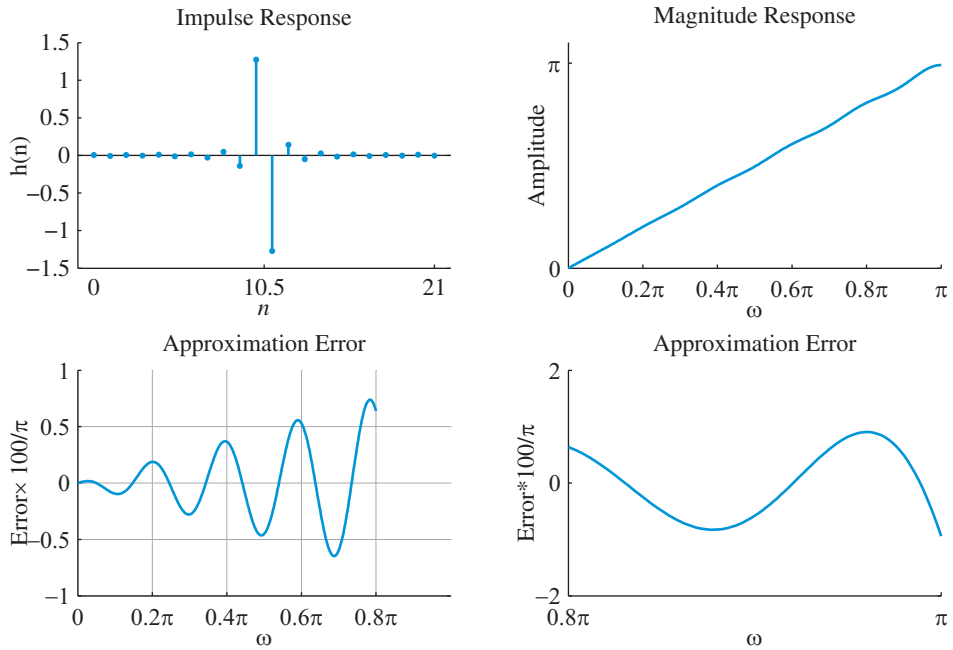
Figure 10.36 Filter response plots for the order $M = 21$ fullband differentiator designed in Example 10.10 using the Parks–McClellan algorithm.

Figure 10.36 shows impulse and frequency response plots for the designed differentiator. Although the approximation error appears to be increasing with frequency, the weighting function in the `firpm` is designed to minimize the maximum relative error (the maximum of the ratio of the error to the desired amplitude). The overall error is still lower compared to that in the Kaiser window design. ∎

Design of a type III differentiator using the frequency sampling approach is provided in Tutorial Problem 18.

10.7.2   Discrete-time Hilbert transformers

The ideal Hilbert transformer is an allpass system that introduces a 90-degree phase shift on the input signal. Such systems are used in a variety of narrowband modulation/demodulation schemes as well as in efficient sampling schemes for narrowband signals, see Proakis and Manolakis (2007) or Oppenheim and Schafer (2010). The discrete-time Hilbert transformer is defined by

$$H(\mathrm{e}^{\mathrm{j}\omega}) = \begin{cases} -\mathrm{j}\,\mathrm{sgn}(\omega)\mathrm{e}^{-\mathrm{j}\omega\alpha}, & \omega_1 < |\omega| < \omega_2 \\ 0, & \text{otherwise} \end{cases} \tag{10.136}$$

where the function $\text{sgn}(\omega) = 1$ for $\omega > 0$ and $\text{sgn}(\omega) = 0$ for $\omega < 0$. This system has impulse response given by

$$h[n] = \frac{\cos(\omega_1[n-\alpha]) - \cos(\omega_2[n-\alpha])}{\pi(n-\alpha)}, \qquad (10.137)$$

which corresponds to a noncausal and unstable system. Due to discontinuity in the amplitude response at 0 and π, a type III linear-phase system is well suited for designing Hilbert transformers.

Example 10.11

We design a type III fullband Hilbert transformer of order $M = 40$ using the Hamming window and frequency sampling methods. For the full-band design we choose $\omega_1 = 0$ and $\omega_2 = \pi$ in (10.137) and $\alpha = M/2 = 20$ to obtain the ideal impulse response. The MATLAB design steps are:

```
>> % Hilbert Transformer Specifications
>> M = 40; L = M+1; n = 0:M;
>> alpha = M/2; na = (n-alpha);
>> Dw = 2*pi/L; % Width between frequency samples
>> w1 = 0*Dw; w2 = L/2*Dw;
>> hd = (cos(w1*na)-cos(w2*na))./(pi*na); hd(alpha+1) = 0;
>> % Transformer design using Hamming window
>> h = hd.*hamming(L)';
```

Figure 10.37 shows impulse and magnitude response plots for the designed Hilbert transformer. Even though it was designed for the fullband, the resulting magnitude response shows that the filter is useful over the wideband approximately from 0.1π to 0.9π, which is sufficient for most applications. Also note that every other sample of the impulse response is approximately zero.

In frequency sampling design with $L = M + 1 = 41$ samples, there is no sample at π. Thus to reduce ripple amplitudes near π, we can let samples at 21 and 22 take variable
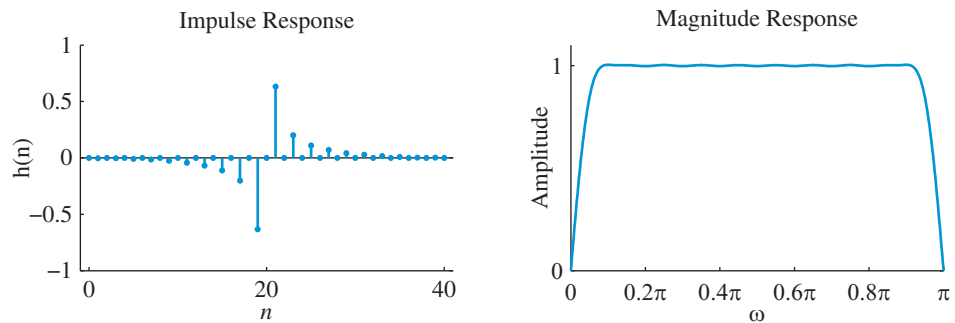
Figure 10.37 Filter response plots for the order $M = 40$ full-band Hilbert transformer designed in Example 10.11 using the Hamming window method.
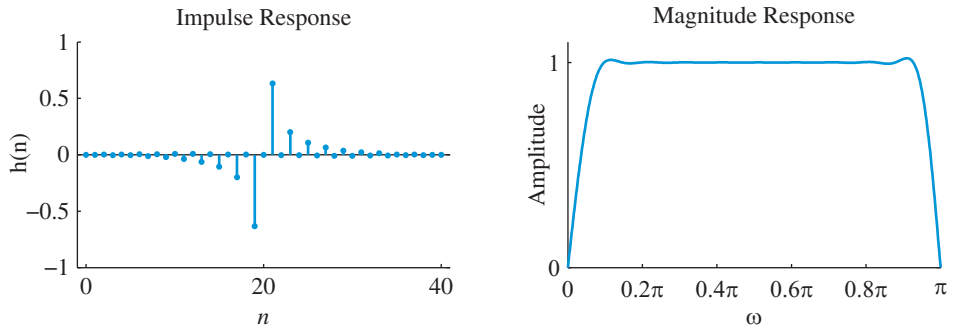
Figure 10.38 Filter response plots for the order $M = 40$ fullband Hilbert transformer designed in Example 10.11 using the frequency sampling method.

value. Similarly, the second and the last (41st) samples should also be variable to reduce ripple amplitudes around 0 frequency. We used the sine roll-off (a variation of raised-cosine approach) for smooth transition. These details are given in the MATLAB script below:

```
>> % Frequency Sampling Design: Assumed Parameters
>> M = 40; n = 0:M; L = M+1; % Impulse response length
>> alpha = M/2; Q = floor(alpha); % phase delay parameters
>> k = 0:M; % Frequency sample index
>> Dw = 2*pi/L; % Width between frequency samples
>> % Transformer Design using Frequency Sampling (Smooth transition)
>> Ad = [0,sin(pi/4),ones(1,18),0.5,-0.5,-ones(1,18),-sin(pi/4)];
>> psid = -alpha*2*pi/L*[(0:Q),-(L-(Q+1:M))]; % Desired Phase
>> Hd = -1j*Ad.*exp(1j*psid); % Desired Freq Resp Samples
>> hd = real(ifft(Hd)); % Desired Impulse response
>> h = hd.*rectwin(L)'; % Actual Impulse response
```

Figure 10.38 shows impulse and magnitude response plots for the designed Hilbert transformer. The magnitude response shows prominent ripple near the band edge which can be further reduced using optimum values for frequency samples near the band edges. The designed transformer is still useful over a wideband from 0.15π to 0.85π. ■

Another design of Hilbert transformer using the `firpm` function is given in Tutorial Problem 19.

10.7.3 Ideal raised-cosine pulse-shaping lowpass filters

A raised-cosine pulse-shaping filter is used in pulse transmission systems to prevent intersymbol interference. The frequency response function is defined by

$$H(e^{j\omega}) = \begin{cases} 1, & 0 \leq |\omega| < (1-\beta)\omega_c \\ \frac{1}{2}\left[1 - \sin\left(\frac{\pi}{2}\frac{\omega - \omega_c}{\beta\omega_c}\right)\right], & (1-\beta)\omega_c \leq |\omega| \leq (1+\beta)\omega_c \\ 0. & (1+\beta)\omega_c \leq |\omega| \leq \pi \end{cases} \quad (10.138)$$

The filter gain is equal to 1/2 at the nominal cutoff frequency ω_c. The parameter β, $0 < \beta < 1$, is known as the roll-off parameter. If the raised-cosine spectrum characteristic is divided between the transmitter and the receiver in a digital communication system, we use two filters, each with frequency response $\sqrt{H(e^{j\omega})}$ to achieve the desired result.

Example 10.12

We design a type-II raised-cosine pulse-shaping lowpass filter of length 40 that has $\omega_c = 0.4\pi$ and $\beta = 0.5$ using the frequency sampling method and a rectangular window. The MATLAB design steps are:

```
>> % Raised-cosine Pulse-shaping filter specifications
>> wc = 0.4*pi; beta = 0.5;
>> M = 39; L = M+1; n = 0:M; % Impulse response length
>> alpha = M/2; Q = floor(alpha); % phase delay parameters
>> om = linspace(-pi,pi,1001); % Frequency array
>> k = 0:M; % Frequency sample index
>> psid = -alpha*2*pi/L*[(0:Q),-(L-(Q+1:M))]; % Desired Phase
>> Dw = 2*pi/L; % Width between frequency samples
>> % Frequency Sampling Design
>> k1 = floor((1-beta)*wc/Dw); % Index for sample nearest to PB edge
>> k2 = ceil((1+beta)*wc/Dw);  % Index for sample nearest to SB edge
>> w = (k1:1:k2)*Dw; % Frequencies in the transition band
>> A = 0.5*(1-sin(pi/2*(w-wc)/(beta*wc))); % LS Trans band samples
>> B = fliplr(A); % Right side Transition band samples
>> Ad = [ones(1,k1),A,zeros(1,L-2*k2-1),B,ones(1,k1-1)]; % Desired
        Ampl
>> Hd = Ad.*exp(1j*psid); % Desired Freq Resp Samples
>> hd = real(ifft(Hd)); % Desired Impulse response
>> h = hd.*rectwin(L)'; % Actual Impulse response
>> H = freqz(h,1,om); % Frequency response of the actual filter
>> Hr = zerophase(h,1,om); % Amplitude Response
```

Figure 10.39 shows filter response plots for the designed raised-cosine pulse-shaping filter. The amplitude response plot in Figure 10.39(b) shows an excellent raised-cosine shape while the approximation error in Figure 10.39(c) shows that the errors are maximum at $\pm(1 \pm \beta)\omega_c$ of value less than 2×10^{-3}. ■

FDATool for special FIR filter design The special FIR filters discussed in this section can also be designed using the FDATool graphical user interface. Figure 10.40(a) shows the selection panel for the differentiator, Figure 10.40(b) shows the selection panel for the Hilbert transformer, and Figure 10.40(c) shows the selection panel for the raised-cosine filter. After selection, the required parameters like filter order, magnitude and frequency specifications, etc., can be entered in the respective panels. The SP toolbox manual should be consulted for complete details.
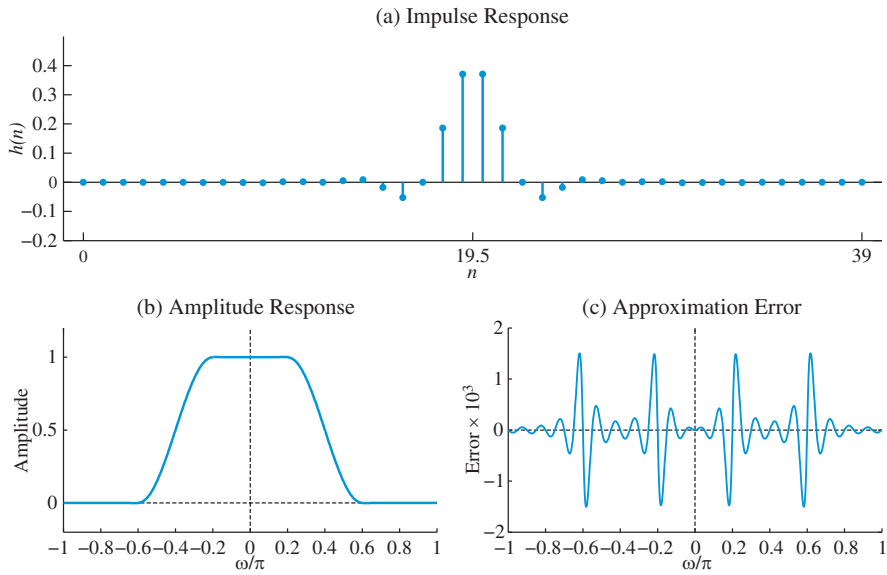
Figure 10.39 Filter response plots for the order $M = 40$ full-band Hilbert transformer designed in Example 10.12 using the Hamming window method.

(a) Differentiator

(b) Hilbert transformer

(c) Raised-cosine filter
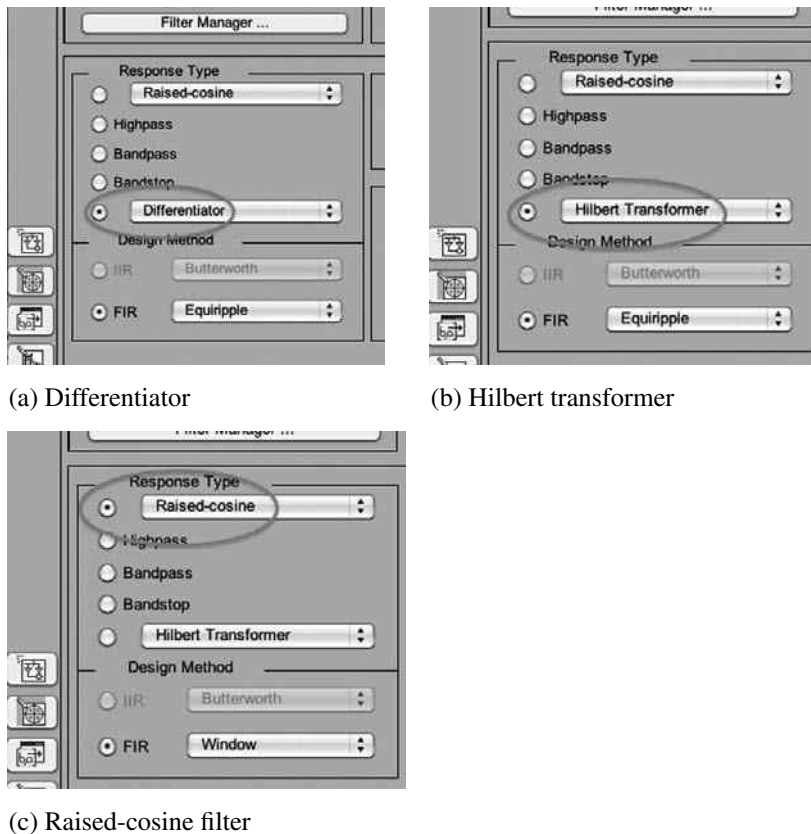
Figure 10.40 Graphical user interface panels of the FDATool for designing special FIR filters.

Learning summary

- The filter design problem consists of finding a practically realizable filter, that is, a causal and stable filter with a rational system function, whose frequency response best approximates the desired ideal magnitude and phase responses within specified tolerances.

- FIR filters with properly chosen symmetric or antisymmetric impulse responses have linear phase or equivalently constant group delay. Besides its importance in practical applications, this constraint simplifies the design of FIR by optimization techniques. Causal IIR filters cannot have linear phase.

- The most widely used techniques for designing FIR filters with prescribed frequency domain specifications are the windowing method, the frequency sampling method, and the Parks–McClellan algorithm. The Parks–McClellan design technique solves a minimax or Chebyshev approximation problem using the Remez exchange algorithm and produces optimum equiripple filters.

- The most useful techniques for filter design have been implemented on various computational environments, like MATLAB, and they are widely available in the form of filter design packages.

TERMS AND CONCEPTS

Absolute specifications Specifications in which nominal and ripples values are given in absolute (voltage) numbers.

Accumulated amplitude function A running integral of the amplitude response of a window function that is useful in determining its minimum stopband attenuation.

Adjustable windows A window function for which the minimum stopband attenuation can be adjusted using its length, for example, a Kaiser window.

Amplitude response A real-valued function that is similar to the magnitude response but can take both positive and negative values.

Analog specifications Specifications native to analog filters in which frequencies are specified in Hz, nominal values are given in absolute numbers, and ripples are given using parameters ϵ (passband) and A (stopband).

Angle response The companion response of the amplitude response that is similar to the phase response but is a linear function of ω for linear-phase FIR filters.

Bandedges Upper and lower limit frequencies of a band of a filter. For a lowpass filter the

lower bandedge is zero and for a highpass filter the upper bandedge is π radians.

Butterworth approximation A criterion in which a maximally-flat approximation in the band response is created.

Chebyshev approximation A criterion in which an equiripple approximation in the band response is created.

Chebyshev polynomial A polynomial of order m given by $T_m(x) = \cos[m\cos^{-1}(x)]$ which can also be generated using a recursive relation.

Cutoff frequency A characteristic frequency of an ideal frequency-selective filter at which there is a sudden transition from passband nominal value to the stopband nominal value, or vice versa.

Digital differentiator A discrete-time system that produces samples of the derivative of a continuous-time signal from its samples.

Digital Hilbert transformer A discrete-time system that produces the samples of the Hilbert transform of a continuous-time signal from its samples.

Equiripple property The alternate maxima and minima of a function over a band are of equal magnitude, that is, there are equal height peaks and valleys in the function.

Extraripple design An equiripple linear-phase FIR filter that achieves the maximum number of alternations in its band responses, thus producing one more ripple than most filters.

Extremal frequencies Frequencies where maxima and minima in an error function are achieved.

Filter design and analysis tool (FDATool) Graphical user interface based digital filter design and analysis software.

Fixed windows A window function for which the minimum stopband attenuation is fixed irrespective of its length, for example, a Hamming window.

Frequency-selective filter A system that facilitates or attenuates signals based on their frequency components and typically implies a lowpass, highpass, bandpass, and bandstop filter.

Linear-phase filters FIR filters that exhibit phase responses that are linear functions of frequency ω. There are four such types.

Maximally-flat approximation A criterion in which a finite number of lower derivatives of two functions at a frequency are made equal resulting in a maximally flat response.

Minimax approximation An optimality criterion in which the maximum error between two functions is minimized.

Mirror-image polynomial A real-coefficient polynomial $H(z)$ that satisfies the condition $H(z) = z^{-M}H(z^{-1})$.

MSE approximation An optimality criterion in which the mean squared error between two functions is minimized.

Nominal value A magnitude response value specified by the ideal frequency-selective filter in each band.

Parks–McClellan algorithm An algorithm that designs equiripple linear-phase FIR filters given frequency and magnitude specifications.

Passband A band of frequencies that are passed by a filter with no or negligible attenuation.

Raised-cosine pulse shaping filter A discrete-time system that is used to prevent intersymbol interference in a pulse transmission system.

Relative specifications Specifications in which nominal and ripples values are given in log-magnitude or decibel (dB) numbers.

Ripple A small variation (oscillation) in the frequency response around the nominal value in a frequency band of a practical filter.

Roll-off A gradual transition from one nominal value to another in a transition band.

Stopband A band of frequencies that are almost completely eliminated by a filter.

Tolerance See ripple.

Transition band A guard band between a passband and a stopband in a practical filter in which the filter response gradually attenuates from passband nominal value to stopband nominal value.

Type-I FIR filter An even-order causal FIR filter with symmetric impulse response.

Type-II FIR filter An odd-order causal FIR filter with symmetric impulse response.

Type-III FIR filter An even-order causal FIR filter with antisymmetric impulse response.

Type-IV FIR filter An odd-order causal FIR filter with antisymmetric impulse response.

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A detailed treatment of discrete-time filter design, at the same level as in this book, is given in Oppenheim and Schafer (2010), Proakis and Manolakis (2007), Mitra (2006), and Rabiner and Gold (1975).
2. The books by Parks and Burrus (1987) and Antoniou (2006) place more emphasis on filter design and provide more details regarding the design of FIR (Parks–McClellan) and IIR (elliptic) filters with equiripple responses.
3. An extensive review of filter design techniques, beyond those discussed in this chapter, is provided by Karam *et al.* (2009) and Saramaki (1993).

Review questions

1. How many stages are needed in designing frequency-selective discrete-time filters? Describe each stage concisely.
2. How do practical frequency-selective filters differ from ideal filters and why?
3. Describe three different approaches used in practice that provide specifications of realizable frequency-selective filters. Why are these approaches needed?
4. Some useful discrete-time filters cannot be described using tolerance schemes. Identify these filters and explain why.

5. It is claimed that a stable and causal ideal frequency-selective filter can be obtained. Do you agree or disagree? If you agree provide an example of such a filter. If you disagree provide a proof of your disagreement.

6. Given the magnitude response of a causal and stable filter, why can we not assign its phase response arbitrarily? How do we deal with this problem in practice?

7. Explain the interdependence between magnitude and phase response of causal and stable systems.

8. Practical filters are designed to satisfy optimality criteria that approximate some ideal filter responses. Describe such criteria used in the chapter. List filter design techniques resulting from these criteria.

9. What is the main purpose and outcome of any filter design technique? How do these outcomes differ for FIR and IIR filters?

10. It is said that IIR filters cannot have linear phase. Do you agree or disagree? Explain.

11. A delayed impulse response of an ideal frequency-selective continuous-time filter is always symmetric but its sampled version is not always a symmetric discrete-time signal. Why? Explain the conditions under which it can be symmetric.

12. Explain the four types of linear-phase discrete-time filters and the resulting impulse responses.

13. Explain the difference between an amplitude response and magnitude response of a filter. How would you describe their corresponding phase responses?

14. Describe the amplitude and angle responses of the four types of linear-phase FIR filter.

15. Describe the zero-pole distributions of the four types of linear-phase FIR filter.

16. What is the unified amplitude response representation for the four types of linear-phase FIR filter? Describe it for each type.

17. Explain the basic philosophy behind FIR filter design by windowing.

18. Why do we get ripples in the passband and stopband responses due to the windowing operation?

19. What are the frequency-domain effects of the windowing operation and how are they used to create design strategies?

20. What is an accumulated amplitude function and what is its importance?

21. List "fixed windows" used in the window designs and explain their design parameters.

22. Why is the Kaiser window termed an adjustable window? What does it adjust and how does it do it?

23. Explain the basic philosophy behind the FIR filter design by frequency-sampling technique.

24. What are the different design approaches in the frequency-sampling design technique? Explain these approaches.

25. Describe the design procedure step-by-step for the frequency-sampling technique.

26. What is the minimax criterion? Explain the relationship between the Chebyshev polynomials and the minimax criterion.

27. Describe the shape of the sixth-order Chebyshev polynomial for $|x| \leq 1$.

28. What does the alternation theorem tell us and how does it help us to understand the minimax polynomial approximation problem?

29. What are the shortcomings in the window and frequency-sampling design techniques and how are these addressed by the Parks–McClellan algorithm.

30. Concisely explain the workings of the Parks–McClellan algorithm.

31. What are the special FIR filters and what are their intended applications?

Problems

Tutorial problems

1. This problem examines conversions between various filter specifications.
    (a) Given the absolute specifications $\delta_p = 0.01$ and $\delta_s = 0.0001$, determine the relative specifications A_p, A_s, and the analog filter specifications ϵ, A.
    (b) Given the analog filter specifications $\epsilon = 0.25$ and $A = 200$, determine the relative specifications A_p, A_s, and the absolute specifications δ_p, δ_s.

2. Using (10.14) and the fact that $H_R(e^{j\omega})$ is the DTFT of $h_e[n]$ prove (10.16).

3. Consider the type-II linear-phase FIR filter characterized by symmetric impulse response and odd-M.
    (a) Show that the amplitude response $A(e^{j\omega})$ is given by (10.31) with coefficients $b[k]$ given in (10.32).
    (b) Show that the amplitude response $A(e^{j\omega})$ can be further expressed as (10.33) with coefficients $\tilde{b}[k]$ given in (10.34).

4. Consider an FIR filter with impulse response $h[n] = u[n] - u[n-4]$.
    (a) Determine and sketch the magnitude response $|H(e^{j\omega})|$.
    (b) Determine and sketch the amplitude response $A(e^{j\omega})$. Compare this sketch with that in (a) and comment on the difference.
    (c) Determine and sketch the phase response $\angle H(e^{j\omega})$.
    (d) Determine and sketch the angle response $\Psi(e^{j\omega})$. Compare this sketch with that in (c) and comment on the difference.

5. Prove expressions for $A(e^{j\omega})$ for $M = 6$ type-III and $M = 5$ type-IV linear-phase FIR filters.

6. In this problem explore the use of the book toolbox function `amplresp`.
    (a) Let $h_1[n] = \{1, -2, 3, -4, 5, -4, 3, -2, 1\}$. Using `amplresp` compute and plot the amplitude response from $h_1[n]$.
    (b) Let $h_2[n] = \{1, -2, 3, -4, 5, 5, -4, 3, -2, 1\}$. Using `amplresp` compute and plot the amplitude response from $h_2[n]$.
    (c) Let $h_3[n] = \{1, -2, 3, -4, 0, 4, -3, 2, -1\}$. Using `amplresp` compute and plot the amplitude response from $h_3[n]$.
    (d) Let $h_4[n] = \{1, -2, 3, -4, 5, -5, 4, -3, 2, -1\}$. Using `amplresp` compute and plot the amplitude response from $h_4[n]$.

7. In this problem we reproduce Figures 10.4 and 10.5. For each of the following linear-phase FIR filters described by $h[n]$, obtain impulse response, amplitude response,

angle response, and pole-zero plots in one figure window. For frequency response plots use the interval $-2\pi \le \omega \le 2\pi$.

(a) Type-I filter: $h[n] = \{\underset{\uparrow}{1}, 2, 3, -2, 5, -2, 3, 2, 1\}$.

(b) Type-II filter: $h[n] = \{\underset{\uparrow}{1}, 2, 3, -2, -2, 3, 2, 1\}$.

(c) Type-III filter: $h[n] = \{\underset{\uparrow}{1}, 2, 3, -2, 0, 2, -3, -2, -1\}$.

(d) Type-IV filter: $h[n] = \{\underset{\uparrow}{1}, 2, 3, -2, 2, -3, -2, -1\}$.

Comment on the symmetry properties of the amplitude responses and the centers of symmetry of the impulse responses.

8. Consider the rectangular window of length $L = 21$.

(a) Compute and plot the log-magnitude response in dB over $-\pi \le \omega \le \pi$. In the plot measure and show the value of the peak of the first sidelobes.

(b) Compute and plot the accumulated amplitude response in dB using the `cumsum` function. In the plot measure and show the value of the peak of the first sidelobe. Also obtain from the plot the exact transition bandwidth by measuring the interval between the peaks on either side of $\omega = 0$. Express this bandwidth as a function of π/M.

(c) Repeat (a) and (b) for $L = 41$.

9. Consider the multiband ideal filter given by the amplitude response

$$
A(e^{j\omega}) = \begin{cases}
0, & 0 < |\omega| < 0.2\pi \\
0.5, & 0.2\pi < |\omega| < 0.4\pi \\
1, & 0.4\pi < |\omega| < 0.6\pi \\
0.5, & 0.6\pi < |\omega| < 0.8\pi \\
1. & 0.8\pi < |\omega| < \pi.
\end{cases}
$$

(a) Obtain a linear-phase FIR filter using the rectangular window of order $M = 10$. Compute and plot the amplitude response over $-\pi \le \omega \le \pi$.

(b) Repeat (a) using $M = 20$, $M = 40$, and $M = 60$.

(c) From the plots in (a) and (b) comment on which ones appear to satisfy (10.78) and (10.79). How far apart should the amplitude response discontinuities be so that (10.78) and (10.79) are satisfied?

10. Design a highpass FIR filter to satisfy the specifications: $\omega_s = 0.3\pi$, $A_s = 50$ dB, $\omega_p = 0.0.5\pi$, and $A_p = 0.001$ dB.

(a) Use an appropriate fixed window to obtain a minimum length linear-phase filter. Provide a plot similar to Figure 10.12.

(b) Repeat (a) using the `fir1` function.

11. Consider an ideal lowpass filter with cutoff frequency $\omega_c = \pi/2$.

(a) Using $L = 20$ samples around the unit circle, compute the resulting impulse response $h[n]$ using the rectangular window in (10.91). Compute and plot the magnitude response over $0 \le \omega \le 2\pi$ and show the frequency samples on the magnitude response plot.

(b) Using $L = 400$ samples around the unit circle, compute one period of the resulting periodic impulse response $\tilde{h}[n]$ in (10.90) and then the impulse response $h[n]$ using

the rectangular window in (10.91). Compute and plot the magnitude response over $0 \leq \omega \leq 2\pi$. Compare the plot with the magnitude response plot in (a) and comment.

(c) Repeat part (b) using Hamming window.

12. Consider the lowpass filter specifications: $\omega_p = 0.2\pi$, $\omega_s = 0.3\pi$, $A_p = 0.2$ dB, and $A_s = 40$ dB.

(a) Design a length $L = 20$ linear-phase FIR filter using the basic frequency sampling technique. Graph the relevant filter response plots.

(b) You should note that the design in (a) cannot satisfy the given specifications. Hence choose $L = 40$ and using frequency sampling design with an optimum approach, design a linear-phase FIR filter. Use the transition coefficient tables given in Proakis and Manolakis (2007) Appendix B. Graph the relevant filter response plots and comment on the design.

(c) Repeat (b) using the `fir2` function.

13. In this problem we develop the Chebyshev polynomials.

(a) Using the trigonometric identity $\cos(A+B) = \cos(A)\cos(B) - \sin(A)\sin(B)$ show that

$$\cos[(n+1)\omega] = 2\cos(\omega)\cos(n\omega) - \cos[(n-1)\omega], \quad n \geq 1.$$

(b) Use the formula in (a) to derive the following recursive formula

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \quad n \geq 1.$$

(c) Use the recursion in (b) to derive the first five Chebyshev polynomials.

14. Consider the polynomial $f(x) = 1 - 2x + 4x^2 - 2x^3$. We want to approximate it using a second-order polynomial $P_2(x) = a_0 + a_1 x + a_2 x^2$ so that the error $e(x) \triangleq f(x) - P_2(x)$ is equiripple over $0 \leq x \leq 1$.

(a) Choose an initial set of $m + 2 = 4$ nodes $\{\xi_k\}_{k=0}^3$ as 0, 1/3, 2/3, and 1 and solve for coefficients $\{a_\ell\}_{\ell=0}^2$ and δ using (10.110). Graph $f(x)$, the resulting $P_2(x)$, and $e(x)$ in one plot.

(b) Using your plot and with care determine the new set of nodes for which $e(x)$ has the maximum error. Use this new set and repeat (a) until the error $e(x)$ is nearly equiripple over $0 \leq x \leq 1$. Graph the final $f(x)$, the resulting $P_2(x)$, and $e(x)$ in one plot.

15. Consider the design of a type-I lowpass filter with $M = 10$, $\omega_p = 0.25\pi$, and $\omega_s = 0.375\pi$. Choose the seven initial frequency set as $\{0, \omega_p, \omega_s, 0.5\pi, 0.7\pi, 0.9\pi, \pi\}$.

(a) Using Problem 14 as a guide, solve the linear system (10.124) to obtain $a[0], a[1] \ldots, a[5]$, and δ. Plot the magnitude response of the resulting filter over $0 \leq \omega \leq \pi$, show tolerance values in each band, and indicate the magnitude values at the initial seven frequencies on the plot. Your plot should be similar to the one in Figure 10.28.

(b) Using your plot and with care determine the new set of extremal frequencies for which the error function has the maximum error over $\mathcal{B}$. Use this new set and repeat (a) until the error function is nearly equiripple over $\mathcal{B}$. Your plot should be similar to the one in Figure 10.29. What is the resulting value of δ?

16. Design a highpass FIR filter using the Parks–McClellan algorithm to satisfy the specifications: $\omega_s = 0.6\pi$, $\omega_p = 0.75\pi$, $A_s = 50$ dB, and $A_p = 0.5$ dB. Graph all relevant filter response plots.

17. Specifications of a multiband digital filter are given below:

$$|H(e^{j\omega})| = \pm 0.001, \qquad\qquad 0 \le |\omega| \le 0.3\pi$$

$$|H(e^{j\omega})| = 0.5 \pm 0.005, \qquad\qquad 0.4\pi \le |\omega| \le 0.7\pi$$

$$|H(e^{j\omega})| = 1 \pm 0.01. \qquad\qquad 0.75\pi \le |\omega| \le \pi$$

Design a linear-phase FIR filter using the Parks–McClellan algorithm.

18. Using the frequency sampling approach design a wideband type-III differentiator of order $M = 32$. Graph its amplitude and impulse response and determine the usable bandwidth of the differentiator.

19. Design a length $L = 50$ Hilbert transformer over the band $0.05\pi \le |\omega| \le 0.95\pi$ using the Parks–McClellan algorithm. Graph its amplitude and impulse response and determine the maximum ripple obtained over the given band.

20. We want to design the raised-cosine pulse-shaping filter given in (10.138) with $\omega_c = 0.5\pi$ and $\beta = 0.75$ using the Parks–McClellan algorithm. Consider it as a three band filter. Develop appropriate band-edge frequencies and nominal values to design a length $L = 41$ filter with equal ripple values in each band of 0.01. Provide an amplitude plot to verify your design.

Basic problems

21. Design the following MATLAB function that implements conversions between various filter specifications:

```
[A,B]=spec_convert(C,D,typein,typeout)
%  typein: 'abs' or 'rel' or 'ana'
% typeout: 'abs' or 'rel' or 'ana'
%     C,D: input specifications
%     A,B: output specifications
```

22. Consider the type-III linear-phase FIR filter characterized by antisymmetric impulse response and even M.
    (a) Show that the amplitude response $A(e^{j\omega})$ is given by (10.36) with coefficients $c[k]$ given in (10.37).
    (b) Show that the amplitude response $A(e^{j\omega})$ can be further expressed as (10.38) with coefficients $\tilde{c}[k]$ given in (10.39).

23. Let the ideal lowpass filter be given by $H_{lp}(e^{j\omega}) = e^{-jn_d\omega}$, $|\omega| \le \pi$.
    (a) Determine the impulse response $h_{lp}[n]$.
    (b) Show that this ideal filter introduced a delay of n_d samples in the input signal.

24. Consider the amplitude response in (10.62). Using procedure similar to that used in obtaining (10.66), show that for $\omega > \omega_c$

$$A(e^{j\omega}) \approx \frac{1}{2} - \frac{1}{\pi}\text{Si}\big[(\omega - \omega_c)L/2\big].$$

25. The Hann window function can be written as

$$w[n] = [0.5 - 0.5\cos(2\pi n/M)]w_R[n].$$

where $w_R[n]$ is the rectangular window of length $M + 1$.
   (a) Express the DTFT of $w[n]$ in terms of the DTFT of $w_R[n]$.
   (b) Explain why the Hann window has the wider mainlobe but lower sidelobes than the rectangular window of the same length.

26. Consider the Bartlett window of length $L = 21$.
   (a) Compute and plot the log-magnitude response in dB over $-\pi \leq \omega \leq \pi$. In the plot measure and show the value of the peak of the first sidelobes.
   (b) Compute and plot the accumulated amplitude response in dB using the `cumsum` function. In the plot measure and show the value of the peak of the first sidelobe. Also obtain from the plot the exact transition bandwidth by measuring the interval between the peaks on either side of $\omega = 0$. Express this bandwidth as a function of π/M.
   (c) Repeat (a) and (b) for $L = 41$.

27. Consider the Hamming window of length $L = 21$.
   (a) Compute and plot the log-magnitude response in dB over $-\pi \leq \omega \leq \pi$. In the plot measure and show the value of the peak of the first sidelobes.
   (b) Compute and plot the accumulated amplitude response in dB using the `cumsum` function. In the plot measure and show the value of the peak of the first sidelobe. Also obtain from the plot the exact transition bandwidth by measuring the interval between the peaks on either side of $\omega = 0$. Express this bandwidth as a function of π/M.
   (c) Repeat (a) and (b) for $L = 41$.

28. Design a lowpass FIR filter to satisfy the specifications: $\omega_p = 0.3\pi$, $A_p = 0.5$ dB, $\omega_s = 0.5\pi$, and $A_s = 50$ dB.
   (a) Use an appropriate fixed window to obtain a minimum length linear-phase filter. Provide a plot similar to Figure 10.12.
   (b) Repeat (a) using the Kaiser window and compare the lengths of the resulting filters.

29. Specifications of a bandstop filter are: $\omega_{p_1} = 0.2\pi$, $\delta_p = 0.056$, $\omega_{s_1} = 0.3\pi$, $\omega_{s_2} = 0.5\pi$, $\delta_s = 0.01$, $\omega_{p_2} = 0.65\pi$, and $\delta_p = 0.056$.
   (a) Design a minimum length linear-phase FIR filter using the Hann window. Provide a plot similar to Figure 10.17.
   (b) Verify your design using the `fir1` function.

30. A bandpass filter is given by the specifications: $\omega_{s_1} = 0.2\pi$, $A_{s_1} = 45$ dB, $\omega_{p_1} = 0.3\pi$, $\omega_{p_2} = 0.5\pi$, $A_p = 0.75$ dB, $\omega_{s_2} = 0.65\pi$, and $A_{s_2} = 50$ dB.
   (a) Design a minimum length linear-phase FIR filter using one of the fixed type windows. Provide a plot similar to Figure 10.17.
   (b) Repeat (a) using the Kaiser window.
   (c) Verify your designs using the `fir1` function.

31. Design a type-IV differentiator of order $M = 61$ that approximates (10.134) using the Blackman window. Provide a plot containing the amplitude and the impulse responses. Verify your design using the `fir1` function.

32. We want to use the frequency-sampling method to design a highpass filter with specifications: $\omega_s = 0.6\pi$, $\omega_p = 0.8\pi$, $A_s = 50$ dB, and $A_p = 1$ dB.
    (a) Choose $M = 33$ so that there are two samples in the transition band. Using a linear transition obtain the filter impulse response. Provide a plot of the log-magnitude and impulse responses. Does this design satisfy the given specifications?
    (b) Repeat (a) using the `fir2` function and the Hamming window. Does this design satisfy the given specifications?

33. A bandpass filter is given by the specifications: $\omega_{s_1} = 0.2\pi$, $A_{s_1} = 40$ dB, $\omega_{p_1} = 0.3\pi$, $\omega_{p_2} = 0.5\pi$, $A_p = 0.2$ dB, $\omega_{s_2} = 0.65\pi$, and $A_{s_2} = 40$ dB.
    (a) Choose $L = 40$ so that there are two samples in the transition band. Using a raised-cosine transition obtain the filter impulse response. Provide a plot of the log-magnitude and impulse responses. Does this design satisfy the given specifications?
    (b) Repeat (a) using the `fir2` function and the Hann window. Does this design satisfy the given specifications?

34. An ideal lowpass filter has a cutoff frequency of $\omega_c = 0.4\pi$. We want to obtain a length $L = 40$ linear-phase FIR filter using the frequency-sampling method.
    (a) Let the sample at ω_c be equal to 0.5. Obtain the resulting impulse response $h[n]$. Plot the log-magnitude response in dB and determine the minimum stopband attenuation.
    (b) Now vary the value of the sample at ω_c (up to four decimals) and find the largest minimum stopband attenuation. Obtain the resulting impulse response $h[n]$ and plot the log-magnitude response in dB in the plot window of (a).
    (c) Compare your results with those obtained using the `fir2` function (choose an appropriate window).

35. Design a length $L = 50$ FIR differentiator using the frequency-sampling method.
    (a) Graph the impulse and amplitude responses of the designed differentiator in one plot.
    (b) Generate 151 samples of the signal $x[n] = 10\cos(0.2\pi n)$, $0 \le n \le 150$ and process them through the differentiator designed in (a) to obtain $y[n]$. Provide `stem` plots of both $x[n]$ and $y[n]$ for $50 \le n \le 100$ as sub-plots in one figure.
    (c) Can you confirm that $y[n]$ corresponds to the samples of the derivative of the signal whose samples are given by $x[n]$?

36. Specifications of a bandpass digital filter are given below:

$$|H(e^{j\omega})| = \pm 0.01, \qquad\qquad 0 \le |\omega| \le 0.25\pi$$

$$|H(e^{j\omega})| = 1 \pm 0.004, \qquad\qquad 0.35\pi \le |\omega| \le 0.7\pi$$

$$|H(e^{j\omega})| = \pm 0.01. \qquad\qquad 0.8\pi \le |\omega| \le \pi$$

Design a linear-phase FIR filter using the Parks–McClellan algorithm. Provide a filter response plot similar to Figure 10.32.

37. A lowpass FIR filter is given by the specifications: $\omega_p = 0.3\pi$, $A_p = 0.5$ dB, $\omega_s = 0.5\pi$, and $A_s = 50$ dB. Use the Parks–McClellan algorithm to obtain a minimum length linear-phase filter. Provide a plot similar to Figure 10.12.

38. Consider a multiband filter given by the specifications:
   - Band-1: $0 \leq \frac{\omega}{\pi} \leq 0.4$, $0.3 \leq A(e^{j\omega}) \leq 0.4$,
   - Band-2: $0.5 \leq \frac{\omega}{\pi} \leq 0.7$, $0.95 \leq A(e^{j\omega}) \leq 1$,
   - Band-3: $0.8 \leq \frac{\omega}{\pi} \leq 1$, $0.05 \leq A(e^{j\omega}) \leq 0.45$.

   Design a minimum length linear-phase filter using the Parks–McClellan algorithm . Provide a plot similar to Figure 10.32.

39. Design an $M = 25$ order FIR Hilbert transformer using the Parks–McClellan algorithm.
   (a) Graph the impulse and amplitude responses of the designed transformer in one plot.
   (b) Generate 101 samples of the signal $x[n] = \cos(0.3\pi n)$, $0 \leq n \leq 100$ and process them through the transformer designed in (a) to obtain $y[n]$. Provide `stem` plots of both $x[n]$ and $y[n]$ for $25 \leq n \leq 75$ as sub-plots in one figure.
   (c) Can you confirm that $y[n]$ corresponds to the samples of the Hilbert transform of the signal whose samples are given by $x[n]$?

40. The zeroth-order modified Bessel function of the first kind is given in (10.83).
   (a) Using the `Izero` function determine the number of terms K needed in (10.83) so that $I_0(x)$ is accurate from 1 to 8 decimals over $20 \leq x \leq 80$. Assume that the accurate value of $I_0(x)$ is obtained when $K = 50$. Determine the value of K for practical convergence.
   (b) Using the `Izero` function and the above value of K design the function `kaiser0(L,beta)` and compare its values with those obtained using the `kaiser` function.

41. In this problem we compare the Kaiser window to fixed windows in terms of A_s and transition bandwidth. Consider a design of a lowpass filter with a cutoff frequency of $\omega_c = \pi/2$. We want to design linear-phase FIR filters of order $M = 32$ using the window technique.
   (a) Use fixed windows to design filters of order 32 and accurately measure their minimum stopband attenuation A_s and transition bandwidth assuming equal ripples in pass- and stopbands.
   (b) Use a Kaiser window of length 33 using β from 1 through 9 in integer steps and design the respective filters. Accurately measure their minimum stopband attenuation A_s and determine the resulting transition bandwidth using (10.85).
   (c) Plot A_s versus transition-widths obtained in (b) and indicate the corresponding pairs for the fixed windows obtained in (a). Comment on your plot.

Assessment problems

42. Consider the type-IV linear-phase FIR filter characterized by antisymmetric impulse response and odd-M.
   (a) Show that the amplitude response $A(e^{j\omega})$ is given by (10.40) with coefficients $d[k]$ given in (10.41).
   (b) Show that the amplitude response $A(e^{j\omega})$ can be further expressed as (10.42) with coefficients $\tilde{d}[k]$ given in (10.43).

43. The Blackman window function can be written as

$$w[n] = \big[0.42 - 0.5\cos(2\pi n/M) + 0.08\cos(4\pi n/M)\big]w_{\mathrm{R}}[n],$$

where $w_{\mathrm{R}}[n]$ is the rectangular window of length $M+1$.

(a) Express the DTFT of $w[n]$ in terms of the DTFT of $w_{\mathrm{R}}[n]$.

(b) Explain why the Blackman window has the wider mainlobe but lower sidelobes than the rectangular window of the same length.

44. Following the steps leading to (10.65)–(10.67) and also Problem 24 show that the amplitude response relation for the nonrectangular windows is given by (10.78).

45. Consider Hann window of length $L = 21$.

(a) Compute and plot the log-magnitude response in dB over $-\pi \le \omega \le \pi$. In the plot measure and show the value of the peak of the first sidelobes.

(b) Compute and plot the accumulated amplitude response in dB using the `cumsum` function. In the plot measure and show the value of the peak of the first sidelobe. Also obtain from the plot the exact transition bandwidth by measuring the interval between the peaks on either side of $\omega = 0$. Express this bandwidth as a function of π/M.

(c) Repeat (a) and (b) for $L = 41$.

46. Consider a Blackman window of length $L = 21$.

(a) Compute and plot the log-magnitude response in dB over $-\pi \le \omega \le \pi$. In the plot measure and show the value of the peak of the first sidelobes.

(b) Compute and plot the accumulated amplitude response in dB using the `cumsum` function. In the plot measure and show the value of the peak of the first sidelobe. Also obtain from the plot the exact transition bandwidth by measuring the interval between the peaks on either side of $\omega = 0$. Express this bandwidth as a function of π/M.

(c) Repeat (a) and (b) for $L = 41$.

47. Design a highpass FIR filter to satisfy the specifications: $\omega_s = 0.4\pi$, $A_s = 60\,\mathrm{dB}$, $\omega_p = 0.5\pi$, and $A_p = 1\,\mathrm{dB}$.

(a) Use an appropriate fixed window to obtain a minimum length linear-phase filter. Provide a plot similar to Figure 10.12.

(b) Repeat (a) using the Kaiser window and compare the lengths of the resulting filters.

48. A bandpass filter is given by the specifications: $\omega_{s_1} = 0.25\pi$, $\delta_s = 0.05$, $\omega_{p_1} = 0.35\pi$, $\omega_{p_2} = 0.65\pi$, $\delta_p = 0.01$, $\omega_{s_2} = 0.75\pi$, and $\delta_s = 0.05$.

(a) Design a minimum length linear-phase FIR filter using one of the fixed windows. Provide a plot similar to Figure 10.17.

(b) Repeat (a) using the Kaiser window.

(c) Verify your designs using the `fir1` function.

49. Specifications of a bandstop filter are: $\omega_{p_1} = 0.4\pi$, $A_{p_1} = 0.5\,\mathrm{dB}$, $\omega_{s_1} = 0.55\pi$, $\omega_{s_2} = 0.65\pi$, $A_s = 55\,\mathrm{dB}$, $\omega_{p_2} = 0.75\pi$, and $A_{p_2} = 1\,\mathrm{dB}$.

(a) Design a minimum length linear-phase FIR filter using the Kaiser window. Provide a plot similar to Figure 10.17.

(b) Verify your design using the `fir1` function.

50. Design a type-III Hilbert transformer of order $M = 40$ that approximates (10.137) using the Hamming window. Provide a plot containing the amplitude and the impulse responses. Verify your design using the `fir1` function.

51. A lowpass FIR filter is given by the specifications: $\omega_p = 0.3\pi$, $A_p = 0.5$ dB, $\omega_s = 0.5\pi$, and $A_s = 50$ dB. Use the `fir2` function to obtain a minimum length linear-phase filter. Use the appropriate window function in the `fir2` function. Provide a plot similar to Figure 10.12.

52. We want to use the frequency-sampling method to design a highpass filter with specifications: $\omega_s = 0.5\pi$, $\omega_p = 0.65\pi$, $A_s = 50$ dB, and $A_p = 1$ dB.
    (a) Choose $M = 31$ so that there are two samples in the transition band. Using a raised-cosine transition obtain the filter impulse response. Provide a plot of the log-magnitude and impulse responses. Does this design satisfy the given specifications?
    (b) Repeat (a) using the `fir2` function and the Hamming window. Does this design satisfy the given specifications?

53. A bandpass filter is given by the specifications: $\omega_{s_1} = 0.2\pi$, $A_{s_1} = 40$ dB, $\omega_{p_1} = 0.3\pi$, $\omega_{p_2} = 0.5\pi$, $A_p = 0.2$ dB, $\omega_{s_2} = 0.65\pi$, and $A_{s_2} = 40$ dB.
    (a) Choose $L = 40$ so that there are two samples in the transition band. Using a linear transition obtain the filter impulse response. Provide a plot of the log-magnitude and impulse responses. Does this design satisfy the given specifications?
    (b) Repeat (a) using the `fir2` function and the Hann window. Does this design satisfy the given specifications?

54. Design a bandstop filter using the frequency-sampling technique. The specifications are: $\omega_{p_1} = 0.3\pi$, $\delta_p = 0.02$, $\omega_{s_1} = 0.4\pi$, $\omega_{s_2} = 0.6\pi$, $\delta_s = 0.0032$, $\omega_{p_2} = 0.7\pi$, and $\delta_p = 0.02$. Choose the length of the filter so that there are two samples in the transition band. Obtain the impulse response of the filter using the optimum values for the transition-band samples (see Problem 12). Provide a plot similar to Figure 10.17. Compare your results with those obtained using the `fir2` function (choose an appropriate window).

55. Design an $M = 50$ order FIR Hilbert transformer using the frequency sampling method.
    (a) Graph the impulse and amplitude responses of the designed transformer in one plot.
    (b) Generate 151 samples of the signal $x[n] = \sin(0.3\pi n)$, $0 \le n \le 150$ and process them through the transformer designed in (a) to obtain $y[n]$. Provide `stem` plots of both $x[n]$ and $y[n]$ for $50 \le n \le 100$ as sub-plots in one figure.
    (c) Can you confirm that $y[n]$ corresponds to the samples of the Hilbert transform of the signal whose samples are given by $x[n]$?

56. Consider the design of a type-I lowpass filter with $M = 10$, $\omega_p = 0.25\pi$, and $\omega_s = 0.375\pi$. Choose the seven initial frequency set as $\{0, \omega_p, \omega_s, 0.5\pi, 0.7\pi, 0.9\pi, \pi\}$.
    (a) Using (10.125) through (10.128) obtain amplitude response $A(e^{j\omega})$ over $0 \le \omega \le \pi$ and δ. Plot $A(e^{j\omega})$, show tolerance values in each band, and indicate the amplitude values at the initial seven frequencies on the plot. Your plot should be similar to the one in Figure 10.28.

(b) Using your plot and with care determine the new set of extremal frequencies for which the error function has the maximum error over $\mathcal{B}$. Use this new set and repeat (a) until the error function is nearly equiripple over $\mathcal{B}$. Your plot should be similar to the one in Figure 10.29. What is the resulting value of δ?

57. A highpass FIR filter is given by the specifications: $\omega_s = 0.7\pi$, $A_s = 55$ dB, $\omega_p = 0.8\pi$, and $A_s = 1$ dB. Use the Parks–McClellan algorithm to obtain a minimum length linear-phase filter. Provide a plot similar to Figure 10.12.

58. Specifications of a bandstop digital filter are given below:

$$|H(e^{j\omega})| = 1 \pm 0.01, \qquad\qquad 0 \le |\omega| \le 0.35\pi$$

$$|H(e^{j\omega})| = \pm 0.004, \qquad\qquad 0.45\pi \le |\omega| \le 0.55\pi$$

$$|H(e^{j\omega})| = 1 \pm 0.01. \qquad\qquad 0.65\pi \le |\omega| \le \pi$$

Design a linear-phase FIR filter using the Parks–McClellan algorithm. Provide a filter response plot similar to Figure 10.32.

59. Consider a multiband filter given by the specifications:
- Band-1: $0 \le \frac{\omega}{\pi} \le 0.3$, Nominal gain $= 0$, ripple $= 0.005$.
- Band-2: $0.4 \le \frac{\omega}{\pi} \le 0.7$, Nominal gain $= 0.5$, ripple $= 0.001$.
- Band-3: $0.8 \le \frac{\omega}{\pi} \le 1$, Nominal gain $= 1$, ripple $= 0.01$.

Design a minimum length linear-phase filter using the Parks–McClellan algorithm . Provide a plot similar to Figure 10.32.

60. Design a length $L = 50$ FIR differentiator using the Parks–McClellan algorithm.
(a) Graph the impulse and amplitude responses of the designed differentiator in one plot.
(b) Generate 151 samples of the signal $x[n] = 10\cos(0.2\pi n)$, $0 \le n \le 150$ and process them through the differentiator designed in (a) to obtain $y[n]$. Provide stem plots of both $x[n]$ and $y[n]$ for $50 \le n \le 100$ as sub-plots in one figure.
(c) Can you confirm that $y[n]$ corresponds to the samples of the derivative of the signal whose samples are given by $x[n]$?

Review problems

61. An analog signal $x_c(t) = 5\cos(400\pi t) + 10\sin(500\pi t)$ is to be processed by a digital signal processor in which the sampling frequency is 1 kHz.
(a) Design a minimum order FIR filter using one of the fixed windows that will pass the first component of $x_c(t)$ with attenuation of less than 1 dB but will attenuate the second component to at least 50 dB. Provide a filter response plot containing sub-plots of impulse, amplitude, log-magnitude, and error response plots.
(b) Repeat (a) using a Kaiser window design.
(c) Repeat (a) using the Parks–McClellan algorithm
(d) Compare filter orders in the above three parts.

(e) Generate 200 samples of the signal $x_c(t)$ and process through each of the designed filters to obtain output sequences. Plot the input and the output signals using linear interpolation and comment on your results.

62. We want to design a bandpass filter that has specifications on the amplitude response given by:

   • Stopband-1: $0 \le \frac{\omega}{\pi} \le 0.3$, $A_s = 50$ dB.
   • Passband: $0.35 \le \frac{\omega}{\pi} \le 0.65$, $A_p = 1$ dB.
   • Stopband-2: $0.7 \le \frac{\omega}{\pi} \le 1$, $A_s = 50$ dB.

   (a) Using a window design approach and a fixed window function, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (b) Using a window design approach and the Kaiser window function, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (c) Using a frequency-sampling design approach and with no more than two samples in the transition bands, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (d) Using the Parks–McClellan design approach, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (e) Let $x[n] = 10\cos(0.2\pi n) + \sin(0.5\pi n) + 15\cos(0.9\pi n + \pi/3)$, $0 \le n \le 200$. Process $x[n]$ through the designed filters and plot the resulting output sequences for $100 \le n \le 200$ and comment on your results.

63. The specifications on the amplitude response of an FIR filter are as follows:

   • Band-1: $0 \le \frac{\omega}{\pi} \le 0.2$, Nominal gain $= 0$, ripple $= 0.05$.
   • Band-2: $0.25 \le \frac{\omega}{\pi} \le 0.45$, Nominal gain $= 2$, ripple $= 0.1$.
   • Band-3: $0.5 \le \frac{\omega}{\pi} \le 0.7$, Nominal gain $= 0$, ripple $= 0.05$.
   • Band-4: $0.75 \le \frac{\omega}{\pi} \le 1$, Nominal gain $= 4.15$, ripple $= 0.15$.

   (a) Using a window design approach and a fixed window function, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (b) Using a window design approach and the Kaiser window function, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (c) Using a frequency-sampling design approach and with no more than two samples in the transition bands, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (d) Using the Parks–McClellan design approach, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

   (e) Compare the above four design methods in terms of the order of the filter, the exact band-edge frequencies, and the exact tolerances in each band.

64. We want to design a multiband filter that has specifications on the amplitude response given by:

- Band-1: $0 \leq \frac{\omega}{\pi} \leq 0.2$, Nominal gain $= 2$, ripple $= 0.2$.
- Band-2: $0.25 \leq \frac{\omega}{\pi} \leq 0.45$, Nominal gain $= 0$, ripple $= 0.05$.
- Band-3: $0.55 \leq \frac{\omega}{\pi} \leq 0.7$, Nominal gain $= 3$, ripple $= 0.3$.
- Band-4: $0.75 \leq \frac{\omega}{\pi} \leq 1$, Nominal gain $= 1$, ripple $= 0.1$.

(a) Using a window design approach and a fixed window function, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

(b) Using a window design approach and the Kaiser window function, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

(c) Using a frequency-sampling design approach and with no more than two samples in the transition bands, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

(d) Using the Parks–McClellan design approach, design a minimum-length linear-phase FIR filter to satisfy the given requirements. Provide a plot of the amplitude response.

(e) Compare the above four design methods in terms of the order of the filter, the exact band-edge frequencies, and the exact tolerances in each band.

11 Design of IIR filters

In this chapter we discuss the design of discrete-time IIR filters. Since these filters have infinitely long filter responses they can be designed using continuous-time filter prototypes. Thus in contrast to FIR filters, the design of IIR filters is commonly done in three steps: transform the discrete-time design specifications into continuous-time design specifications, design a continuous-time filter using well-established closed-form formulas, and convert the continuous-time filter into a discrete-time filter using a suitable mapping.

This approach is very advantageous because both continuous-time filter design formulas and continuous- to discrete-time filter mappings are available extensively in literature. Using these we can design stable lowpass filters with relative ease. However, in practice, we also need other standard frequency-selective filters such as highpass or multiband filters. Therefore, we will also study frequency-band mappings to convert lowpass into other frequency selective filters. These mappings are complex-valued and also extensively available in literature.

Study objectives

After studying this chapter you should be able to:

- Understand the zero-phase filtering operation using IIR filters.
- Design continuous-time lowpass filters using the Butterworth, Chebyshev I and II, and elliptic approximations.
- Convert continuous-time filters to discrete-time filters using the impulse-invariance and bilinear transformations.
- Convert normalized continuous-time or discrete-time lowpass filters to arbitrary lowpass, highpass, bandpass, and bandstop filters using frequency transformations.
- Understand the syntax and use of MATLAB's IIR filter design functions including the filter design and analysis tool.

11.1   Introduction to IIR filter design

The system function of a causal, stable, and realizable IIR discrete-time filter can be represented in terms of impulse response, difference-equation coefficients, or zero-pole locations (see Section 3.6) and is, respectively, given by the formulas

$$H(z) = \sum_{n=0}^{\infty} h[n]z^{-n} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} = b_0 \frac{\prod_{k=1}^{M}(z - z_k)}{\prod_{k=1}^{N}(z - p_k)}. \qquad (11.1)$$

Furthermore, causality requires that $h[n] = 0$ for $n < 0$; the system is causal and stable if all poles are inside the unit circle, that is, $|p_k| < 1$ for $k = 1, 2, \ldots, N$; and the zeros can be everywhere.

The objective in IIR filter design is to determine the coefficients of (11.1) so that its frequency response $H(e^{j\omega})$ approximates an ideal desired response $H_d(e^{j\omega})$ according to some criterion of performance. Design of IIR filters by minimizing the mean square error or the Chebyshev error requires the solution of difficult optimization problems, due to the nonlinear dependence of $H(e^{j\omega})$ on the filter coefficients; see Antoniou (2006), Rabiner and Gold (1975), or Steiglitz *et al.* (1992). These techniques are used to design filters with arbitrary frequency responses, that is, responses different from those of the standard frequency selective filters.

The techniques for the design of standard frequency selective discrete-time IIR filters are based on well-developed continuous-time filter design methods. In Section 5.11, we stated that causal, stable, and realizable continuous-time filters have rational system functions:

$$H_c(s) = \int_{0}^{\infty} h_c(t)e^{-st}dt = \frac{\sum_{k=0}^{M} \beta_k s^k}{1 + \sum_{k=1}^{N} \alpha_k s^k} = \beta_0 \frac{\prod_{k=1}^{M}(s - \zeta_k)}{\prod_{k=1}^{N}(s - s_k)}. \qquad (11.2)$$

In this case, causality requires that $h_c(t) = 0$ for $t < 0$; the system is causal and stable if all poles are on the left-half plane, that is, $\mathcal{R}e(s_k) < 0$ for $k = 1, 2, \ldots, N$; and the zeros can be everywhere. To ensure that the response $|H(j\Omega)| \to 0$ as $\Omega \to \infty$, we require that $M < N$. Due to the similarities between the system functions $H_c(s)$ and $H(z)$, the most popular techniques for designing IIR filters are, in some way, discrete-time versions of continuous-time filter design methods. Such design techniques include the following steps (see Figure 11.1):

Step 1 Convert the discrete-time design specifications into continuous-time specifications. This step depends on the the transformation used in Step 3.
Step 2 Design a continuous-time filter, that is, obtain a system function $H_c(s)$ that satisfies the continuous-time specifications. An introduction to continuous-time lowpass filter design techniques is given in Section 11.2.
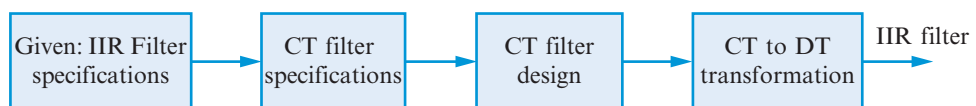
Figure 11.1 Procedure for designing IIR filters from continuous-time filters.

Approach 1
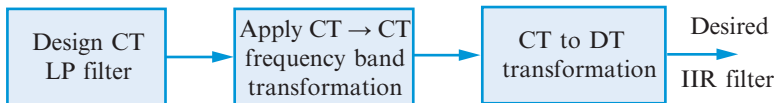
Approach 2

Figure 11.2 Frequency transformations for lowpass filters.

Step 3 Convert $H_c(s)$ to an appropriate system function $H(z)$, which meets the spec-
ifications, using a continuous-time to discrete-time transformation. Such mapping
procedures are the subject of Section 11.3.

Examples of IIR lowpass filter design using the approach described in Figure 11.1 are
provided in Section 11.6. This approach is most useful for designing standard filters such
as lowpass, bandpass, highpass, and bandreject filters, for which a considerable body of
continuous-time design techniques based on closed-form design formulas is available.

The mathematical approximation theory for continuous-time lowpass filters has been
highly developed and has produced simple closed-form design formulas for various
types of filter. Given a continuous-time lowpass filter prototype, there are two dis-
tinct approaches to design IIR filters of bandpass, highpass, and bandstop types (see
Figure 11.2). The two procedures differ in that approach-1 does the frequency band trans-
formation in continuous-time, whereas approach-2 does the frequency band transformation
in discrete-time. These approaches to frequency band transformations are discussed in
Section 11.5.

As we stated in Section 10.2, causal IIR filters *cannot* have linear phase. Therefore, the
approximation problem for IIR filters involves both the magnitude and phase responses,
that is, a complex system function $H(e^{j\omega})$. In FIR filter design we avoided this problem
by showing that a magnitude response with linear phase can be expressed by an equivalent
real-valued amplitude response function; this is *not* possible for IIR filters. Since the phase
response of IIR filters is generally highly nonlinear, we should always examine the group-
delay response to see how much frequency dispersal we have within the passband.

FIR versus IIR filters Complexity in the approximation problem or phase linearity are
not the only differences between FIR filters and IIR filters in terms of design and imple-
mentation. FIR filters have both advantages and disadvantages compared to IIR filters as
explained below.

Advantages: FIR filters can have exactly linear phase, are always stable, have design
methods that are generally linear in filter parameters, can have great flexibility in
choosing their frequency response, can be realized efficiently in hardware, and have
finite-duration transients (or start-up responses).
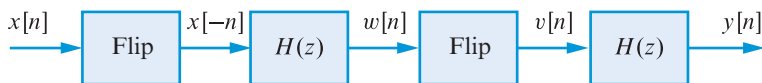
Figure 11.3 Implementation of a zero-phase IIR filtering procedure.

Disadvantages: FIR filters often require a much higher filter order than IIR filters to achieve a given level of performance, the delay in the output response is often much greater than for an equal performance IIR filter, and the design methods often are iterative in nature requiring computer-aided techniques.

Even though FIR filters enjoy many advantages, for most applications, IIR filters are desirable due to their lower order and hence lower cost compared to FIR filters, but if linear-phase response is of paramount interest then FIR filters are preferable.

Zero-phase IIR filtering Before we conclude this section, we mention that it is possible to achieve a linear-phase response with an IIR filter, if we remove the causality constraint. We recall from Section 5.8 that, if the impulse response $h[n]$ in (11.1) is real, we have

$$g[n] \triangleq h[n] * h[-n] \xrightarrow{\text{DTFT}} G(e^{j\omega}) = |H(e^{j\omega})|^2 = H(z)H(1/z)|_{z=e^{j\omega}}. \qquad (11.3)$$

Since $h[n]$ is causal, the filter $h[-n]$ is anticausal with system function $H(1/z)$. Clearly, the filter $g[n] = h[n] * h[-n]$ is noncausal and has zero-phase response because $G(e^{j\omega})$ is always nonnegative. The response to an input $x[n]$ is given by

$$Y(z) = H(z)H(1/z)X(z). \qquad (11.4)$$

Assuming that a time-reversal (flip) system is defined by $f[n] = x[-n]$ then, using z-transform properties, we have $F(z) = X(1/z)$. Let $V(z) \triangleq H(1/z)X(z)$, then we have $V(1/z) = H(z)X(1/z) = H(z)F(z) \triangleq W(z)$. Thus the system (11.4) can be implemented as shown in Figure 11.3. This system cannot be realized in real-time because the time-reversal operation is noncausal. A practical version of this approach is provided by MATLAB function `y=filtfilt(b,a,x)` (see Tutorial Problem 1); another approach is discussed in Tutorial Problem 20. In real-time systems, we can use an allpass equalizer to "linearize" the phase response of IIR filters (see Section 5.9).

11.2 Design of continuous-time lowpass filters

We now consider the approximation problem for continuous-time filters. Although we limit our attention to the most widely used techniques, these techniques are optimum according to the maximally flat criterion or the equiripple criterion in each band.

Before we proceed to discuss the various design techniques, let us consider some basic properties of magnitude-squared functions. For a continuous-time filter with real coefficients we have (see Section 5.11.5)

$$|H_c(j\Omega)|^2 = H_c(s)H_c(-s)|_{s=j\Omega}, \qquad (11.5)$$

which can be written in terms of poles and zeros using (11.2). A typical pair of factors, like $(s - s_k)(-s - s_k) = s_k^2 - s^2$, evaluated at $s = j\Omega$ becomes $(s_k^2 + \Omega^2)$. Hence, the magnitude-squared function can always be written as

$$|H_c(j\Omega)|^2 = G^2 \, \frac{(\Omega^2 + \zeta_1^2)(\Omega^2 + \zeta_2^2) \cdots (\Omega^2 + \zeta_M^2)}{(\Omega^2 + s_1^2)(\Omega^2 + s_2^2) \cdots (\Omega^2 + s_N^2)}. \tag{11.6}$$

Since the coefficients of $H_c(s)$ are real, its poles and zeros are either real or they appear in complex conjugate pairs. A term like $(\Omega^2 + s_k^2)$ is real when s_k is real; if $s_1 = re^{j\theta}$ and $s_2 = re^{-j\theta}$ are two complex conjugate poles, we have $(\Omega^2 + s_1^2)(\Omega^2 + s_2^2) = (\Omega^2 - r^2)^2 \geq 0$ for all Ω. Thus, $|H_c(j\Omega)|^2$ is a positive and real rational function of Ω^2. Design techniques for continuous-time filters use $|H_c(j\Omega)|^2$ because it is real, differentiable, and a rational function of Ω^2; the function $|H_c(j\Omega)|$ is real, but it lacks the other two properties. Because of the causality and stability requirements we can specify either the magnitude response or the phase response, but *not* both.

Analog lowpass filter specifications The design techniques discussed in this section approximate the magnitude-squared response of an ideal lowpass filter with cutoff frequency Ω_c, which is defined by

$$|H_d(j\Omega)|^2 = \begin{cases} 1, & 0 \leq |\Omega| \leq \Omega_c \\ 0. & |\Omega| > \Omega_c \end{cases} \tag{11.7}$$

As discussed in Section 10.1.1 a practical lowpass filter has a transition band around the cutoff frequency Ω_c creating a passband and a stopband. It also exhibits ripples around unity in the passband and around zero in the stopband as shown in Figure 10.1. A similar tolerance diagram that is more appropriate for analog filters is shown in Figure 11.4 in which Ω_p is the passband edge, Ω_s is the stopband edge. The magnitude-squared specifications are given in terms of passband ripple A_p (in dB) or ϵ and stopband attenuation A_s (in dB) or A. Conversion formulas between these quantities were given in (10.8) and (10.9).

System function from magnitude-squared response Referring to (11.7), the approximating function must approximate a constant in each of two ranges: unity in the range $|\Omega| < \Omega_c$ and zero for $|\Omega| > \Omega_c$. The classical approximation techniques use a function of the form

$$|H_c(j\Omega)|^2 = \frac{1}{1 + V^2(\Omega)}, \tag{11.8}$$

where $V^2(\Omega) \ll 1$ for $|\Omega| \leq \Omega_c$ and $V^2(\Omega) \gg 1$ for $|\Omega| > \Omega_c$. Different choices for $V(\Omega)$ lead to different design techniques. However, all these techniques produce a function $|H_c(j\Omega)|^2$ like (11.6). The problem is now reduced to obtaining a causal and stable system $H_c(s)$ from the magnitude-squared function (11.5). This requires solution of a spectral factorization problem (see Section 5.11.5). Since $H_c(s)$ has real coefficients, the poles and zeros of $H_c(-s)H_c(s)$ are symmetrically located with respect to both the real and the imaginary axes (quadrantal symmetry). Therefore, we can obtain a causal and stable
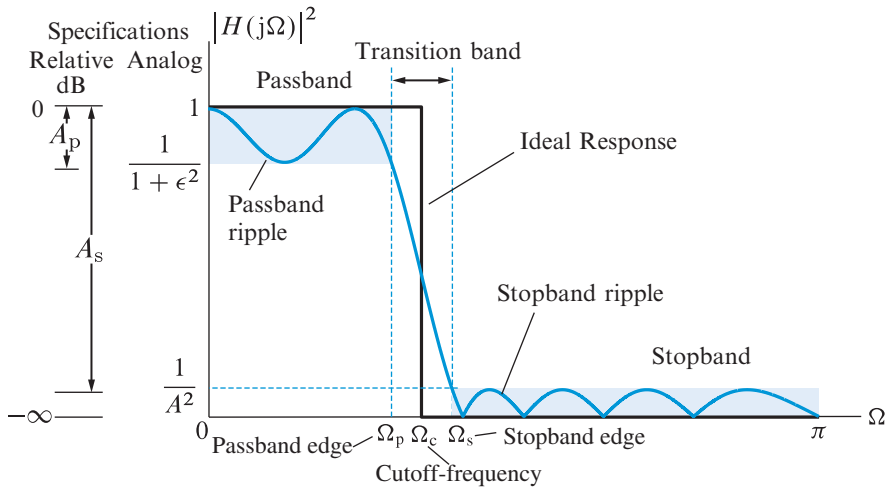
Figure 11.4 Magnitude-squared specifications for lowpass analog filter.

system by choosing the poles on the left-half plane; the zeros can be anywhere. However, we typically choose the zeros on the left-half plane, which results in a minimum-phase system (see Section 5.11.5). This approach is used by all filter design techniques discussed in this section.

11.2.1   The Butterworth approximation

In this section we develop properties of the Butterworth filter, which is based on a Taylor series approximation of the ideal response at a single frequency. The resulting magnitude response is optimum according to the maximally flat criterion.

Definition and properties Butterworth (1930) suggested that $V(\Omega) = (\Omega/\Omega_c)^{2N}$ be used in (11.8) as an approximation. Thus, we obtain the Butterworth magnitude-squared response

$$|H_B(j\Omega)|^2 \triangleq \frac{1}{1 + (\Omega/\Omega_c)^{2N}}. \quad N = 1, 2, \ldots \tag{11.9}$$

From (11.9) we see that for every value of N we have

$$|H_B(j0)|^2 = 1, \quad |H_B(j\Omega_c)|^2 = 1/2, \quad \text{and} \quad |H_B(j\infty)|^2 = 0. \tag{11.10}$$

This implies that the gain at $\Omega = 0$ (dc) is 1 and the 3 dB cutoff frequency at Ω_c. Figure 11.5 shows the Butterworth approximation function for various values of N. We note that the characteristic is monotonically decreasing in both the passband and stopband. Hence, $|H_B(j\Omega)|$ has its maximum value at $\Omega = 0$. Observe that as $N \to \infty$, the Butterworth magnitude function approaches the ideal lowpass filter characteristics. Hence, we can choose the parameter N to satisfy prescribed filter design specifications.
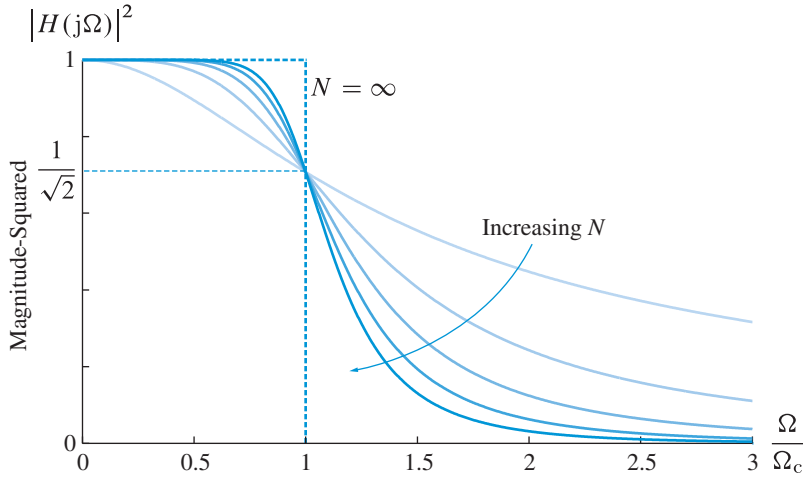
Figure 11.5 The magnitude of Butterworth functions of various orders.

The Taylor series expansion of (11.9) about $\Omega = 0$ can be found from the series $1/(1+x) = 1 - x + x^2 - x^3 + \cdots$, where $|x| < 1$, by letting $x = (\Omega/\Omega_c)^2$. Therefore, the error in the passband is given by

$$E_c^2(j\Omega) = 1 - |H_B(j\Omega)|^2 = (\Omega/\Omega_c)^{2N} - (\Omega/\Omega_c)^{4N} + \cdots \qquad (11.11)$$

Since the Taylor series for the error starts with the Ω^{2N} power, this means the first $(2N-1)$ derivatives with respect to Ω are 0 at $\Omega = 0$. Thus, Butterworth filters are also called maximally flat magnitude filters (see Section 10.1.3).

For frequencies $|\Omega| \gg \Omega_c$ we have the asymptotic approximation

$$|H_B(j\Omega)|^2 = \frac{1}{1 + (\Omega/\Omega_c)^{2N}} \simeq \frac{1}{(\Omega/\Omega_c)^{2N}}. \qquad (11.12)$$

The high-frequency roll-off is equal to $-10\log(\Omega/\Omega_c)^{2N}$, that is, about $6N$ dB per octave.

Pole locations The poles of $H_B(s)H_B(-s)$ are found by solving the equation

$$1 + (s/j\Omega_c)^{2N} = 0 \quad \text{or} \quad (s/j\Omega_c)^{2N} = -1 = e^{j(2k-1)\pi}, \qquad (11.13)$$

where $k = 1, 2, \ldots, 2N$ so that the poles are counted in the anticlockwise direction starting at the first pole after the positive imaginary axis. The solution of (11.13) yields the poles $s_k = \sigma_k + j\Omega_k$ for any even or odd value of N. Thus, the poles of a Butterworth filter are given by

$$\sigma_k = \Omega_c \cos\theta_k, \qquad (11.14a)$$

$$\Omega_k = \Omega_c \sin\theta_k, \qquad (11.14b)$$

where

$$\theta_k \triangleq \frac{\pi}{2} + \frac{2k-1}{2N}\pi. \qquad k = 1, 2, \ldots, 2N \qquad (11.15)$$
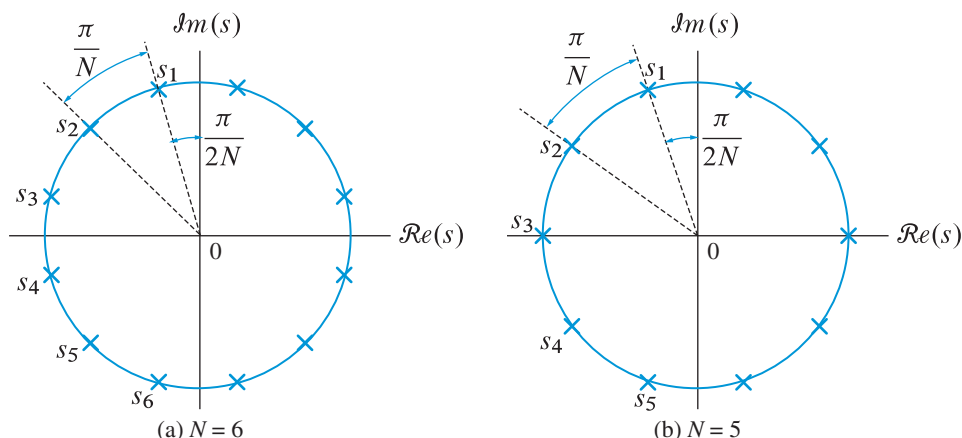
Figure 11.6 Pole locations of Butterworth magnitude-squared function in the s-plane for:
(a) even N, and (b) odd N. Poles on the left-half plane correspond to a stable system.

Using (11.14) we note that

$$|s_k|^2 = \sigma_k^2 + \Omega_k^2 = \Omega_c^2 \cos^2 \theta_k + \Omega_c^2 \sin^2 \theta_k = \Omega_c^2. \tag{11.16}$$

In view of (11.15) and (11.16), the poles of a Butterworth filter lie on a circle with radius
Ω_c and are equiangularly spaced with angular separation π/N (see Figure 11.6). If s_k is a
real pole, then $\theta_k = \pi$ and $s_k = -\Omega_c$. From (11.15) this can occur only when $N = 2k - 1$,
that is when N is an odd number. Poles never fall on the imaginary axis. We form the
stable system function $H_B(s)$ by choosing poles for $k = 1, 2, \ldots, N$, which clearly lie in
the left-half plane, that is,

$$H_B(s) = \frac{\Omega_c^N}{(s - s_1)(s - s_2) \ldots (s - s_N)}. \tag{11.17}$$

Conventionally, the numerator is set equal to Ω_c^N to assure that $|H_B(j0)| = 1$.

Design procedure Suppose we wish to design a Butterworth lowpass filter specified by
the parameters Ω_p, A_p, Ω_s, and A_s (see Figure 11.4). The design process consists of
determining the parameters N and Ω_c in (11.17) so that

$$\frac{1}{1 + (\Omega_p/\Omega_c)^{2N}} \geq \frac{1}{1 + \epsilon^2} \quad \text{or} \quad (\Omega_p/\Omega_c)^{2N} \leq \epsilon^2, \tag{11.18a}$$

$$\frac{1}{1 + (\Omega_s/\Omega_c)^{2N}} \leq \frac{1}{A^2} \quad \text{or} \quad (\Omega_s/\Omega_c)^{2N} \geq A^2 - 1. \tag{11.18b}$$

Solving the first inequality and substituting in the second one, yields

$$\Omega_s^N \geq \Omega_c^N \sqrt{A^2 - 1} \geq \Omega_p^N \frac{\sqrt{A^2 - 1}}{\epsilon}. \tag{11.19}$$

Solving (11.19) for the filter order N we obtain the following design equation

$$N \geq \frac{\ln \beta}{\ln \alpha}, \tag{11.20}$$

where

$$\alpha \triangleq \frac{\Omega_s}{\Omega_p}, \quad \beta \triangleq \frac{1}{\epsilon}\sqrt{A^2 - 1} = \frac{\sqrt{10^{A_s/10} - 1}}{\sqrt{10^{A_p/10} - 1}}. \tag{11.21}$$

The value of N is chosen as the largest integer satisfying (11.20); hence, the design specifications at Ω_p, Ω_s, or both, may be exceeded. The frequency Ω_c can be chosen anywhere in the interval

$$\Omega_p(10^{A_p/10} - 1)^{-1/(2N)} \leq \Omega_c \leq \Omega_s(10^{A_s/10} - 1)^{-1/(2N)}. \tag{11.22}$$

The left limit satisfies exactly the specifications at Ω_p; the right limit satisfies exactly the specifications at Ω_s. To ensure a smaller ripple in the passband, we choose Ω_c using the right limit. After N and Ω_c are chosen, we use (11.14), (11.15), and (11.17) to obtain the system function $H_B(s)$. This procedure is explained in Example 11.1.

MATLAB functions for analog Butterworth lowpass filters The SP toolbox provides two functions to design analog Butterworth lowpass filters. The function
        [N, Omegac] = buttord(Omegap, Omegas, Ap, As, 's')
computes the order N and cutoff frequency Ω_c given the design parameters. The filter design and computation of its system function is then obtained using the
        [C,D] = butter(N, Omegac, 's')
function, which provides the numerator and denominator polynomials in arrays C and D, respectively.

Example 11.1   Design procedure – Butterworth approximation

We describe the Butterworth approximation design procedure to obtain an analog lowpass filter that satisfies

$$-6 \text{ dB} \leq 20 \log_{10} |H(j\Omega)| \leq 0, \quad 0 \leq |\Omega| \leq 2\tfrac{\text{rad}}{\text{sec}}, \tag{11.23a}$$

$$20 \log_{10} |H(j\Omega)| \leq -20 \text{ dB}, \quad 3\tfrac{\text{rad}}{\text{sec}} \leq |\Omega| < \infty. \tag{11.23b}$$

Thus $A_p = 6$ and $A_s = 20$. From (10.9), the analog magnitude specifications are

$$\epsilon = \sqrt{10^{0.1(6)}} = 1.7266 \quad \text{and} \quad A = 10^{0.05(20)} = 10.$$

We illustrate the design procedure using the following steps:

Step-1 Compute the parameters α and β using (11.21):

$$\alpha = \frac{3}{2} = 1.5; \quad \beta = \frac{1}{1.7266}\sqrt{10^2 - 1} = 5.7628.$$

Step-2 Compute order N using (11.20):

$$N = \left\lceil \frac{\ln(5.7628)}{\ln(1.5)} \right\rceil = \lceil 4.3195 \rceil = 5.$$

Step-3 Determine 3 dB cutoff frequency Ω_c. Using (11.22), the lower and upper values of Ω_c are

$$2(10^{6/10} - 1)^{-1/(10)} = 1.7931 \text{ and } 3(10^{20/10} - 1)^{-1/(10)} = 1.8948,$$

respectively. We choose the upper value $\Omega_c = 1.8948$ rad/s, which satisfies the specifications at Ω_s and provides a smaller ripple at Ω_p.

Step-4 Compute pole locations. From (11.14) and (11.15), the poles of $H_B(j\Omega)$ are located on a circle of radius $\Omega_c = 1.8948$ at angles

$$\theta_k = \frac{\pi}{2} + \frac{2k-1}{10}\pi = 0.4\pi + 0.2k\pi, \quad k = 1, 2, 3, 4, 5$$

with poles given by

$$s_k = 1.8948\cos(0.4\pi + 0.2k\pi) + j1.8948\sin(0.4\pi + 0.2k\pi), \ k = 1, \ldots, 5.$$

Step-5 Compute the system function $H_B(j\Omega)$ using (11.17):

$$H_B(j\Omega) = \frac{1.8948^5}{\Pi_{k=1}^{5}(s - s_k)}$$

$$= \frac{24.42}{s^5 + 6.13s^4 + 18.80s^3 + 35.61s^2 + 41.71s + 24.42}. \tag{11.24}$$

The required filter can also be obtained using the following MATLAB script:

```
>> [N, Omegac] = buttord(2, 3, 6, 20, 's');
N =
     5
Omegac =
    1.8948
>> [C,D] = butter(N,Wn,'s')
C =
         0         0         0         0         0    24.4224
D =
    1.0000    6.1316   18.7984   35.6187   41.7108   24.4224
```

which agrees with the result obtained in (11.24). ∎

Example 11.2   Butterworth filter design

We want to design an analog lowpass filter using the Butterworth approximation that satisfies the specifications:

$$\text{passband edge: } F_p = 40 \text{ Hz}, \quad \text{passband ripple: } A_p = 1 \text{ dB}, \quad (11.25a)$$

$$\text{stopband edge: } F_s = 50 \text{ Hz}, \quad \text{stopband attenuation: } A_s = 30 \text{ dB}. \quad (11.25b)$$

Using the MATLAB script

```
>> [N,Omegac] = buttord(2*pi*40,2*pi*50,1,30,'s'); N
N =
    19
>> Fc = Omegac/(2*pi)
Fc =
   41.6902
>> [C,D] = butter(N,Omegac,'s');
```

we obtain a 19th-order filter that has the 3 dB cutoff frequency of $F_c = \Omega_c/(2\pi) = 41.7$ Hz. Figure 11.7 shows various response plots of the designed filter. In the magnitude response plot the magnitude at 41.7 Hz is down to 3 dB $\equiv 1/\sqrt{2}$ while in the log magnitude plot the response at $F_s = 50$ Hz is exactly 30 dB. The group-delay response shows a nonlinear but smooth function. The pole location plot shows that poles in the left-half of the s-plane are equiangularly distributed along a circle of radius $\Omega_c = 262$ with one pole on the real axis since N is odd. ◼

We note that Butterworth filters have the maximum flatness, that is, they are very close to the ideal response, at $\Omega = 0$ and $\Omega = \infty$. The maximum error occurs at $\Omega = \Omega_c$ and the response decreases monotonically from $\Omega = 0$ to $\Omega = \infty$. Although, the phase response was not part of the approximation problem, we note that Butterworth filters have a very smooth group-delay response.

11.2.2   The Chebyshev approximation

The Chebyshev approximation is optimum according to the minimax criterion which results in equiripple behavior. In the case of FIR filters, the optimum filter was determined using the Remez exchange algorithm. For continuous-time filters, we can attain optimum equiripple approximation in the passband, stopband, or both bands without carrying out an explicit minimization procedure. The existence of a very elegant and powerful theory enables the derivation of closed form design formulas for optimum Chebyshev filters with equiripple behavior in the passband (Chebyshev I filters), stopband (inverse Chebyshev or Chebyshev II filters), or both passband and stopband (elliptic or Cauer filters).
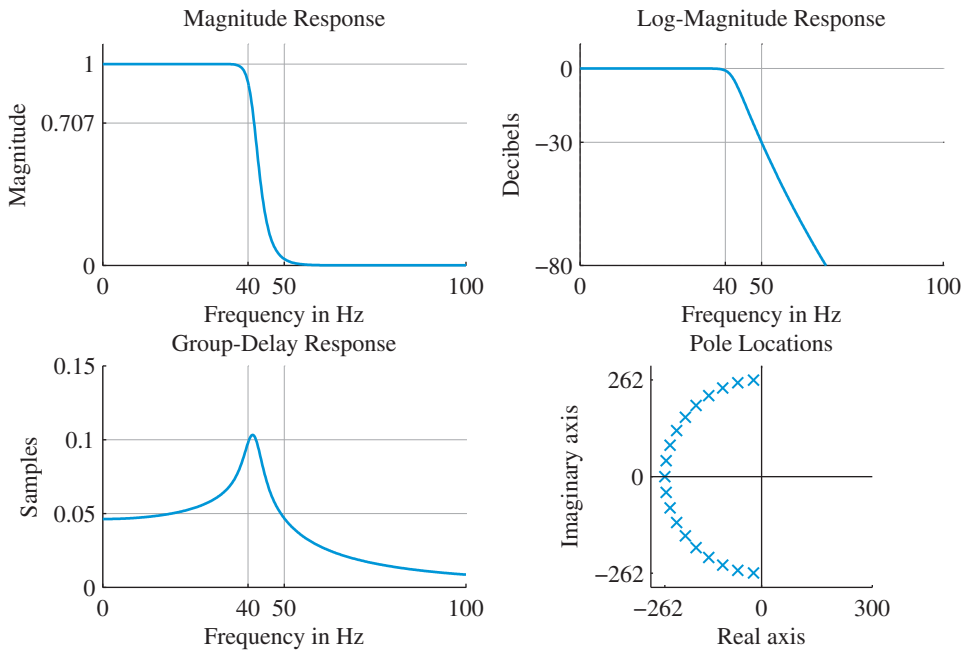
Figure 11.7 Design plots for the 19th-order lowpass Butterworth filter in Example 11.2.

Definition and properties We start with the Chebyshev I or simply Chebyshev lowpass filter approximation

$$|H_C(j\Omega)|^2 = \frac{1}{1 + \epsilon^2 T_N^2(\Omega/\Omega_c)}, \tag{11.26}$$

where $T_N(x)$, $x = \Omega/\Omega_c$, is the Nth order Chebyshev polynomial discussed in Section 10.5. Since $|T_N(x)| \leq 1$ for $|x| \leq 1$ we have $|T_N(\Omega/\Omega_c)| \leq 1$ for $|\Omega| \leq \Omega_c$. If we choose $\epsilon^2 \ll 1$, the approximation error in the passband is given by

$$E_C^2(\Omega/\Omega_c) = 1 - \frac{1}{1 + \epsilon^2 T_N^2(\Omega/\Omega_c)} \simeq \epsilon^2 T_N^2(\Omega/\Omega_c). \quad |\Omega| \leq \Omega_c \tag{11.27}$$

Since we can express the weighted error $(1/\epsilon)E_C(\Omega/\Omega_c)$ as a single Chebyshev polynomial $T_N(\Omega/\Omega_c)$, we conclude that (11.26) provides the optimum equiripple lowpass filter approximation within the entire passband (see Section 10.5.2).

Since the leading term of $T_N(x)$ is $2^{N-1}x^N$, the values of $T_N^2(x)$ grow very fast for $|x| > 1$. Thus, in the stopband we have $T_N^2(\Omega/\Omega_c) \gg 1$ or equivalently $|H_C(j\Omega)|^2 \ll 1$, for $|\Omega| > \Omega_c$. The formula used for $T_N(x)$ in the passband

$$T_N(x) = \cos(N \cos^{-1} x), \quad |x| \leq 1 \tag{11.28}$$

is not valid for $|x| > 1$. To develop the theory of Chebyshev filters, we need a similar expression for the stopband. This should be possible because the polynomials listed in
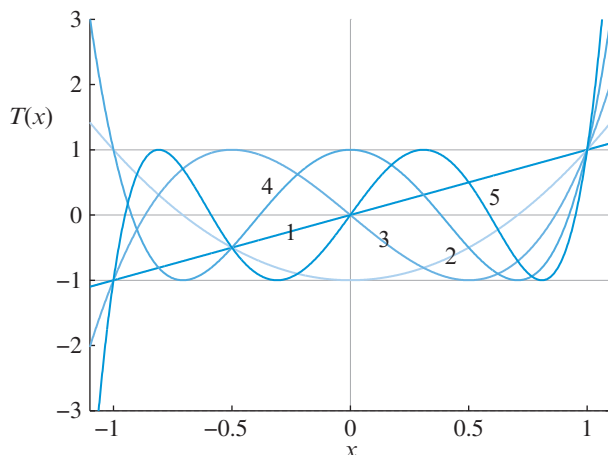
Figure 11.8 Graphs of Chebyshev polynomials $T_N(x)$ for $N = 1, 2, \ldots, 5$.

Table 10.4 can be evaluated for all values of x, real or complex. It turns out (see Tutorial Problem 4), that this can be done by replacing the trigonometric functions in (10.100) by their hyperbolic counterparts (see Zill and Shanahan (2009)):

$$\cosh(x) \triangleq \frac{1}{2}\left(e^x + e^{-x}\right), \quad \cosh^{-1}(x) = \ln\left(x + \sqrt{x^2 - 1}\right), \tag{11.29a}$$

$$\sinh(x) \triangleq \frac{1}{2}\left(e^x - e^{-x}\right), \quad \sinh^{-1}(x) = \ln\left(x + \sqrt{x^2 + 1}\right). \tag{11.29b}$$

We first note that $x = \cos(j\phi) = (e^{j(j\phi)} + e^{-j(j\phi)})/2 = \cosh\phi$ or $\phi = \cosh^{-1} x$. Hence, we have $\cos(N\phi) = \cos(Nj\phi) = \cosh(N\phi)$. This leads to the formula

$$T_N(x) = \cosh(N\cosh^{-1} x). \quad |x| > 1 \tag{11.30}$$

Chebyshev polynomials can be computed in MATLAB by `T1=cos(N*acos(x))` for $|x| \le 1$ and `T2=cosh(N*acosh(x))` for $|x| \ge 1$. Figure 11.8 shows a plot of the first few Chebyshev polynomials; because the values grow very fast (exponentially), we include only a small segment of $T_N(x)$ outside the range $-1 \le x \le 1$.

Figure 11.9 shows the magnitude response of the Chebyshev I approximation (11.26) for $N = 6$, $N = 7$, and $\epsilon = 0.75$. Based on (11.26) and the properties of Chebyshev polynomials, an Nth-order prototype lowpass Chebyshev I filter has the following basic properties:

1. For $|\Omega| \le \Omega_c$, $|H_C(j\Omega)|^2$ has equiripple behavior between $\frac{1}{1+\epsilon^2}$ and 1.
2. For $|\Omega| \ge \Omega_c$, $|H_C(j\Omega)|^2$ decreases monotonically toward zero. The high-frequency roll-off is $20N$ dB/decade.
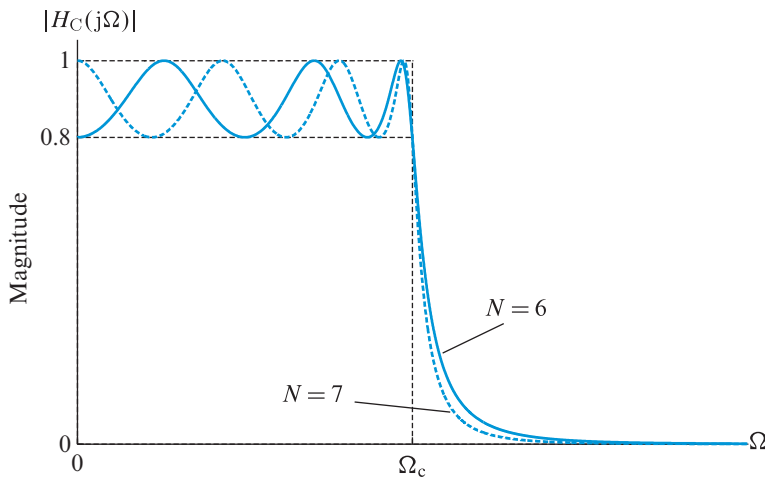
Figure 11.9 Magnitude responses of two Chebyshev I lowpass filters with ripple parameter $\epsilon = 0.75$ and orders $N = 6$ (solid-line) and $N = 7$ (dashed-line).

3. From the definition of Chebyshev polynomials we have

$$|H_C(j0)|^2 = \begin{cases} 1, & N \text{ odd} \\ 1/(1 + \epsilon^2), & N \text{ even} \end{cases} \tag{11.31a}$$

$$|H_C(j\Omega_c)|^2 = \frac{1}{1 + \epsilon^2}. \tag{11.31b}$$

For $|\Omega| \gg \Omega_c$ we have the approximation (using the leading term of T_N)

$$|H_C(j\Omega)|^2 \simeq \left[\epsilon^2 2^{2(N-1)} (\Omega/\Omega_c)^{2N} \right]^{-1}. \tag{11.32}$$

We first note that the high-frequency roll-off is about $6N$ dB per octave, as in the Butterworth filter. If we set $\epsilon = 1$, we conclude from (11.26), that the Chebyshev and Butterworth filters have the same 3 dB cutoff frequency. Since Ω_c plays the same role in both filters, we can compare their performance. Comparing (11.32) and (11.12), we see that the Chebyshev filter has $10 \log 2^{2(N-1)}$ or about $6(N-1)$ dB greater attenuation than the Butterworth filter.

Pole locations The poles of the product $H_C(s)H_C(-s)$ are obtained by solving the equation

$$T_N(s/j\Omega_c) = \pm j/\epsilon. \tag{11.33}$$

Using the trigonometric form of Chebyshev polynomials, we may write

$$T_N(s/j\Omega_c) = \cos[N \cos^{-1}(s/j\Omega_c)] = \pm j/\epsilon. \tag{11.34}$$

To solve this equation, we first define a complex function as

$$w \triangleq u + jv = \cos^{-1}(s/j\Omega_c). \tag{11.35}$$

Substituting (11.35) in (11.34), we obtain

$$\cos[N(u + jv)] = \cos(Nu)\cosh(Nv) - j\sin(Nu)\sinh(Nv) = \pm j/\epsilon, \tag{11.36}$$

because $\sinh(x) = -j\sin(jx)$. Equating the real parts of the second and third members of this relation gives $\cos(Nu)\cosh(Nv) = 0$. Since $\cosh(Nv) \geq 1$ for all values of Nv, this equality requires $\cos(Nu) = 0$. This may be expressed as

$$u_k = \frac{2k-1}{N}\frac{\pi}{2}, \quad k = 1, 2, \ldots, 2N. \tag{11.37}$$

Equating the imaginary parts of (11.36) and recognizing that for all values of u defined by (11.37), $\sin(Nu) = \pm 1$, we obtain

$$v = -\frac{1}{N}\sinh^{-1}\frac{1}{\epsilon} \triangleq -\phi. \tag{11.38}$$

The poles of $H_C(s)H_C(-s)$ can now be found by putting (11.34) in the form

$$s_k = j\Omega_c \cos(u_k + jv) = \Omega_c \sin(u_k)\sinh(v) + j\Omega_c \cos(u_k)\cosh(v). \tag{11.39}$$

Thus, the poles $s_k = \sigma_k + j\Omega_k$ are given by

$$\sigma_k = -[\Omega_c \sinh(\phi)]\sin u_k, \quad \Omega_k = [\Omega_c \cosh(\phi)]\cos u_k. \tag{11.40}$$

Using the identities $\cos(\theta - \pi/2) = \sin(\theta)$ and $\sin(\theta - \pi/2) = -\cos(\theta)$, we can express the poles in a form similar to that used for Butterworth filters

$$\sigma_k = [\Omega_c \sinh(\phi)]\cos(\theta_k), \tag{11.41a}$$

$$\Omega_k = [\Omega_c \cosh(\phi)]\sin(\theta_k), \tag{11.41b}$$

where the angle θ_k is given by

$$\theta_k = \frac{\pi}{2} + \frac{2k-1}{2N}\pi, \quad k = 1, 2, \ldots, 2N, \tag{11.42}$$

such that the first angle is assigned to the first pole in the negative left-half of the s-plane after the positive imaginary axis. To obtain a stable system, we assign to $H_C(s)$ the poles located on the left-half plane ($\sigma_k < 0$), as we did for Butterworth filters. Using the assignment in (11.42), the system function $H_C(s)$ is given by

$$H_C(s) = \frac{G}{\prod_{k=1}^{N}(s - s_k)}, \quad G = \prod_{k=1}^{N}(-s_k) \times \begin{cases} 1/\sqrt{1+\epsilon^2}, & N \text{ even} \\ 1, & N \text{ odd} \end{cases} \tag{11.43}$$

where G is selected to satisfy the normalization condition (11.31a).
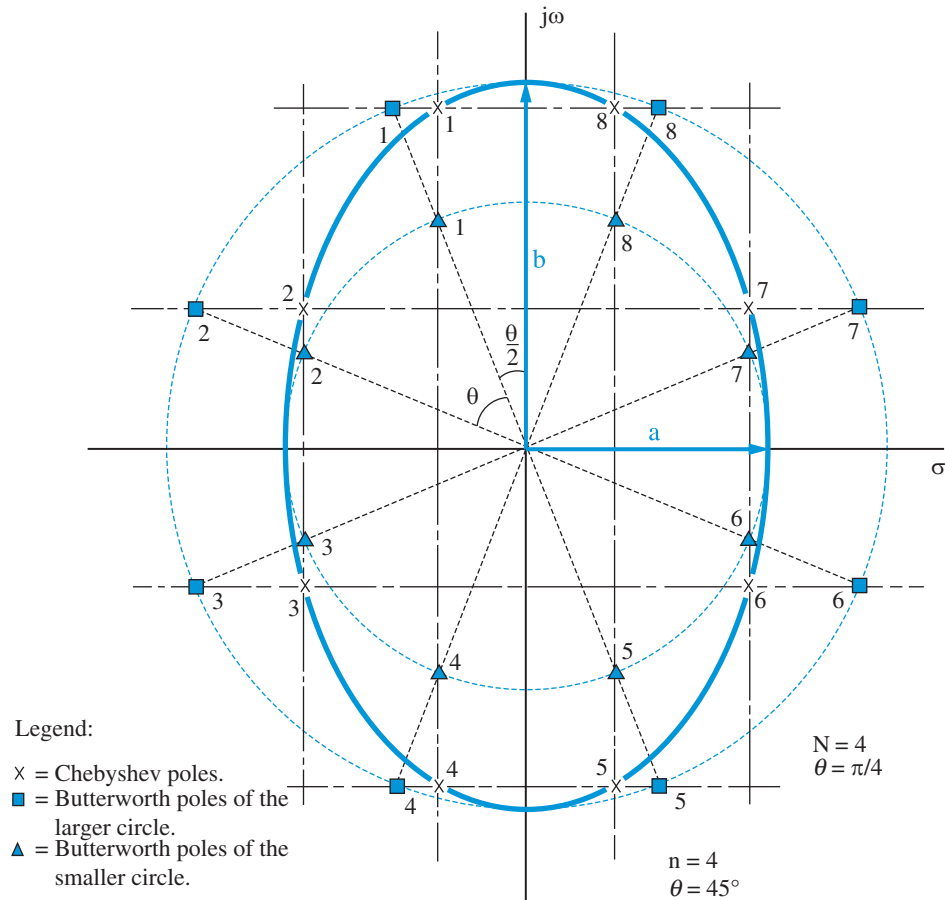
Figure 11.10 Geometrical construction of Chebyshev filter poles.

The expressions (11.41) for the poles of the Chebyshev filter can be simplified by using (11.38). Indeed, after some algebraic manipulations we obtain

$$a \triangleq \sinh(\phi) = \frac{1}{2}(\gamma - \gamma^{-1}), \tag{11.44a}$$

$$b \triangleq \cosh(\phi) = \frac{1}{2}(\gamma + \gamma^{-1}), \tag{11.44b}$$

where

$$\gamma \triangleq \left(1/\epsilon + \sqrt{1 + 1/\epsilon^2}\right)^{1/N}. \tag{11.45}$$

Figure 11.10 illustrates the geometrical construction of Chebyshev poles using a method developed by Guillemin (1957). Using the identity $\sin^2\theta_k + \cos^2\theta_k = 1$ and (11.41) we obtain

$$\left(\frac{\sigma_k}{\Omega_c a}\right)^2 + \left(\frac{\Omega_k}{\Omega_c b}\right)^2 = 1. \tag{11.46}$$

If we drop the index k and let σ and Ω have any values, we note that (11.46) is the equation for an ellipse with major semi-axis $\Omega_c b$ and minor semi-axis $\Omega_c a$. Since $b > a$, the major axis of the ellipse lies along the $j\Omega$ axis. Thus, the poles of $H_C(s)H_C(-s)$ are distributed on an ellipse in the s-plane. The poles on the ellipse may be geometrically related to the poles of two Butterworth circles with radii a and b. Comparison of (11.14) and (11.41) reveals that the poles of an Nth order Chebyshev filter are related to the poles of two Nth order Butterworth filters with $\Omega_{c_1} = a\Omega_c$ and $\Omega_{c_2} = b\Omega_c$. More specifically each Chebyshev pole has a real (imaginary) part equal to the real (imaginary) part of a Butterworth pole on the smaller (larger) circle. This explains the positions of vertical and horizontal lines drawn in Figure 11.10.

Design procedure Suppose we wish to design a Chebyshev lowpass filter specified by the parameters Ω_p, A_p, Ω_s, and A_s (see Figure 11.2). For equiripple response in the passband and using Figure 11.9 we choose $\Omega_c = \Omega_p$. Thus, the constraint on the stopband is given by

$$\frac{1}{1 + \epsilon^2 T_N^2(\Omega_s/\Omega_p)} \leq \frac{1}{A^2} \quad \text{or} \quad T_N(\Omega_s/\Omega_p) \geq \frac{1}{\epsilon}\sqrt{A^2 - 1}. \tag{11.47}$$

The passband constraint is satisfied by definition of (11.31b) for all $0 \leq \Omega \leq \Omega_p$. Since $\Omega_s/\Omega_p > 1$, using relation (11.30) for the Chebyshev polynomials, we have

$$\cosh\left[N\cosh^{-1}(\Omega_s/\Omega_p)\right] \geq \frac{1}{\epsilon}\sqrt{A^2 - 1}. \tag{11.48}$$

Solving this inequality for the filter order N and using (11.29a), yields the key design formula

$$N \geq \frac{\cosh^{-1}(\beta)}{\cosh^{-1}(\alpha)} = \frac{\ln(\beta + \sqrt{\beta^2 - 1})}{\ln(\alpha + \sqrt{\alpha^2 - 1})}, \tag{11.49}$$

where, as in the Butterworth approximation case, we have

$$\alpha = \frac{\Omega_s}{\Omega_p}, \quad \beta = \frac{1}{\epsilon}\sqrt{A^2 - 1} = \frac{\sqrt{10^{A_s/10} - 1}}{\sqrt{10^{A_p/10} - 1}}. \tag{11.50}$$

Substituting $\lceil N \rceil$ in (11.41)–(11.43) we obtain the system function $H_C(s)$. We illustrate the design procedure for lowpass Chebyshev filters in Example 11.3.

MATLAB functions for analog Chebyshev I lowpass filters The SP toolbox provides two functions to design analog Chebyshev I lowpass filters. The function
```
[N, Omegac] = cheb1ord(Omegap, Omegas, Ap, As, 's')
```
computes the order N and returns the passband edge frequency Ω_p in `Omegac`, given the design parameters. The filter design and computation of its system function $H_C(s)$ are then obtained by using the

```
[C,D] = cheby1(N, Ap, Omegac, 's')
```
function, which provides the numerator and denominator polynomials in arrays C and D, respectively.

Example 11.3   Design procedure – Chebyshev I approximation

Consider the specifications of the analog lowpass filter given in Example 11.1 and repeated below:

$$-6 \text{ dB} \leq 20 \log_{10} |H(j\Omega)| \leq 0, \quad 0 \leq |\Omega| \leq 2 \tfrac{\text{rad}}{\text{sec}},$$

$$20 \log_{10} |H(j\Omega)| \leq -20 \text{ dB}, \quad 3 \tfrac{\text{rad}}{\text{sec}} \leq |\Omega| < \infty.$$

Then from (10.9), the analog magnitude specifications are

$$\epsilon = 1.7266 \quad \text{and} \quad A = 10.$$

We illustrate the design procedure using the following steps:

Step-1 Compute the parameters α and β using (11.50):

$$\alpha = \frac{3}{2} = 1.5, \quad \beta = \frac{1}{1.7266}\sqrt{10^2 - 1} = 5.7628.$$

Step-2 Compute order N using (11.49) and round upwards to the nearest integer:

$$N = \left\lceil \frac{\ln\left(5.7628 + \sqrt{5.7628^2 - 1}\right)}{\ln\left(1.5 + \sqrt{1.5^2 - 1}\right)} \right\rceil = \lceil 2.5321 \rceil = 3.$$

Step-3 Set $\Omega_c = \Omega_p$ and compute a and b using (11.44) and (11.45):

$$\Omega_c = \Omega_p = 2; \quad \gamma = \left(1/1.7266 + \sqrt{1 + 1/1.7266^2}\right)^{1/3} = 1.2016,$$

$$a = \frac{1}{2}(1.2016 - 1/1.2016) = 0.1847,$$

$$b = \frac{1}{2}(1.2016 + 1/1.2016) = 1.0169.$$

Step-4 Compute the pole locations using (11.41) and (11.42):

$$s_1 = (0.1847)(2) \cos(\tfrac{\pi}{2} + \tfrac{\pi}{6}) + j(1.0169)(2) \sin(\tfrac{\pi}{2} + \tfrac{\pi}{6})$$
$$= -0.1847 - j1.7613,$$

$$s_2 = (0.1847)(2) \cos(\tfrac{\pi}{2} + \tfrac{3\pi}{6}) + j(1.0169)(2) \sin(\tfrac{\pi}{2} + \tfrac{3\pi}{6})$$
$$= -0.3693,$$

$$s_3 = (0.1847)(2) \cos(\tfrac{\pi}{2} + \tfrac{5\pi}{6}) + j(1.0169)(2) \sin(\tfrac{\pi}{2} + \tfrac{5\pi}{6})$$
$$= -0.1847 + j1.7613.$$

Step-5 Compute the filter gain G and the system function $H_C(j\Omega)$ from (11.43):

$$G = -(-0.1847 - j1.7613)(-0.3693)(-0.1847 + j1.7613)(1)$$
$$= 1.1584,$$

$$H_C(s) = \frac{1.1584}{(s + 0.1847 + j1.7613)(s + 0.3693)(s + 0.1847 - j1.7613)}$$

$$= \frac{1.1584}{s^3 + 0.7387s^2 + 3.2728s + 1.1584}. \tag{11.51}$$

The required filter can also be obtained using the following MATLAB script:

```
>> [N, Omegac] = cheb1ord(2, 3, 6, 20, 's')
N =
     3
Omegac =
     2
>> [C,D] = cheby1(N,Ap, Omegac,'s')
C =
          0          0          0    1.1584
D =
     1.0000     0.7387     3.2728     1.1584
```

which agrees with the result obtained in (11.51).   ∎

Example 11.4   Chebyshev I filter design

Consider the analog filter specifications given in Example 11.2 and repeated below.

> Passband edge: $F_p = 40$ Hz,   Passband ripple: $A_p = 1$ dB,
>
> Stopband edge: $F_s = 50$ Hz,   Stopband attenuation: $A_s = 30$ dB.

We want to obtain a lowpass filter using the Chebyshev I approximation. Using the following MATLAB script:

```
>> [N,Omegac] = cheb1ord(2*pi*40,2*pi*50,1,30,'s'); N
N =
     7
>> Fc = Omegac/(2*pi)
Fc =
    40
>> [C,D] = cheby1(N,Ap,Omegac,'s');
```

we obtain a seventh-order filter. Figure 11.11 shows various response plots of the designed filter. In the magnitude response plot the magnitude at $F_c = 40$ Hz is down to $1/\sqrt{1 + \epsilon^2} = 0.89$, while in the log-magnitude plot the response at $F_s = 50$ Hz exceeds 20 dB. Thus the
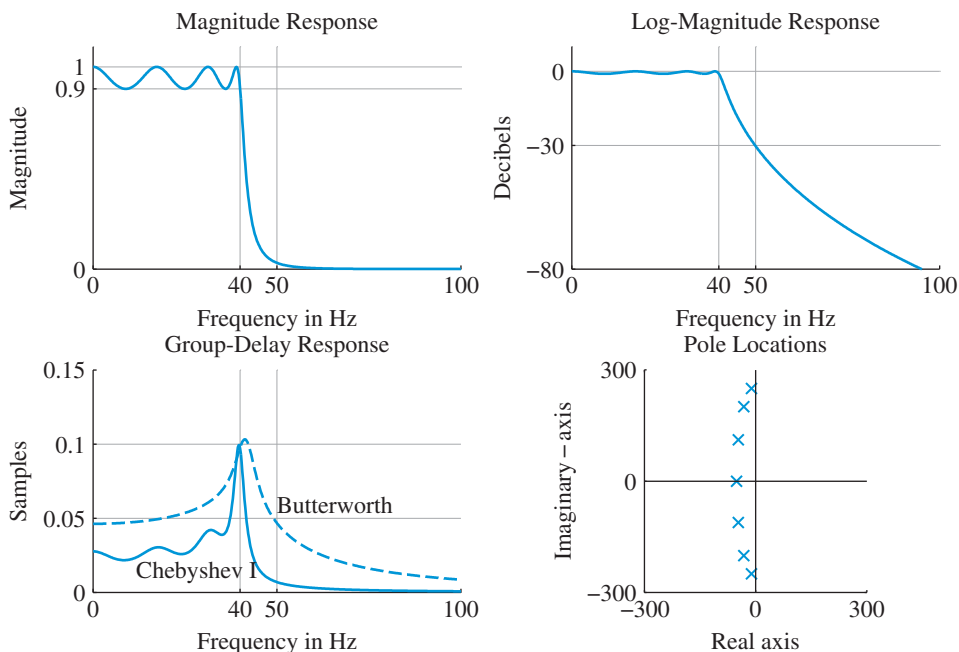
Figure 11.11 Design plots for the seventh-order lowpass Chebyshev I filter in Example 11.4.

design meets the specifications. The group-delay response is more nonlinear than that of the Butterworth design as shown in the group-delay plot. The poles are distributed equiangularly on an ellipse in the left-half of the s-plane with one pole on the real axis since N is odd. Finally, we note that the Chebyshev I design meets the given specification using a much smaller order of 7 compared to 19 for the Butterworth design. ∎

11.2.3   The inverse Chebyshev or Chebyshev II approximation

The Chebyshev magnitude-squared function (11.26) has equiripple behavior in the passband and monotonic maximally flat behavior in the stopband. Consider the magnitude-squared Chebyshev characteristics shown in Figure 11.12(a) in which the passband ripple is $(A^2 - 1)/A^2$ and the stopband attenuation is $\epsilon^2/(1 + \epsilon^2)$, as shown. Replacing the frequency variable Ω/Ω_c in (11.26) by Ω_c/Ω converts the Chebyshev lowpass filter to a Chebyshev highpass filter, by interchanging the characteristics at $\Omega = 0$ and $\Omega = \infty$, as illustrated in Figure 11.12(b). If we next subtract the highpass characteristic from unity, which is an allpass filter, we obtain the *inverse Chebyshev* or *Chebyshev II* characteristic shown in Figure 11.12(c) in which the passband ripple is $1/(1 + \epsilon^2)$ and the stopband attenuation is $1/A^2$, as desired. The resulting magnitude-squared function is

$$|H_{\text{IC}}(j\Omega)|^2 \triangleq 1 - \frac{1}{1 + \epsilon^2 T_N^2(\Omega_c/\Omega)} = \frac{\epsilon^2 T_N^2(\Omega_c/\Omega)}{1 + \epsilon^2 T_N^2(\Omega_c/\Omega)}. \tag{11.52}$$
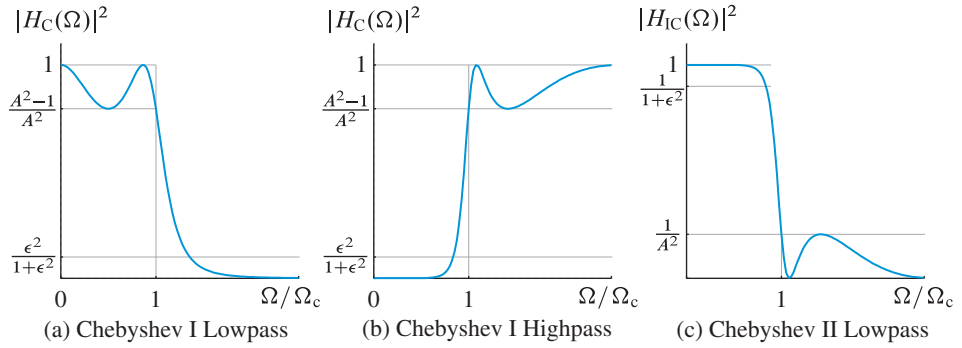
Figure 11.12 Steps for the conversion of Chebyshev I to Chebyshev II magnitude-squared response characteristics.

The Chebyshev II approximation exhibits equiripple behavior in the stopband and monotonic maximally flat behavior in the passband. Thus it has both zeros and poles. Solving the equation

$$T_N(j\Omega_c/s) = \cos[N\cos^{-1}(j\Omega_c/s)] = 0, \tag{11.53}$$

using an approach that parallels that used to solve (11.34), we obtain the zeros

$$\zeta_k = j\frac{\Omega_c}{\cos u_k}, \tag{11.54}$$

where u_k values are given by (11.37) and are on an imaginary axis (see Problem 24). The poles can be found by solving the equation

$$T_N(j\Omega_c/s) = \pm j/\epsilon. \tag{11.55}$$

Comparing (11.55) to (11.33), we conclude that the poles of a Chebyshev II filter are simply the reciprocal of the ones found for the Chebyshev I filter. Thus, we have (see Problem 24)

$$s_k = \frac{\Omega_c}{\sigma_k/\Omega_c + j\Omega_k/\Omega_c} = \frac{\Omega_c^2}{\sigma_k + j\Omega_k}, \tag{11.56}$$

where σ_k and Ω_k are given by (11.41), (11.42), (11.44), and (11.45). In contrast to the poles of Butterworth and Chebyshev filters, the poles and zeros of the inverse Chebyshev filter do *not* lie on any simple geometric curve.

Design procedure Suppose we wish to design a Chebyshev II lowpass filter specified by the parameters Ω_p, A_p, Ω_s, and A_s (see Figure 11.2). Since the response is equiripple in the stopband we choose $\Omega_c = \Omega_s$. Hence, using specifications shown in Figure 11.12, the constraint on the passband is given by

$$\frac{\epsilon^2 T_N^2(\Omega_s/\Omega_p)}{1 + \epsilon^2 T_N^2(\Omega_s/\Omega_p)} \leq \frac{A^2 - 1}{A^2} \quad \text{or} \quad T_N(\Omega_s/\Omega_p) \geq \frac{1}{\epsilon}\sqrt{A^2 - 1}, \tag{11.57}$$

which is the same as in (11.47). Thus order N is given by (11.49) using (11.50). Furthermore, by substituting $\lceil N \rceil$ in (11.49) and solving for α we obtain the exact value of stopband edge frequency Ω'_s at which A_s is satisfied:

$$\Omega'_s \triangleq \Omega_p \cosh(\cosh^{-1}(\beta)/N), \tag{11.58}$$

where β is given in (11.50). Note that $\Omega_p < \Omega'_s \leq \Omega_s$ because N must be an integer. Using $\Omega_c = \Omega'_s$ we compute zeros and poles of $H_{IC}(s)$ from (11.54) and (11.56), respectively. The filter system function $H_{IC}(s)$ is now completely determined. We illustrate the design procedure for lowpass Chebyshev II filters in Example 11.5.

MATLAB functions for analog Chebyshev II lowpass filters The SP toolbox provides two functions to design analog Chebyshev II lowpass filters. Given design parameters, the function

```
[N, Ws] = cheb2ord(Omegap, Omegas, Ap, As, 's')
```
computes the order N in `N` and returns the exact stopband edge frequency Ω'_s in `Ws`. The filter design in the form of its system function is then obtained by using the function

```
[C,D] = cheby2(N, As, Ws, 's')
```
which provides the numerator and denominator polynomials of $H_{IC}(s)$ in arrays `C` and `D`, respectively.

Example 11.5   Design procedure – Chebyshev II approximation
We again consider the specifications of the analog lowpass filter given in Example 11.1:

$$-6 \text{ dB} \leq 20 \log_{10} |H(j\Omega)| \leq 0, \quad 0 \leq |\Omega| \leq 2\tfrac{\text{rad}}{\text{sec}},$$

$$20 \log_{10} |H(j\Omega)| \leq -20 \text{ dB}, \quad 3\tfrac{\text{rad}}{\text{sec}} \leq |\Omega| < \infty.$$

Then from (10.9), the analog magnitude specifications are $\epsilon = 1.7266$ and $A = 10$. We illustrate the design procedure using the following steps:

Step-1 Compute the parameters α and β using (11.50):

$$\alpha = \frac{3}{2} = 1.5, \quad \beta = \frac{1}{1.7266}\sqrt{10^2 - 1} = 5.7628.$$

Step-2 Compute order N using (11.49) and round upwards to the nearest integer:

$$N = \left\lceil \frac{\ln\left(5.7628 + \sqrt{5.7628^2 - 1}\right)}{\ln\left(1.5 + \sqrt{1.5^2 - 1}\right)} \right\rceil = \lceil 2.5321 \rceil = 3.$$

Step-3 Compute exact stopband edge Ω'_s and set $\Omega_c = \Omega'_s$. From (11.58):

$$\Omega'_s = 2\cosh(\cosh^{-1}(5.7628)/3) = 2.697 = \Omega_c.$$

Step-4 Compute a and b. Using (11.44) and (11.45) but with ϵ replaced by $\dfrac{1}{\sqrt{A^2-1}}$ from Figure 11.12, we have:

$$\gamma = \left(10 + \sqrt{10^2 - 1}\,\right)^{1/3} = 2.7121,$$

$$a = \tfrac{1}{2}(2.7121 - 1/2.7121) = 1.1717,$$

$$b = \tfrac{1}{2}(2.7121 + 1/2.7121) = 1.5404.$$

Step-5 Compute zero locations. Using (11.37), (11.54), and $\Omega_c = 2.679$:

$$\zeta_1 = \mathrm{j}2.679/\cos(\pi/6) = \mathrm{j}3.1141,$$

$$\zeta_2 = \mathrm{j}2.679/\cos(3\pi/6) = \mathrm{j}\infty,$$

$$\zeta_3 = \mathrm{j}2.679/\cos(5\pi/6) = -\mathrm{j}3.1141.$$

Thus there is one zero at $\mathrm{j}\infty$ which always exists for odd N. It will be excluded from the numerator polynomial calculations.

Step-6 Compute pole locations. First we compute poles of the corresponding Chebyshev I filter using using (11.41) and (11.42) and then use (11.56):

$$s_1 = \frac{2.679^2}{(1.17)(2.679)\cos(\frac{\pi}{2} + \frac{\pi}{6}) + \mathrm{j}(1.5404)(2.679)\sin(\frac{\pi}{2} + \frac{\pi}{6})} = -0.7443 - \mathrm{j}1.6948,$$

$$s_2 = \frac{2.679^2}{(1.17)(2.679)\cos(\frac{\pi}{2} + \frac{3\pi}{6}) + \mathrm{j}(1.5404)(2.679)\sin(\frac{\pi}{2} + \frac{3\pi}{6})} = -2.3017,$$

$$s_3 = \frac{2.679^2}{(1.17)(2.679)\cos(\frac{\pi}{2} + \frac{5\pi}{6}) + \mathrm{j}(1.5404)(2.679)\sin(\frac{\pi}{2} + \frac{5\pi}{6})} = -0.7443 + \mathrm{j}1.6948.$$

Step-7 Compute the filter gain G and the system function $H_C(\mathrm{j}\Omega)$. The numerator and denominator polynomials are:

$$C(s) \triangleq (s - \zeta_1)(s - \zeta_3) = s^2 + 9.6981,$$

$$D(s) \triangleq (s - s_1)(s - s_2)(s - s_3) = s^3 + 3.7903s^2 + 6.8524s + 7.8861.$$

To obtain the unity filter gain at $\Omega = 0$, $G = 7.8861/9.6981 = 0.8132$. Hence the system function is

$$H_{\mathrm{IC}}(s) = \frac{0.8132s^2 + 7.8861}{s^3 + 3.7903s^2 + 6.8524s + 7.8861}. \tag{11.59}$$

The required filter can also be obtained using the following MATLAB script:

```
>> [N, Ws] = cheb2ord(2, 3, 6, 20, 's');
N =
     3
Omegac =
     2.697
>> [C,D] = cheby2(N,As, Ws,'s')
C =
         0    0.8132    0.0000    7.8861
D =
    1.0000    3.7903    6.8524    7.8861
```

which agrees with the result obtained in (11.59).   ∎

Example 11.6   Chebyshev II filter design

Consider the analog filter specifications given in Example 11.2:

$$\text{Passband edge: } F_p = 40 \text{ Hz}, \quad \text{Passband ripple: } A_p = 1 \text{ dB},$$

$$\text{Stopband edge: } F_s = 50 \text{ Hz}, \quad \text{Stopband attenuation: } A_s = 30 \text{ dB}.$$

We want to obtain a lowpass filter using the Chebyshev II approximation. Using the following MATLAB script:

```
>> [N,Omegac] = cheb2ord(2*pi*40,2*pi*50,1,30,'s'); N
N =
     7
>> Fc = Omegac/(2*pi)
Fc =
    49.8720
>> [C,D] = cheby2(N,As,Omegac,'s');
```

we obtain a seventh-order filter. Figure 11.13 shows various response plots of the designed Chebyshev II filter. In the magnitude response plot the magnitude at $F_p = 40$ Hz is down to $1/\sqrt{1 + \epsilon^2} = 0.89$ while in the log-magnitude plot the response at $F_s = 50$ Hz exceeds stopband attenuation 30 dB but it is met exactly at 49.872 Hz. Thus the design meets specifications. In the group delay response plot, group delays of both Chebyshev I and II are shown. Clearly, Chebyshev II has a better group delay than Chebyshev I in the passband because the zeros of the Chebyshev II filter are on the $j\Omega$ axis while those of the Chebyshev I are all at infinity. Furthermore, it is better (less nonlinear) than that of the Butterworth filter because the Chebyshev II magnitude response is flatter in
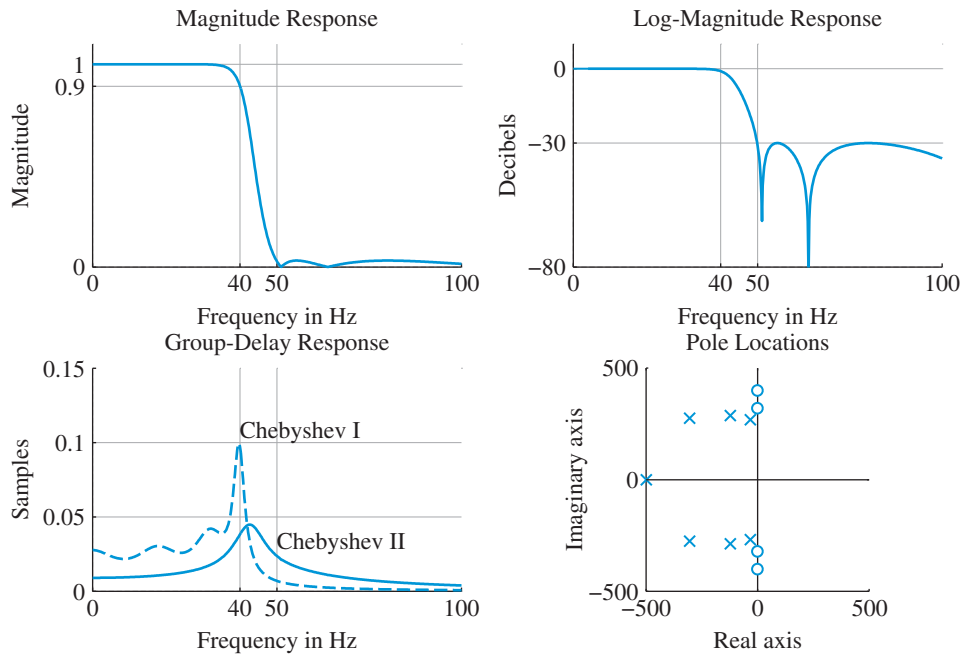
Figure 11.13 Design plots for the seventh-order lowpass Chebyshev II filter in Example 11.6.

the passband due to a sharper transition band. Finally, as discussed, the pole-zero plot shows that poles do not lie on any simple geometrical curve while the zeros are on the $j\Omega$-axis. ∎

11.2.4   The elliptic or Cauer approximation

We have observed that by using the equiripple approximation in one band (Chebyshev and inverse Chebyshev filters) the design requirements are satisfied with a lower order filter compared to maximally flat approximation in both bands (Butterworth filters). Thus, it would be reasonable to expect additional improvement in performance by using equiripple approximation in each band. Indeed, this is made possible by a magnitude-squared function of the form

$$|H_E(j\Omega)|^2 \triangleq \frac{1}{1 + \epsilon^2 R_N^2(\Omega/\Omega_c)}, \tag{11.60}$$

where the Chebyshev polynomial in (11.26) has been replaced by the rational function

$$R_N(\Omega) \triangleq \begin{cases} v^2 \dfrac{\Omega_1^2 - \Omega^2}{1 - \Omega_1^2 \Omega^2} \dfrac{\Omega_3^2 - \Omega^2}{1 - \Omega_3^2 \Omega^2} \cdots \dfrac{\Omega_{2N-1}^2 - \Omega^2}{1 - \Omega_{2N-1}^2 \Omega^2}, & N \text{ even} \\[3mm] v^2 \Omega \dfrac{\Omega_2^2 - \Omega^2}{1 - \Omega_2^2 \Omega^2} \dfrac{\Omega_4^2 - \Omega^2}{1 - \Omega_4^2 \Omega^2} \cdots \dfrac{\Omega_{2N}^2 - \Omega^2}{1 - \Omega_{2N}^2 \Omega^2}, & N \text{ odd} \end{cases} \tag{11.61}$$
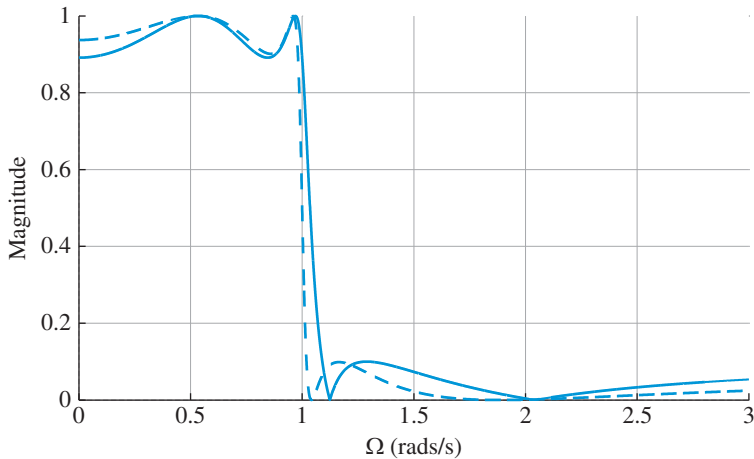
Figure 11.14 Elliptic filter magnitude response for $N = 4$ and $\Omega_c = 1$: Nonequiripple (dashed line) and equiripple (solid line) responses.

specified by the parameters ν, Ω_k, and N. The function $R_N(\Omega)$ in (11.61) has the following properties: (a) $R_N(1/\Omega) = 1/R_N(\Omega)$, (b) $R_N(\Omega)$ is even for even N and odd for odd N, and (c) the denominator roots are reciprocal of the numerator roots.

If we choose Ω_k to lie within the interval $0 \le \Omega < 1$, the function $R_N^2(\Omega)$ will have zero values at Ω_k and infinite values at $1/\Omega_k$. Thus, $|H_E(j\Omega)| = 1$ when $R_N(\Omega) = 0$ and $|H_E(j\Omega)| = 0$ when $R_N(\Omega) = \infty$. Figure 11.14 shows a plot of $|H_E(j\Omega)|$ for $N = 4$, $\Omega_c = 1$ rads/s, $\Omega_1 = 0.53$ rads/s, $\Omega_3 = 0.96$ rads/s, and $\epsilon = 1$ (dashed line). We note that the arbitrary selection of Ω_k always produces maxima equal to one at frequencies Ω_k of the passband and minima equal to zero at frequencies $1/\Omega_k$ of the stopband. However, the minima in the passband and the maxima in the stopband are *not* equiripple. It has been shown, see Papoulis (1957), that if we make these minima and maxima equiripple, the result is a filter with the minimum transition band among all filters with the same order, passband ripple, and stopband attenuation. Elliptic filters are optimal in the same sense that are linear phase FIR filters designed using the Parks–McClellan method. Figure 11.14 shows the filter obtained by optimally selecting the values of Ω_1 and Ω_3 to attain equiripple behavior (solid line). This selection of frequencies Ω_k makes $R_N(j\Omega)$ a *rational Chebyshev function*.

Filters with equiripple behavior in both the passband and the stopband were introduced by W. Cauer (1932), see Cauer *et al.* (2000), and are also known as *Cauer filters*. However, because the derivation of Cauer filters relies heavily on the theory of elliptic functions the more widely used name is *elliptic filters*. For example, the the required filter order is given by

$$N \ge \frac{K(1/\alpha)K(\sqrt{1 - 1/\beta^2})}{K(1/\beta)K(\sqrt{1 - 1/\alpha^2})}, \quad K(x) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - x^2 \sin^2 \theta}}, \quad (11.62)$$

where $K(x)$ is the complete elliptic integral of the first kind, which can be evaluated by MATLAB function `ellipke`; the parameters α and β are given by (11.50). Typical magnitude responses for odd and even N are shown in Figure 11.15.
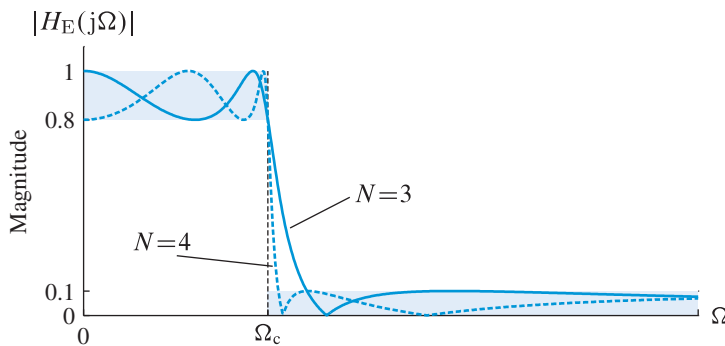
Figure 11.15 Magnitude responses of two Chebyshev I lowpass filters with ripple parameter $\epsilon = 0.75$ and orders $N = 3$ (solid-line) and $N = 4$ (dashed-line).

The selection of Ω_k value to achieve equiripple behavior in both bands is a complicated process which can be done either analytically or numerically. These techniques are beyond the scope of this book. Complete treatments of elliptic filters are given by Antoniou (2006), Parks and Burrus (1987), and Daniels (1974). Therefore, we use functions provided in MATLAB.

MATLAB functions for analog elliptic lowpass filters The SP toolbox provides two functions to design analog elliptic lowpass filters. Given design parameters, the function

```
[N, Omegac] = ellipord(Omegap,Omegas,Ap,As,'s')
```
computes the order N in `N` and returns the passband edge frequency Ω_p in `Omegac`. The filter design in the form of its system function $H_E(s)$ is then obtained by using the function

```
[C,D] = ellip(N,Ap,As,Omegac,'s'),
```
which provides the numerator and denominator polynomials of $H_E(s)$ in arrays `C` and `D`, respectively.

Example 11.7   Elliptic filter design using MATLAB

Consider again the specifications of the analog lowpass filter given in Example 11.1:

$$-6 \text{ dB} \leq 20 \log_{10} |H(j\Omega)| \leq 0, \qquad 0 \leq |\Omega| \leq 2\tfrac{\text{rad}}{\text{sec}},$$

$$20 \log_{10} |H(j\Omega)| \leq -20 \text{ dB}, \qquad 3\tfrac{\text{rad}}{\text{sec}} \leq |\Omega| < \infty.$$

We obtain system function $H_E(s)$ using the following MATLAB script:

```
>> [N, Omegac] = ellipord(2,3,6,20,'s'); N
N =
      2
>> [C,D] = ellip(N,Ap,As,Omegac,'s')
C =
    0.1000            0    1.3526
D =
    1.0000    0.6926    2.6987
```

Hence the system function is given by

$$H_E(s) = \frac{0.1s^2 + 1.3526}{s^2 + 0.6926s + 2.6987}, \tag{11.63}$$

which satisfies the same specifications as the other analog filters using the lowest order $N = 2$. ∎

Example 11.8   Elliptic filter design
We now design an elliptic lowpass filter using the specifications given in Example 11.2:

> Passband edge: $F_p = 40$ Hz,   Passband ripple: $A_p = 1$ dB,
>
> Stopband edge: $F_s = 50$ Hz,   Stopband attenuation: $A_s = 30$ dB.

Using the following MATLAB script:

```
>> [N,Omegac] = ellipord(2*pi*40,2*pi*50,1,30,'s'); N
N =
    5
>> [C,D] = ellip(N,Ap,As,Omegac,'s');
```
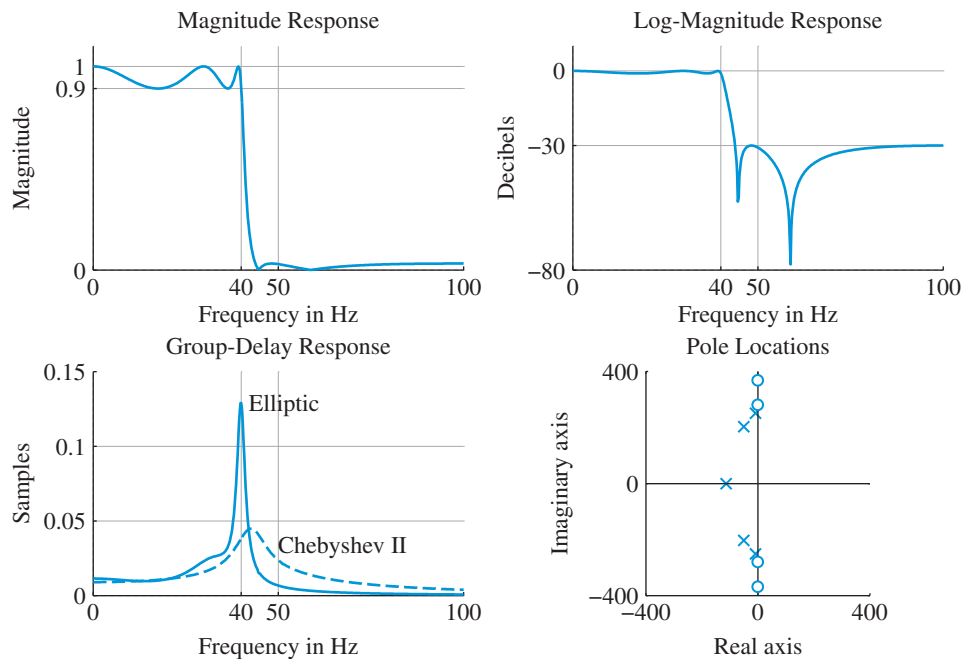

Figure 11.16 Design plots for the fifth-order lowpass elliptic filter in Example 11.8.

we obtain a fifth-order filter. Figure 11.16 shows various response plots of the designed elliptic lowpass filter. In the magnitude response plot the magnitude at 40 Hz is down to $1/\sqrt{1 + \epsilon^2} = 0.89$ while in the log-magnitude plot the response at 60 Hz exceeds stopband attenuation 20 dB. In the group-delay response plot, group delays of elliptic and Chebyshev II are shown. Clearly, Chebyshev II has a better group delay than elliptic in the passband. The pole-zero plot shows that poles do not lie on any simple geometrical curve while the zeros are on the $j\Omega$-axis.   ■

After a careful study of analog lowpass designs of Butterworth, Chebyshev I & II, and elliptic filters in Examples 11.2, 11.4, 11.6, and 11.8 it is evident that the elliptic filter has the lowest order and the sharpest transition band while the Butterworth filter has the opposite characteristics for the same set of specifications. The Chebyshev filters perform somewhere in between these two extremes. However, if group-delay characteristics are important then Butterworth and Chebyshev-II filters have the better responses than those of the Chebyshev I and elliptic filters. Therefore in practice, it is essential to consider each one of the approximations depending on the application and the cost of the implementation.

Figure 11.17 shows design plots of an order $N = 7$ elliptic filter designed for more realistic specifications of passband edge at 40 Hz with ripple of 0.1 dB and stopband edge at 50 Hz with stopband attenuation of 50 dB. It was designed using the MATLAB script:
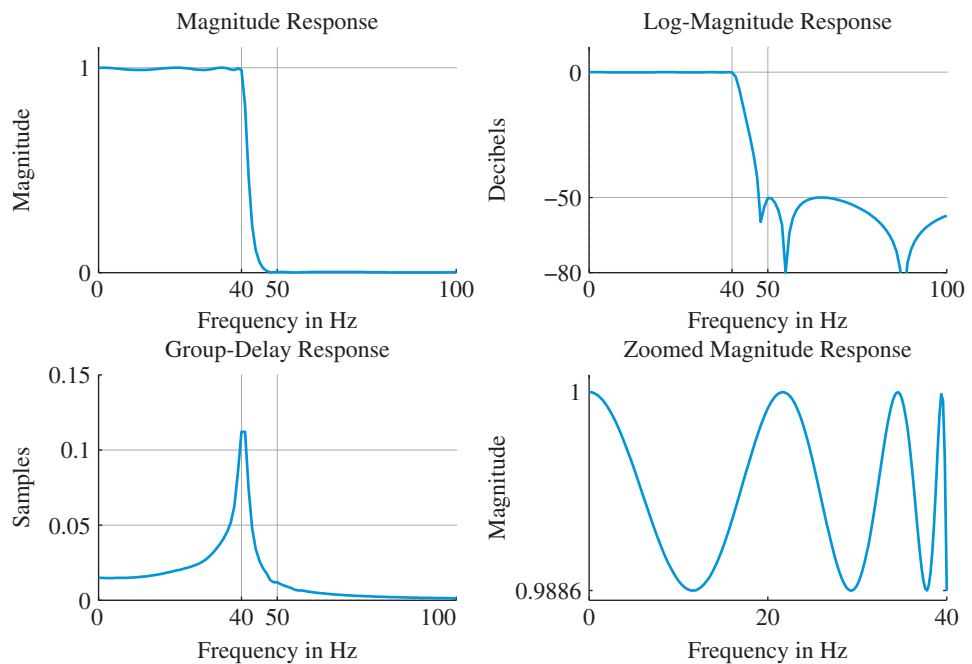
Figure 11.17 Design plots for the seventh-order lowpass elliptic filter with specifications $F_p = 40$ Hz, $F_s = 50$ Hz, $A_p = 0.1$ dB, and $A_s = 50$ dB.

```
>> [N, Omegac] = ellipord(2*pi*40, 2*pi*50, 0.1, 50, 's');
>> [C,D] = ellip(N,0.1,50,Omegac,'s').
```

The zoomed magnitude response shows $N = 7$ maxima and minima values in the passband. The same number of optimum values are also exhibited in the stopband.

11.3   Transformation of continuous-time filters to discrete-time IIR filters

Several procedures for conversion of continuous-time filters (analog prototypes) to discrete-time filters, see Rabiner and Gold (1975), have been developed over the years. However, we only discuss two methods: impulse-invariance transformation and bilinear transformation. The first has limited applicability, but it has educational value; the second has universal applicability and it is the most widely used method in IIR filter design software packages.

Each transformation is equivalent to a mapping function $s = f(z)$ from the s-plane to the z-plane. Any useful mapping should satisfy three desirable conditions:

• A rational $H_c(s)$ should be mapped to a rational $H(z)$ (realizability):

$$\text{Rational } H_c(s) \longrightarrow \text{Rational } H(z). \tag{11.64}$$

• The imaginary axis of the s-plane is mapped on the unit circle of the z-plane:

$$\{s = j\Omega \mid -\infty < \Omega < \infty\} \longrightarrow \{z = e^{j\omega} \mid -\pi < \omega \le \pi\}. \tag{11.65}$$

• The left-half s-plane is mapped into the interior of the unit circle of the z-plane:

$$\{s \mid \mathcal{R}e(s) < 0\} \longrightarrow \{z \mid |z| < 1\}. \tag{11.66}$$

Condition (11.64) is needed to preserve the frequency characteristics of the continuous-time filter. Condition (11.65) guarantees that a stable continuous-time filter is mapped into a stable discrete-time filter. Any mapping procedure must satisfy (11.66). The reason is that stable continuous-time systems have their poles on the left-half s-plane, whereas stable discrete-time systems have their poles inside the unit circle of the z-plane as shown in Figure 11.18. Clearly, different procedures give rise to different mapping functions, and, hence, the resulting discrete-time filters are different.

11.3.1   Impulse-invariance transformation

The most natural way to convert a continuous-time filter to a discrete-time filter is by sampling its impulse response (see Section 6.3),

$$h[n] \triangleq T_d \, h_c(nT_d), \tag{11.67}$$

where T_d is called the design sampling period. This transformation is known as *impulse-invariance* because it preserves the shape of the impulse response. The frequency response
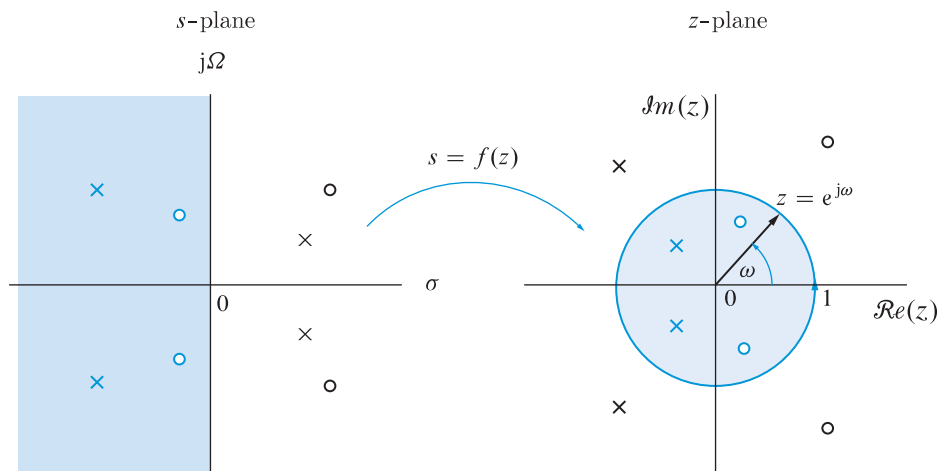
Figure 11.18 Two desirable requirements, (11.65) and (11.66), for functions of the form $s = f(z)$ that map the continuous-time s-plane to the discrete-time z-plane.

of the resulting discrete-time filter is related to the frequency response of the continuous-time filter by

$$H(e^{j\omega}) = \sum_{k=-\infty}^{\infty} H_c\left(j\frac{\omega}{T_d} + j\frac{2\pi}{T_d}k\right). \tag{11.68}$$

Thus, in general, the impulse-invariance mapping causes aliasing, as illustrated in Figure 6.16. The fundamental difference between continuous-time and discrete-time filters is the periodicity of frequency-response for discrete-time systems, that is, $H(e^{j\omega})$ is periodic whereas $H_c(j\Omega)$ is nonperiodic. If the continuous-time filter is bandlimited, that is,

$$H_c(j\Omega) = 0, \quad |\Omega| \geq \pi/T_d \tag{11.69}$$

then, we have

$$H(e^{j\omega}) = H_c\left(j\frac{\omega}{T_d}\right). \quad |\omega| \leq \pi \tag{11.70}$$

Mapping for the impulse-invariance transformation To determine the transformation $s = f(z)$ corresponding to (11.67), we start with the partial fraction expansion of $H_c(s)$, which for $M < N$ is given by (see Section 5.11.3)

$$H_c(s) = \sum_{k=1}^{N} \frac{A_k}{s - s_k}. \tag{11.71}$$

For simplicity we assume that the poles are distinct; multiple order poles are discussed in Problem 52. Taking the inverse Laplace transform yields the impulse response of the continuous-time filter

$$h_c(t) = \sum_{k=1}^{N} A_k\, e^{s_k t}\, u(t). \tag{11.72}$$

Hence the impulse response of the discrete-time filter is given by

$$h[n] = T_d h_c(nT_d) = \sum_{k=1}^{N} T_d A_k \left(e^{s_k T_d}\right)^n u[n], \tag{11.73}$$

and the system function of the discrete-time system is therefore given by

$$H(z) = \sum_{n=0}^{\infty} h[n] z^{-n} = \sum_{n=0}^{\infty} \sum_{k=1}^{N} T_d A_k \left(e^{s_k T_d}\right)^n. \tag{11.74}$$

Interchanging orders of summation, assuming that $\left|e^{s_k T_d}\right| < 1$, and summing over n yields the formula

$$H(z) = \sum_{k=1}^{N} \frac{T_d A_k}{1 - e^{s_k T_d} z^{-1}}. \tag{11.75}$$

Comparing (11.75) to (11.71) we conclude that, for single poles, $H(z)$ is obtained from $H_c(s)$ by using the following mapping

$$\frac{1}{s - s_k} \longrightarrow \frac{T_d}{1 - e^{s_k T_d} z^{-1}} = \frac{T_d}{1 - p_k z^{-1}}, \tag{11.76}$$

where

$$p_k \triangleq e^{s_k T_d} \tag{11.77}$$

maps the poles of the continuous-time filter to the poles of the discrete-time filter. We note that mapping (11.76) relates the locations of the poles of $H_c(s)$ and $H(z)$ but not the locations of the zeros. A method that maps both poles and zeros using (11.76) is the matched z-transformation (see Problem 34).

From (11.77) it is obvious that the mapping $s = f(z)$ corresponding to impulse-invariance is $s = \ln(z)/T_d$ or

$$z = e^{sT_d}. \tag{11.78}$$

Since $s = \sigma + j\Omega$ and $z = re^{j\omega}$, substitution into (11.78) yields

$$r = e^{\sigma T_d}, \tag{11.79a}$$

$$\omega = \Omega T_d. \tag{11.79b}$$

We note that $\sigma < 0$ implies that $0 < r < 1$ and $\sigma > 0$ implies that $r > 1$. Therefore, impulse-invariance satisfies the desirable condition (11.66), that is, the left-half s-plane is mapped inside the unit circle of the z-plane. Since $\sigma = 0$ yields $r = 1$, the frequency axis $s = j\Omega$ is mapped on the unit circle; however, this mapping is *not* one-to-one. Indeed, a careful inspection of the periodic relationship

$$z = e^{\sigma T_d} e^{j(\Omega + 2\pi k/T_d)T_d} = e^{\sigma T_d} e^{j\Omega T_d} \tag{11.80}$$

shows that a horizontal strip of height $2\pi/T_d$ in the s-plane is mapped into the entire z-plane. The left-half of the strip is mapped into the interior of the unit circle, the right-half
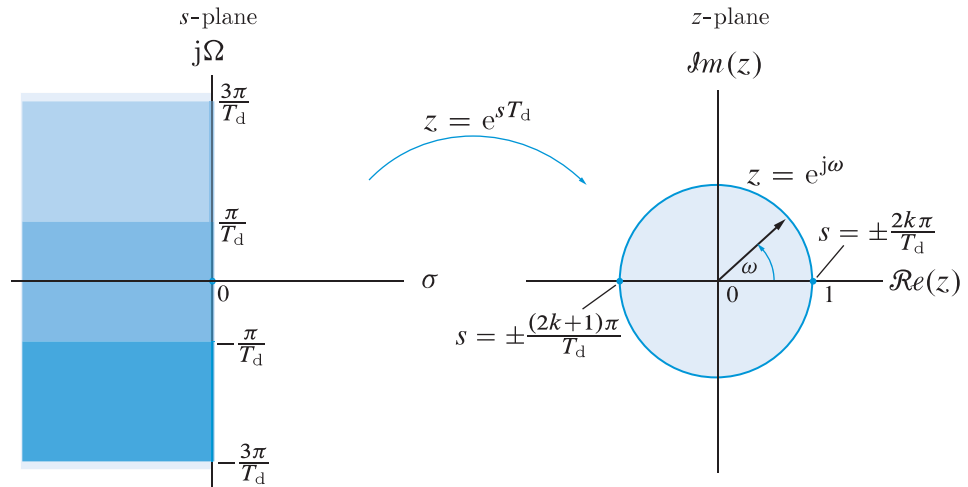
Figure 11.19 The mapping from the s-plane to the z-plane corresponding to the impulse-invariance transformation.

of the strip into the exterior of the unit circle, and the imaginary axis of the strip onto the unit circle. From Figure 11.19 we can see that the source of the aliasing effect is that the mapping of (11.78) in not one-to-one, that is, the desirable condition (11.65) is not satisfied. Substituting (11.79b) and $s = j\Omega$ in (11.68) yields

$$H(z)|_{z=e^{sT_d}} = \sum_{k=-\infty}^{\infty} H_c\left(s + j\frac{2\pi k}{T_d}\right), \tag{11.81}$$

which provides the relationship between the system function of the discrete-time filter and the corresponding continuous-time filter under the impulse-invariance transformation.

MATLAB function for impulse-invariance The SP toolbox provides the function `[B,A] = impinvar(C,D, Fd)` that computes the numerator and denominator polynomial coefficients of the digital filter $H(z)$ in arrays `B` and `A` respectively, given the same for the analog filter $H_c(s)$ in arrays `C` and `D` respectively, and the design sampling frequency $1/T_d$ in `Fd`. This function can also transform analog filter system functions with multiple poles.

Example 11.9

Consider a continuous-time first-order filter with system function

$$h_c(t) = e^{-2t}u(t) \overset{\mathcal{L}}{\longleftrightarrow} H_c(s) = \frac{1}{s+2}. \quad \mathcal{R}e(s) > -2$$

The discrete-time filter obtained by the impulse-invariance transformation using $T_d = 0.1$ is

$$h[n] = 0.1h_c(0.1n) = 0.1e^{-0.2n}u[n] \overset{\mathcal{Z}}{\longleftrightarrow} H(z) = \frac{0.1}{1 - e^{-0.2}z^{-1}}, \quad |z| > e^{-0.2}$$

or

$$H(z) = \frac{0.1}{1 - 0.8187z^{-1}}, \quad |z| > 0.8187 \tag{11.82}$$

We see that a first-order continuous-time system is transformed into a first-order discrete-time system. Furthermore, the mapping preserves stability and the lowpass characteristic of the magnitude response. The system function $H(z)$ can also be obtained using the MATLAB script

```
>> [B,A]=impinvar(1,[1,2],10)
B =
    0.1000
A =
    1.0000   -0.8187
```

which is the same as in (11.81).

Consider next the first-order highpass filter obtained by

$$H'_c(s) = 1 - H_c(s) = 1 - \frac{1}{s+2} = \frac{s+1}{s+2}.$$

The impulse response of the highpass filter is given by

$$h'_c(t) = \delta(t) - e^{-2t}u(t).$$

The presence of a delta function creates fundamental theoretical problems in the sampling of $h'_c(t)$; basically, we cannot sample $h'_c(t)$ at $t = 0$. Since the delta function appears when $N = M$, we cannot use the impulse-invariant transformation for systems with improper system functions, like highpass and bandstop filters (see Tutorial Problem 8 for an illustration of the implications). Impulse-invariance can be applied to lowpass and bandpass filters that have strictly proper system functions. The `impinvar` function should not be used to transform an improper rational function $H_c(s)$. ■

Design procedure Suppose we wish to design a digital lowpass filter $H(z)$ specified by the parameters ω_p, A_p, ω_s, and A_s (see Figure 10.1). We start by choosing the design sampling interval T_d which is arbitrary, and then using (11.79b) we map ω_p into $\Omega_p = \omega_p/T_d$ and ω_s into $\Omega_s = \omega_s/T_d$. Next, we design the equivalent analog filter $H_c(s)$ using the Butterworth or Chebyshev I approximations that satisfies the specifications Ω_p, A_p, Ω_s, and A_s. The Chebyshev II and elliptic approximations are not suitable due to their equiripple response in the stopband which leads to aliasing (see Example 11.12 and Tutorial Problem 9). We perform a partial fraction expansion on the rational function H_c and map its poles $\{s_k\}$ into digital poles $\{p_k\}$ using (11.78). Finally, we assemble the desired digital filter system function $H(z)$ using (11.75). Alternatively, we use the `impinvar` function in MATLAB. We illustrate this design procedure in Examples 11.10 and 11.11. It should be noted that MATLAB does not provide a function to obtain digital lowpass filter design using the impulse-invariance transformation.

Example 11.10   Impulse-invariance transformation – Butterworth

We want to design a lowpass digital Butterworth filter to satisfy specifications:

Passband edge: $\omega_p = 0.25\pi$ rad,   Passband ripple: $A_p = 1$ dB,

Stopband edge: $\omega_s = 0.4\pi$ rad,   Stopband attenuation: $A_s = 30$ dB.

We illustrate the design procedure using the following steps:

Step-1 Choose design sampling interval T_d. Let $T_d = 0.1$ s.
Step-2 Compute the equivalent analog filter band edge frequencies. Using (11.79b), we obtain

$$\Omega_p = \frac{0.25\pi}{0.1} = 7.8540 \quad \text{and} \quad \Omega_s = \frac{0.4\pi}{0.1} = 12.5664.$$

Step-3 Design the analog lowpass filter $H_c(s)$. Using the MATLAB script

```
>> [N,Omegac] = buttord(7.8540,12.5664,1,30,'s'); N
N =
      9
>> [C,D] = butter(N,Omegac,'s');
```

we obtain a ninth-order Butterworth approximation.
Step-4 Transform $H(s)$ into $H(z)$ using MATLAB. Finally using the script

```
>> [B,A] = impinvar(C,D,1/Td);
```

we obtain the desired digital filter $H(z) = B(z)/A(z)$ using the coefficients in the arrays B and A.

Figure 11.20 shows design plots of the resulting digital Butterworth filter. The magnitude response specifications are met exactly at $\omega_s = 0.4\pi$ but are exceeded at $\omega_p = 0.25\pi$. The impulse response plot shows the impulse response $h_c(t)$ of the analog filter (solid line) with samples of $h[n]$, scaled by $1/T_d$, superimposed on it. Clearly, the impulse-invariance preserves the shape of the analog impulse response.   ■

Example 11.11   Impulse-invariance transformation – Chebyshev I

To design a digital lowpass Chebyshev I filter that satisfies the specifications given in Example 11.10, we use the MATLAB script:

```
>> omegap = 0.25*pi; omegas = 0.4*pi; Ap = 1; As = 30;
>> Td = 0.1; Omegap = omegap/Td; Omegas = omegas/Td;
>> [N,Omegac] = cheb1ord(Omegap,Omegas,Ap,As,'s'); N
N =
    5
>> [C,D] = cheby1(N,Ap,Omegac,'s'); [B,A] = impinvar(C,D,1/Td);
```
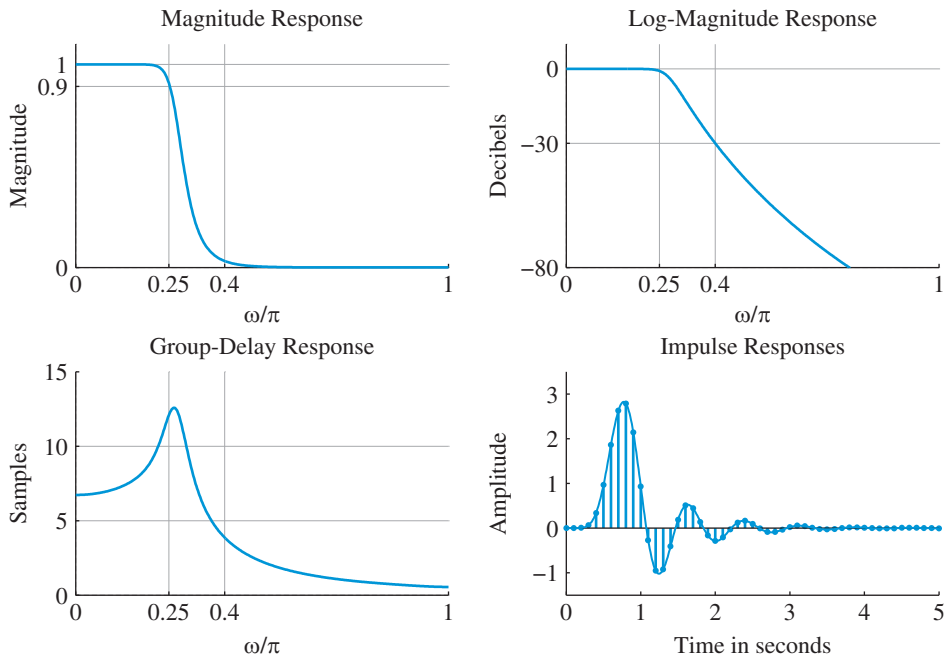
Figure 11.20 Design plots for the ninth-order digital lowpass Butterworth filter in Example 11.10 with specifications $\omega_p = 0.25\pi$, $\omega_s = 0.4\pi$, $A_p = 1$ dB, and $A_s = 30$ dB.

that designs a fifth-order Chebyshev I digital filter. Figure 11.21 shows design plots of the resulting filter. The magnitude response specifications are met exactly at $\omega_p = 0.25\pi$ but are exceeded at $\omega_s = 0.4\pi$. The impulse response plots of the prototype analog filter $h_c(t)$ (solid line) and the digital filter $h[n]$ (samples scaled by $1/T_d$) show that the impulse-invariance preserves the shape of the analog impulse response. ∎

We note that as T_d gets smaller, that is $F_d \triangleq 1/T_d$ gets larger, the effects of aliasing become negligible and the continuous-time and discrete-time frequency responses become comparable (see Tutorial Problem 10). Because of the aliasing effect, the impulse-invariance method is meaningful for filters with bandlimited frequency responses, like lowpass and bandpass filters. However, as Example 11.12 illustrates, design by impulse-invariance may be problematic even for lowpass filters.

Example 11.12   Impulse-invariance transformation – Chebyshev II

Again consider specifications given in Example 11.10. We want to design the digital filter using the Chebyshev II approximation. The MATLAB script

```
>> omegap = 0.25*pi; omegas = 0.4*pi; Ap = 1; As = 30;
>> Td = 0.1; Omegap = omegap/Td; Omegas = omegas/Td;
>> [N,Omegac] = cheb2ord(Omegap,Omegas,Ap,As,'s'); N
```
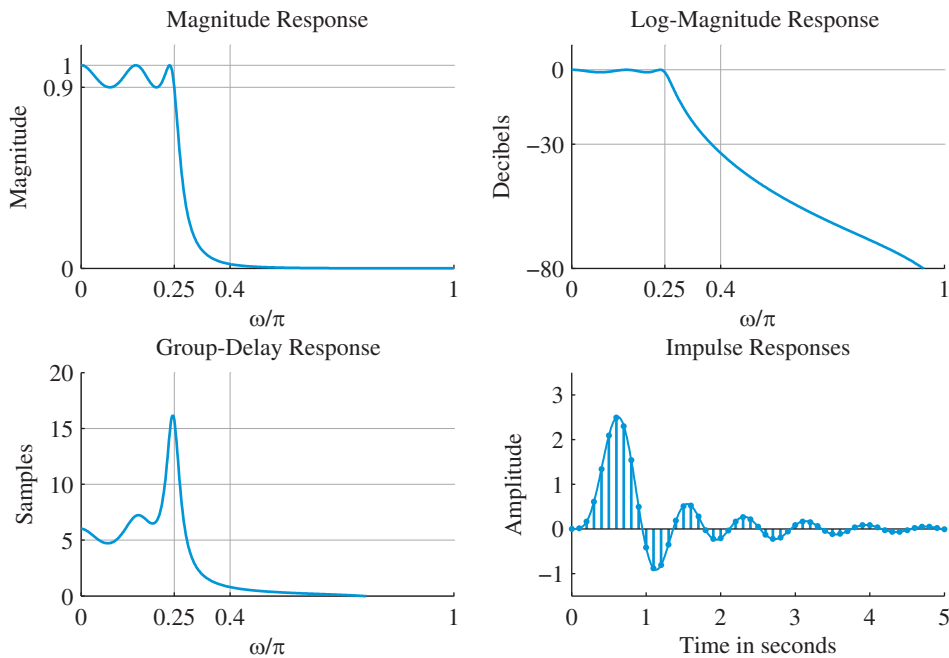
Figure 11.21 Design plots for the fifth-order digital lowpass Chebyshev I filter in Example 11.11 with specifications $\omega_p = 0.25\pi$, $\omega_s = 0.4\pi$, $A_p = 1$ dB, and $A_s = 30$ dB.

```
N =
     5
>> [C,D] = cheby2(N,As,Omegac,'s'); [B,A] = impinvar(C,D,1/Td);
```

designs a fifth-order Chebyshev II digital filter. Figure 11.22 shows design plots of the resulting filter. Clearly the magnitude response specifications are not met either at $\omega_p = 0.25\pi$ or at $\omega_s = 0.4\pi$. The equiripple behavior of the analog filter in the stopband has resulted in a considerable amount of aliasing there by making the digital filter ineffective. The impulse response plots, however, do show that the digital filter $h[n]$ (samples scaled by $1/T_d$) preserves the impulse response of the analog filter $h_c(t)$ (solid line). ■

11.3.2 Bilinear transformation

To avoid the limitations of impulse-invariance transformation caused by the aliasing effect, we need a one-to-one mapping from the s-plane to the z-plane. The *bilinear transformation* is an invertible nonlinear mapping between the s-plane and the z-plane defined by

$$s = f(z) \triangleq \frac{2}{T_d} \frac{1 - z^{-1}}{1 + z^{-1}}. \tag{11.83}$$

The parameter T_d, which has no effect on the design process, may be given any value that simplifies the derivations. We emphasize that, in contrast to the impulse-invariance
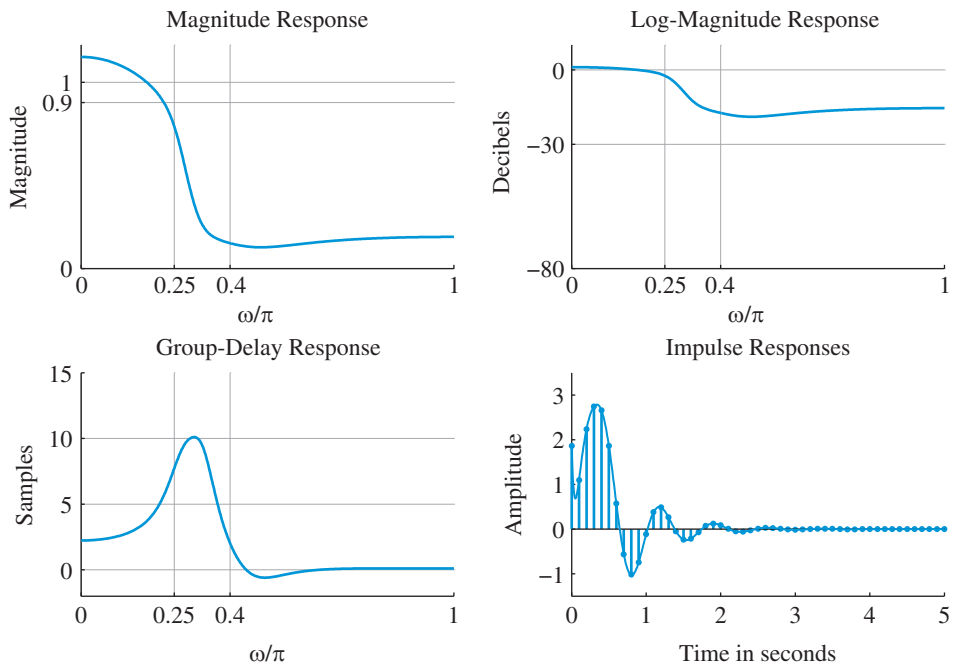
Figure 11.22 Design plots for the fifth-order digital lowpass Chebyshev II filter with specifications $\omega_p = 0.25\pi$, $\omega_s = 0.4\pi$, $A_p = 1$ dB, and $A_s = 30$ dB.

case, T_d does *not* have any useful interpretation as a sampling interval because the bilinear transformation does *not* involve any sampling operation.

The mapping defined by (11.83) satisfies the three desirable conditions (11.64)–(11.66); this makes the bilinear transformation the most popular technique for mapping continuous-time filters to discrete-time filters.

Realizability The bilinear transformation is applied by replacing each occurrence of s in $H_c(s)$ by the transformation function (11.83). Formally, we write

$$H(z) = H_c(s)\Big|_{s = \frac{2}{T_d}\frac{1-z^{-1}}{1+z^{-1}}}. \tag{11.84}$$

We can easily show that a single zero (or pole) ν_k is transformed by the bilinear transformation as follows:

$$s - \nu_k = \frac{2(1 - T_d\nu_k/2)}{T_d(1 + z^{-1})}\left[1 - \frac{1 + T_d\nu_k/2}{1 - T_d\nu_k/2}z^{-1}\right]. \tag{11.85}$$

Therefore, we can implement the bilinear transformation by individually mapping the zeros and poles of $H_c(s)$ in (11.2). The result is

$$H(z) = G \frac{(1 + z^{-1})^{N-M} \prod_{k=1}^{M}(1 - z_k z^{-1})}{\prod_{k=1}^{N}(1 - p_k z^{-1})}, \tag{11.86}$$

where

$$z_k = \frac{1 + T_d \zeta_k/2}{1 - T_d \zeta_k/2}, \quad p_k = \frac{1 + T_d s_k/2}{1 - T_d s_k/2}, \quad G = \frac{\beta_0 \left(\frac{T_d}{2}\right)^{N-M} \prod_{k=1}^{M}\left(1 - \zeta_k \frac{T_d}{2}\right)}{\prod_{k=1}^{N}\left(1 - s_k \frac{T_d}{2}\right)}. \tag{11.87}$$

Because the mapping (11.83) is a rational function, a rational $H_c(s)$ always gives a rational $H(z)$. The bilinear mapping preserves the order of the system (number of poles N), but increases the number of zeros from M to N (when $N > M$) by placing $(N - M)$ zeros at $z = -1$.

MATLAB function for bilinear transformation The approach given in (11.86) and (11.87) is available in the SP toolbox as the function

      [zd,pd,G] = bilinear(zc,pc,beta0,Fd)

that computes zeros z_k, poles p_k, and gain G in column vectors `zd`, `pd`, and `G` of the digital filter $H(z)$ respectively, given the zeros ζ_k, poles s_k, and gain β_0 in column vectors `zc`, `pc`, and `beta0` of the analog filter $H_c(s)$ respectively, and the design sampling frequency $1/T_d$ in `Fd`. Similarly, the invocation

      [B,A] = bilinear(C,D,Fd)

computes the numerator and denominator polynomial coefficients of the digital filter $H(z)$ in arrays `B` and `A` respectively, given the same for the analog filter $H_c(s)$ in arrays `C` and `D` respectively, and the design sampling frequency $1/T_d$ in `Fd`.

Example 11.13
Consider the following analog filter system function:

$$H_c(s) = \frac{5(s + 2)}{(s + 3)(s + 4)} = \frac{5s + 10}{s^2 + 7s + 12}.$$

To transform this into a digital filter system function $H(z)$ let $T_d = 2$ or $T_d/2 = 1$. Using (11.87), the zeros of $H(z)$ are given by

$$z_1 = \frac{1 + (-2)}{1 - (-2)} = -\frac{1}{3} \quad \text{and} \quad z_2 = -1 \quad (\because 1 + z^{-1} \text{ factor}),$$

the poles of $H(z)$ are given by

$$p_1 = \frac{1 + (-3)}{1 - (-3)} = -\frac{1}{2} \quad \text{and} \quad p_2 = \frac{1 + (-4)}{1 - (-4)} = -\frac{3}{5},$$

and the gain G is given by

$$G = \frac{5(1)(1 - (-2))}{(1 - (-3))(1 - (-4))} = \frac{3}{4}.$$

Hence

$$H(z) = \frac{3}{4} \frac{(1 + z^{-1})(1 + \frac{1}{3}z^{-1})}{(1 + \frac{1}{2}z^{-1})(1 + \frac{3}{5}z^{-1})} = \frac{0.75 + z^{-1} + 0.25z^{-2}}{1 + 1.1z^{-1} + 0.3z^{-2}}.$$

Using the MATLAB script

```
>> zc = -2; pc = [-3,-4]'; beta0 = 5; Td = 2;
>> [zd,pd,G] = bilinear(zc,pc,beta0,1/Td); zd', pd', G
zd =
   -0.3333   -1.0000
pd =
   -0.5000   -0.6000
kd =
    0.7500
>> C = [5,10]; D = [1,7,12]; [Bd,Ad] = bilinear(C,D,1/Td)
Bd =
    0.7500    1.0000    0.2500
Ad =
    1.0000    1.1000    0.3000
```

we obtain the same values as above.   ■

Mapping properties To develop the mapping properties of the bilinear transformation we solve (11.83) for z to obtain

$$z = re^{j\omega} = \frac{2/T_d + s}{2/T_d - s} = \frac{2/T_d + \sigma + j\Omega}{2/T_d - \sigma - j\Omega}. \tag{11.88}$$

If we express z in polar coordinates we have

$$r = |z| = \left[\frac{(2/T_d + \sigma)^2 + \Omega^2}{(2/T_d - \sigma)^2 + \Omega^2} \right]^{1/2} \Rightarrow \begin{cases} \text{if } \sigma < 0 \text{ then } r < 1 \\ \text{if } \sigma = 0 \text{ then } r = 1 \\ \text{if } \sigma > 0 \text{ then } r > 1 \end{cases} \tag{11.89a}$$

$$\omega = \tan^{-1}\left(\frac{\Omega}{2/T_d + \sigma}\right) + \tan^{-1}\left(\frac{\Omega}{2/T_d - \sigma}\right). \tag{11.89b}$$

Thus, the bilinear transformation maps (a) the open left-half s-plane in the interior of the unit circle, (b) the $j\Omega$ axis onto the unit circle, and (c) the right-half s-plane on the exterior of the unit circle. We note that, if a pole of $H_c(s)$ is on the left-half s-plane, its image in the z-plane will be inside the unit circle. Therefore, causal and stable continuous-time filters will yield stable and causal discrete-time filters, that is, *the bilinear transformation preserves stability.*
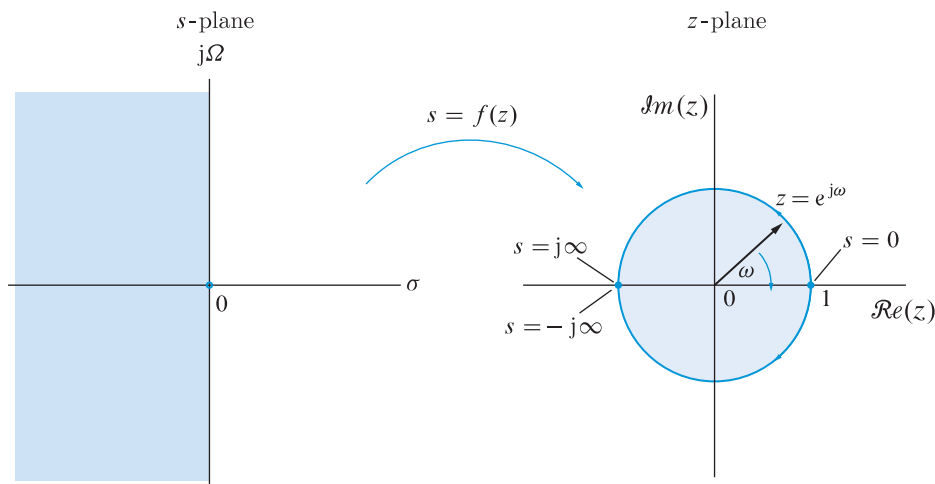
Figure 11.23 Mapping of the s-plane onto the z-plane using the bilinear transformation.

Since the continuous-time frequency axis $j\Omega$ is mapped onto the discrete-time frequency circle $e^{j\omega}$, we can derive the exact relationship between the variables ω and Ω by setting $\sigma = 0$ into (11.89b). The result is

$$\omega = 2\tan^{-1}\left(\frac{\Omega T_d}{2}\right) \quad \text{or} \quad \Omega = \frac{2}{T_d}\tan\left(\frac{\omega}{2}\right), \tag{11.90}$$

which shows that

$$\begin{aligned} \Omega &= 0 &&\Rightarrow& \omega &= 0 \\ \Omega &\to \infty &&\Rightarrow& \omega &\to \pi \\ \Omega &\to -\infty &&\Rightarrow& \omega &\to -\pi. \end{aligned} \tag{11.91}$$

Therefore, the origin of the s-plane is mapped onto the point $z = 1$ of the z-plane, the positive $j\Omega$ axis onto the upper unit circle, and the negative $j\Omega$ onto the lower unit circle of the z-plane. This mapping is illustrated in Figure 11.23.

Since the entire $j\Omega$ axis of the s-plane is mapped onto the unit circle in the z-plane, the bilinear transformation avoids the aliasing problems inherent in the impulse-invariance transformation. However, the highly nonlinear relation between ω and Ω, which is illustrated in Figure 11.24, imposes some critical restrictions on the use of bilinear transformation in certain situations.

Frequency warping The nonlinear relationship between ω and Ω in (11.90) is known as *frequency warping*. The bilinear transformation converts $H_c(j\Omega)$ to $H(e^{j\omega})$ by compressing the continuous-time frequency axis according to (11.91). To understand the implications of frequency warping, we look carefully at Figure 11.25:

1. We note that for ω less than about 0.3π, the relation between Ω and ω is approximately linear (recall that $\tan\phi \approx \phi$ for small ϕ). Thus, any shape of magnitude response in this range is preserved.
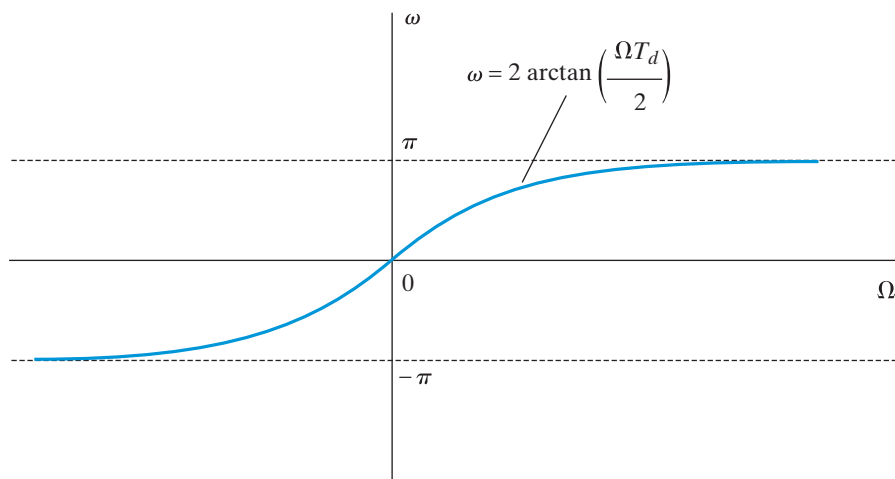
Figure 11.24 The relation established between the discrete-time frequency ω and the continuous-time frequency Ω by the bilinear transformation.

2. Frequency warping does not distort "flat" magnitude responses because any "rearrangement" of equal values will preserve the flatness of the shape. Thus, equiripple bands are mapped into equiripple bands.
3. Frequency warping distorts "non flat" magnitude responses. This is evident in Figure 11.25 by the transformation of the linearly increasing magnitude response of the third band. Thus, a continuous-time differentiator cannot be converted to a discrete-time one using the bilinear transformation.
4. Frequency warping distorts the location of frequency bands and their width. Thus, to design a discrete-time filter with specified band edges ω_k, we determine the design specifications for the continuous-time filter by *prewarping* the frequencies ω_k according to

$$\Omega_k = \frac{2}{T_{\mathrm{d}}} \tan \frac{\omega_k}{2}. \tag{11.92}$$

The subsequent application of bilinear transformation "unwarps" the frequencies Ω_k to the correct specifications ω_k by

$$\omega_k = 2 \tan^{-1} \frac{\Omega_k T_{\mathrm{d}}}{2}. \tag{11.93}$$

The consecutive application of (11.92) and (11.93) during the design process cancels the effect of T_{d}. Thus, we are free to choose a convenient value, such as $T_{\mathrm{d}} = 2$.
5. The phase response of continuous-time filters is affected in a similar manner.

We conclude from the previous discussion that the bilinear transformation is most appropriate for filters with piecewise-constant magnitude responses, such as lowpass, highpass, and bandpass filters. It does not preserve responses with linear magnitude or linear phase
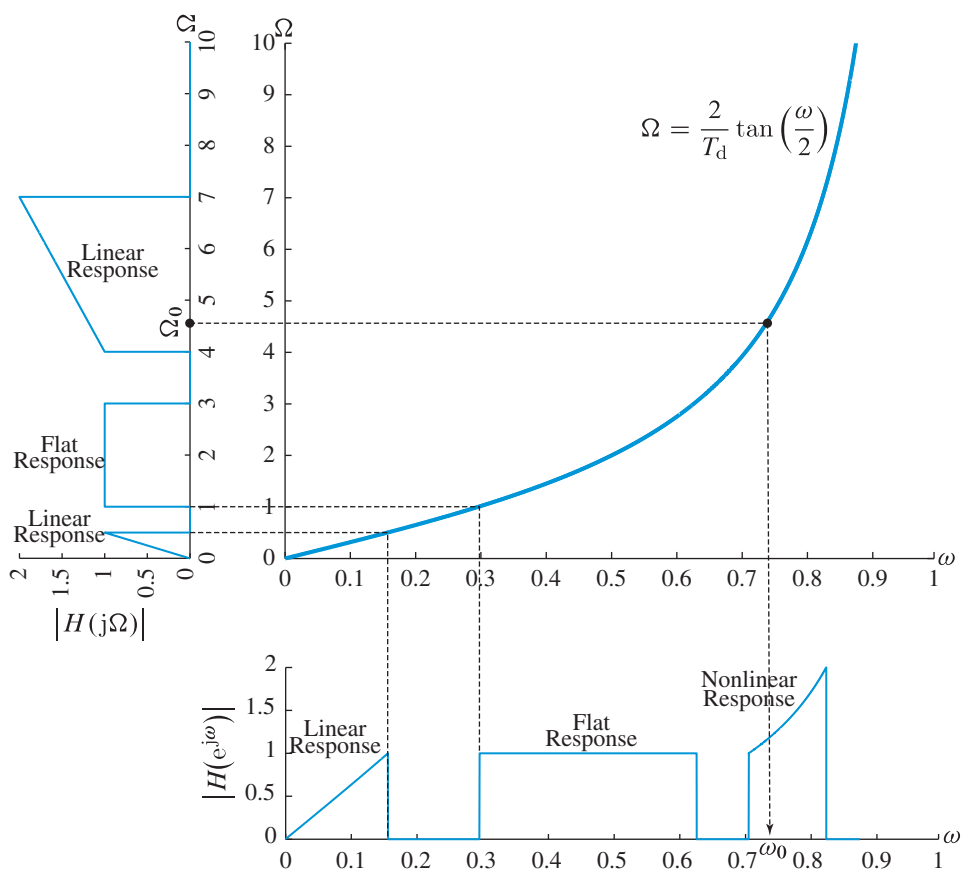
Figure 11.25 The effects of bilinear transformation on the characteristics of magnitude response; similar effects apply to the phase response.

characteristics; therefore, bilinear transformation should not be used to design wideband discrete-time differentiators and linear-phase filters.

Design procedure Suppose we wish to design a digital lowpass filter $H(z)$ specified by the parameters ω_p, A_p, ω_s, and A_s (see Figure 10.1). We start by choosing the design sampling interval $T_d = 2$ as discussed above and then using (11.92) we map ω_p into $\Omega_p = \tan(\omega_p/2)$ and ω_s into $\Omega_s = \tan(\omega_s/2)$. Next, we design the equivalent analog filter $H_c(s)$ using any one of the four analog filter approximations given in Section 11.2 that satisfy the specifications Ω_p, A_p, Ω_s, and A_s. We determine zeros, poles, and gain of $H_c(s)$ and map these quantities into the corresponding zeros, poles, and gain using (11.87). Finally, we assemble the desired digital filter system function $H(z)$ using (11.86). Alternatively, we can use the `bilinear` function in MATLAB. We illustrate this design procedure in Examples 11.14 and 11.15. It should be noted that MATLAB provides several functions, one for each prototype, to obtain digital lowpass filter designs via the bilinear transformation. We discuss and use these functions in Section 11.6.

Example 11.14   Bilinear transformation – Butterworth

We want to design a lowpass digital Butterworth filter to satisfy specifications:

$$\text{Passband edge: } \omega_p = 0.25\pi \text{ rad,} \qquad \text{Passband ripple: } A_p = 1 \text{ dB,}$$

$$\text{Stopband edge: } \omega_s = 0.4\pi \text{ rad,} \qquad \text{Stopband attenuation: } A_s = 30 \text{ dB.}$$

We illustrate the design procedure using the following steps:

Step-1 Choose design parameter T_d. Let $T_d = 2$.

Step-2 Compute the equivalent analog filter band edge frequencies. Using (11.92), we obtain:

$$\Omega_p = \tan(0.25\pi/2) = 0.4142 \quad \text{and} \quad \Omega_s = \tan(0.35\pi/2) = 0.7265.$$

Step-3 Design the analog lowpass filter $H_c(s)$. Using the MATLAB script

```
>> [N,Omegac] = buttord(0.4142,0.7265,1,30,'s'); N
N =
     8
>> [C,D] = butter(N,Omegac,'s');
```

we obtain an eighth-order Butterworth approximation which is one less than that using the impulse-invariance transformation.

Step-4 Transform $H(s)$ into $H(z)$ using MATLAB. Using the script

```
>> [B,A] = bilinear(C,D,1/Td);
```

we obtain the desired digital filter $H(z) = B(z)/A(z)$ using the coefficients in the arrays B and A.

Figure 11.26 shows design plots of the resulting digital Butterworth filter. The magnitude response specifications are met exactly at $\omega_s = 0.4\pi$ but are exceeded at $\omega_p = 0.25\pi$. The impulse responses plot shows that the impulse response $h_c(t)$ of the analog filter (dashed line) and samples of $h[n]$, scaled by $1/T_d$, do not agree although they are close in values. Clearly, the bilinear transformation does not preserve the shape of the impulse response of the analog prototype.   ■

Example 11.15   Bilinear transformation – Chebyshev II

Consider again specifications given in Example 11.14. We want to design the digital filter using the Chebyshev II approximation. The MATLAB script

```
>> omegap = 0.25*pi; omegas = 0.4*pi; Ap = 1; As = 30;
>> Td = 2; Omegap = tan(omegap/2); Omegas = tan(omegas/2);
```
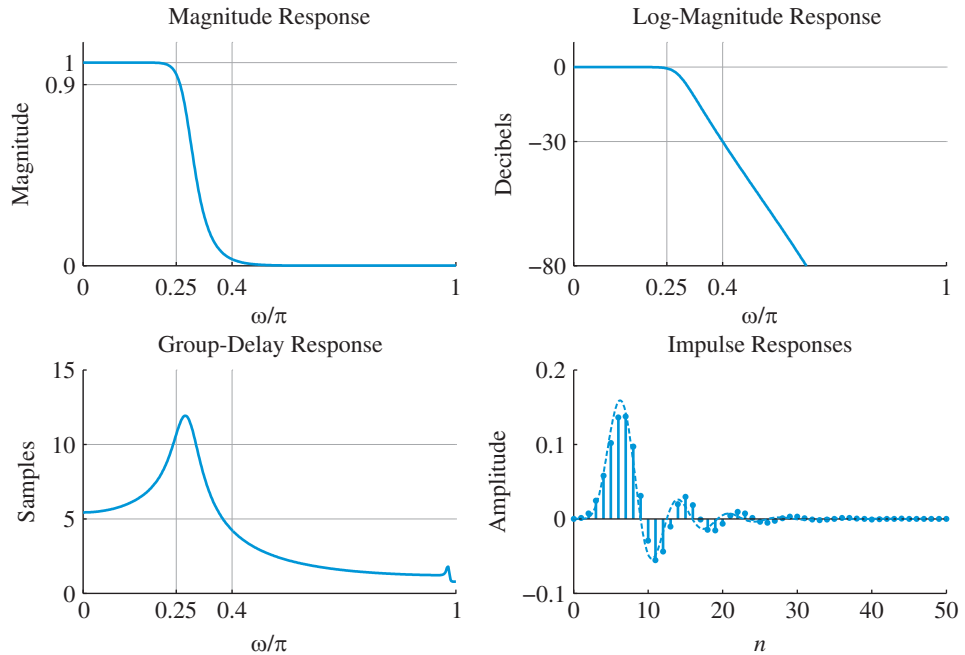
Figure 11.26 Design plots for the eighth-order digital lowpass Butterworth filter in Example 11.14 with specifications $\omega_p = 0.25\pi$, $\omega_s = 0.4\pi$, $A_p = 1$ dB, and $A_s = 30$ dB.

```
>> [N,Omegac] = cheb2ord(Omegap,Omegas,Ap,As,'s'); N
N =
    5
>> [C,D] = cheby2(N,As,Omegac,'s'); [B,A] = bilinear(C,D,1/Td);
```

designs a fifth-order Chebyshev II digital filter. Figure 11.27 shows design plots of the resulting filter. In contrast to the impulse-invariance mapping, the bilinear transformation provides a digital filter that satisfies the given specifications. The magnitude response specifications are met exactly at $\omega_p = 0.25\pi$ and are exceeded at $\omega_s = 0.4\pi$. The equiripple behavior of the digital filter in the stopband is evident in magnitude and log-magnitude responses. The group-delay response is better than that of the Butterworth in Figure 11.26. The impulse responses plot, however, does show that the impulse response $h_c(t)$ of the analog filter (dashed line) and samples of $h[n]$, scaled by $1/T_d$, do not agree although they are close in value.   ■

11.4   Design examples for lowpass IIR filters

We now provide representative examples of more realistic lowpass digital filter designs. To obtain these designs we first describe MATLAB's IIR lowpass filter design functions
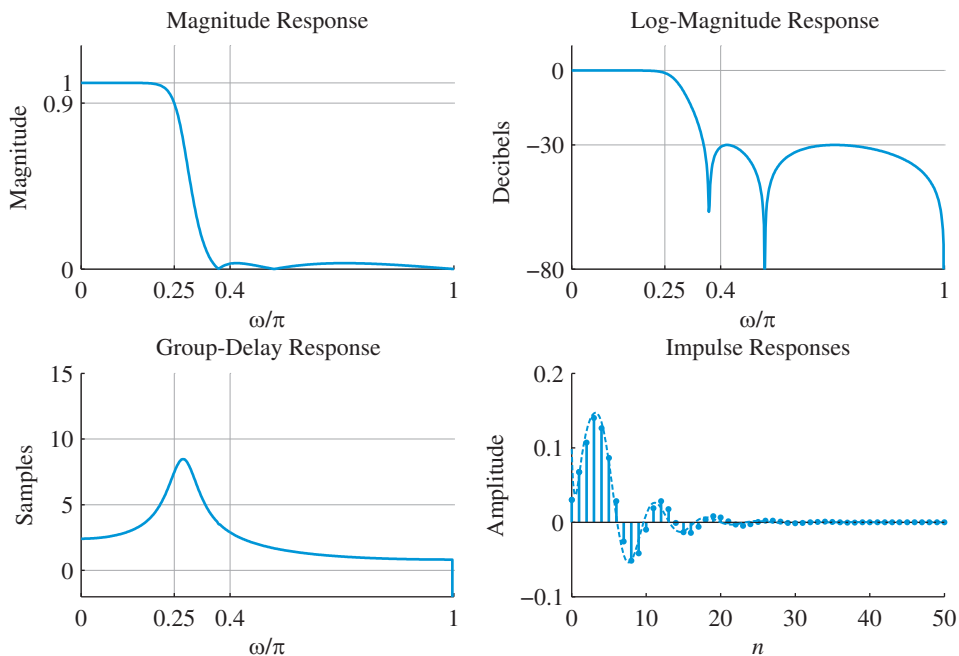
Figure 11.27 Design plots for the fifth-order digital lowpass Chebyshev II filter in Example 11.15 with specifications $\omega_p = 0.25\pi$, $\omega_s = 0.4\pi$, $A_p = 1$ dB, and $A_s = 30$ dB.

that combine the analog approximations described in Section 11.2 and the bilinear transformation which, as previously discussed, can design any piecewise-constant magnitude multiband filters. Below we design digital filters using each of the analog approximations.

Let the digital filter be specified by the parameters: ω_p, A_p, ω_s, and A_s. Let MATLAB parameter omegap contain ω/π, Ap contain A_p, omegas contain ω_s/π, and As contain A_s. The following design functions from the SP toolbox are the same that we used for analog filter designs. However, note carefully in their descriptions that their input *does not* contain the 's' argument.

Butterworth The functions

```
>> [N,omegac] = buttord(omegap,omegas,Ap,As);
>> [B,A] = butter(N,omegac);
```

design an Nth-order lowpass digital Butterworth filter and return the filter coefficients in length $N + 1$ arrays B and A. The variable omegac contains the digital cutoff frequency in units of π at which the log-magnitude response is down to 3 dB.

Chebyshev I The functions

```
>> [N,omegac] = chb1ord(omegap,omegas,Ap,As);
>> [B,A] = cheby1(N,Ap,omegac);
```

design an Nth-order lowpass digital Chebyshev I filter and return the filter coefficients in length $N + 1$ arrays B and A. The variable omegac contains the digital passband edge frequency in units of π at which the log-magnitude response is down to A_p.

Chebyshev II The functions

```
>> [N,omegac] = chb2ord(omegap,omegas,Ap,As);
>> [B,A] = cheby2(N,As,omegac);
```

design an Nth-order lowpass digital Chebyshev II filter and return the filter coefficients in length $N + 1$ arrays B and A. The variable omegac contains the *exact* digital stopband edge frequency in units of π at which the log-magnitude response is down to A_s.

Elliptic The functions

```
>> [N,omegac] = ellipord(omegap,omegas,Ap,As);
>> [B,A] = ellip(N,Ap,As,omegac);
```

design an Nth-order lowpass digital elliptic filter and return the filter coefficients in length $N + 1$ arrays B and A. The variable omegac contains the digital passband edge frequency in units of π at which the log-magnitude response is down to A_p.

Example 11.16   Digital Butterworth lowpass filter

Consider the design of a digital lowpass filter specified by

$$\omega_p = 0.2\pi, \quad A_p = 0.1 \text{ dB}, \quad \omega_s = 0.3\pi, \quad A_s = 50 \text{ dB}. \tag{11.94}$$

To obtain a Butterworth prototype design we use the following MATLAB script:

```
>> omegap = 0.2; Ap = 0.1; omegas = 0.3; As = 50;
>> [N,omegac] = buttord(omegap,omegas,Ap,As)
N =
    17
omegac =
    0.2218
>> [B,A] = butter(N,omegac);
```

The designed filter is 17th-order and the 3 dB cutoff frequency is 0.2218π radians. Figure 11.28 shows magnitude, log-magnitude, group-delay, and impulse response plots of the designed filter. As expected, the filter satisfies all design parameters. ■

Example 11.17   Digital Chebyshev I lowpass filter

Consider the design of a digital lowpass filter specified by

$$\omega_p = 0.3\pi, \quad A_p = 0.5 \text{ dB}, \quad \omega_s = 0.4\pi, \quad A_s = 60 \text{ dB}. \tag{11.95}$$
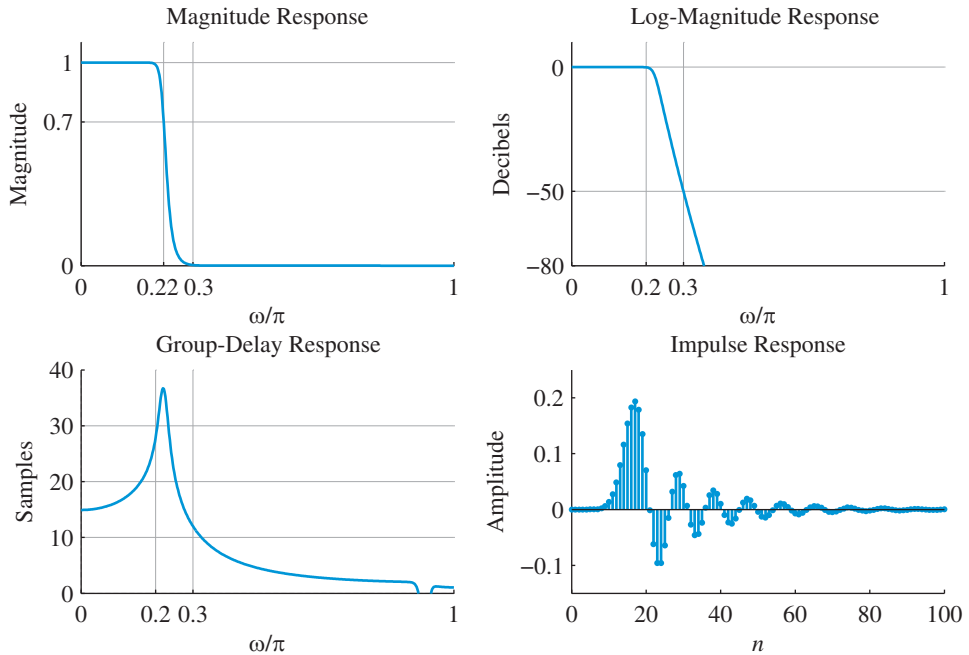
Figure 11.28 Design plots for the 17th-order digital Butterworth lowpass filter in Example 11.16 with specifications $\omega_p = 0.2\pi$, $\omega_s = 0.3\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB.

To obtain a Chebyshev I prototype design we use the following MATLAB script:

```
>> omegap = 0.3; Ap = 0.5; omegas = 0.4; As = 60;
>> [N,omegac] = cheb1ord(omegap,omegas,Ap,As)
N =
    10
omegac =
    0.3000
>> [B,A] = cheby1(N,Ap,omegac);
```

The designed filter is tenth-order and the 0.5 dB cutoff frequency is 0.3π radians as given. Figure 11.29 shows magnitude, log-magnitude, group-delay, and impulse response plots of the designed filter. The filter satisfies all design parameters. ∎

Example 11.18 Digital Chebyshev II lowpass filter

Consider the design of a digital lowpass filter specified by

$$\omega_p = 0.35\pi, \quad A_p = 0.1 \text{ dB}, \quad \omega_s = 0.45\pi, \quad A_s = 60 \text{ dB}. \tag{11.96}$$

To obtain a Chebyshev II prototype design we use the following MATLAB script:
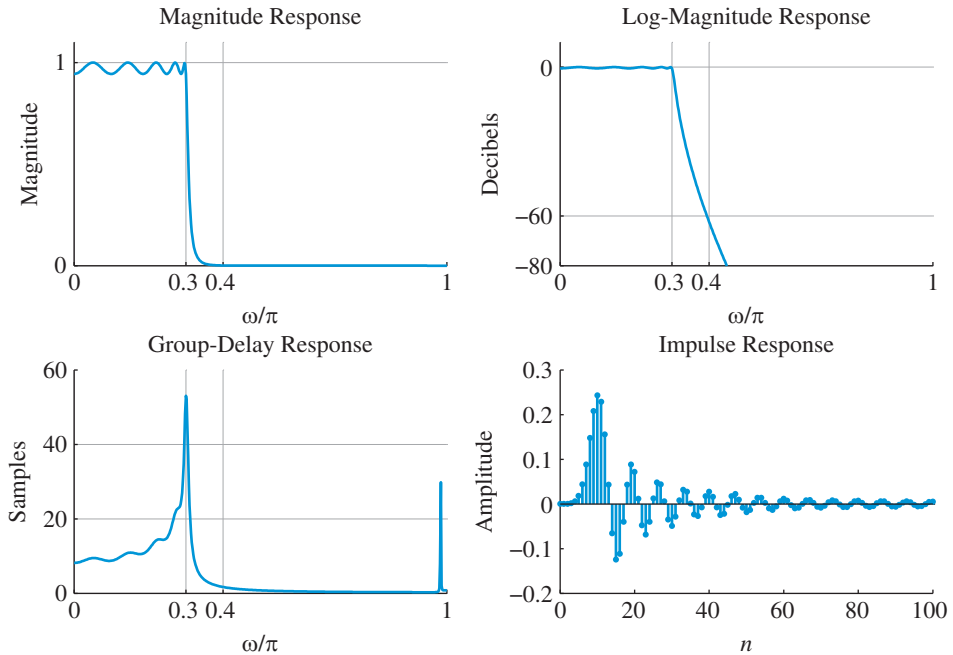
Figure 11.29 Design plots for the tenth-order digital Chebyshev I lowpass filter in Example 11.17 with specifications $\omega_p = 0.3\pi$, $\omega_s = 0.4\pi$, $A_p = 0.5$ dB, and $A_s = 60$ dB.

```
>> omegap = 0.35; Ap = 0.1; omegas = 0.45; As = 60;
>> [N,omegac] = cheb2ord(omegap,omegas,Ap,As)
N =
    12
omegac =
    0.4500
>> [B,A] = cheby2(N,As,omegac);
```

The designed filter is 12th-order and the log-magnitude response at 60 dB is exactly satisfied at 0.45π radians as given. Figure 11.30 shows magnitude, log-magnitude, group-delay, and impulse response plots of the designed filter. The filter satisfies all design parameters exactly. ∎

Example 11.19   Digital elliptic lowpass filter

Consider the design of a digital lowpass filter specified by

$$\omega_p = 0.4\pi, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.55\pi, \quad A_s = 80 \text{ dB}. \tag{11.97}$$

To obtain an ellptic prototype design we use the following MATLAB script:

```
>> omegap = 0.4; Ap = 1; omegas = 0.55; As = 80;
```
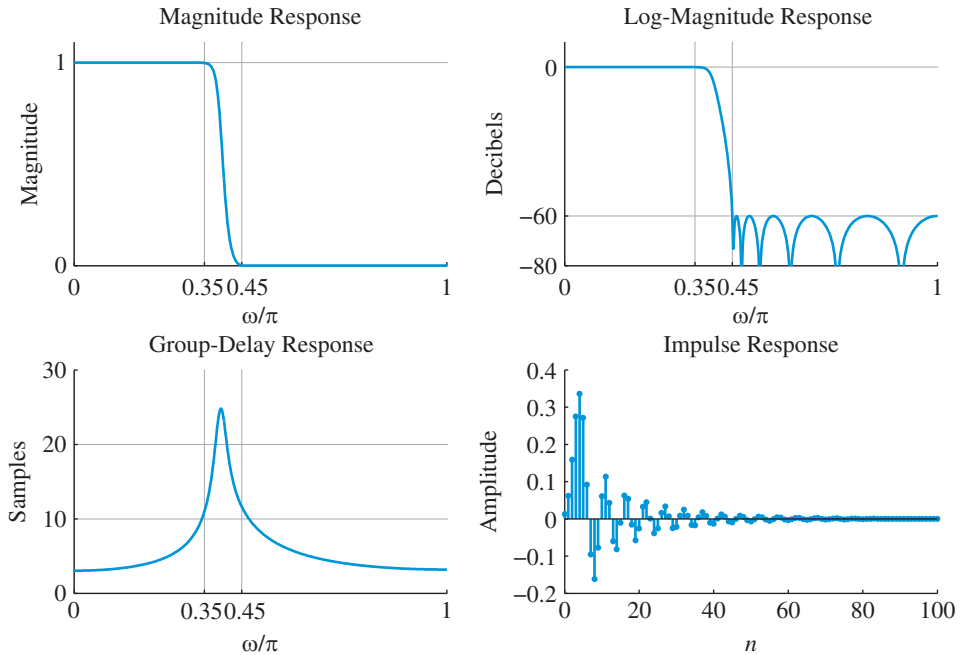
Figure 11.30 Design plots for the 12th-order digital Chebyshev II lowpass filter in Example 11.18 with specifications $\omega_p = 0.35\pi$, $\omega_s = 0.45\pi$, $A_p = 0.1$ dB, and $A_s = 60$ dB.

```
>> [N,omegac] = ellipord(omegap,omegas,Ap,As)
N =
     7
omegac =
    0.4000
>> [B,A] = ellip(N,As,omegac);
```

The designed filter is seventh-order and the 1 dB cutoff frequency is 0.4π radians as given. Figure 11.31 shows magnitude, log-magnitude, group-delay, and impulse response plots of the designed filter. The filter satisfies all design parameters. ∎

11.5   Frequency transformations of lowpass filters

In the preceding sections we have discussed the design of continuous-time and discrete-time lowpass filters. In this section we discuss *frequency (band) transformations* for conversion of lowpass filters to highpass, bandpass, or bandstop filters. Frequency transformations may be performed on either continuous-time or discrete-time designs. The choice depends on the following considerations:
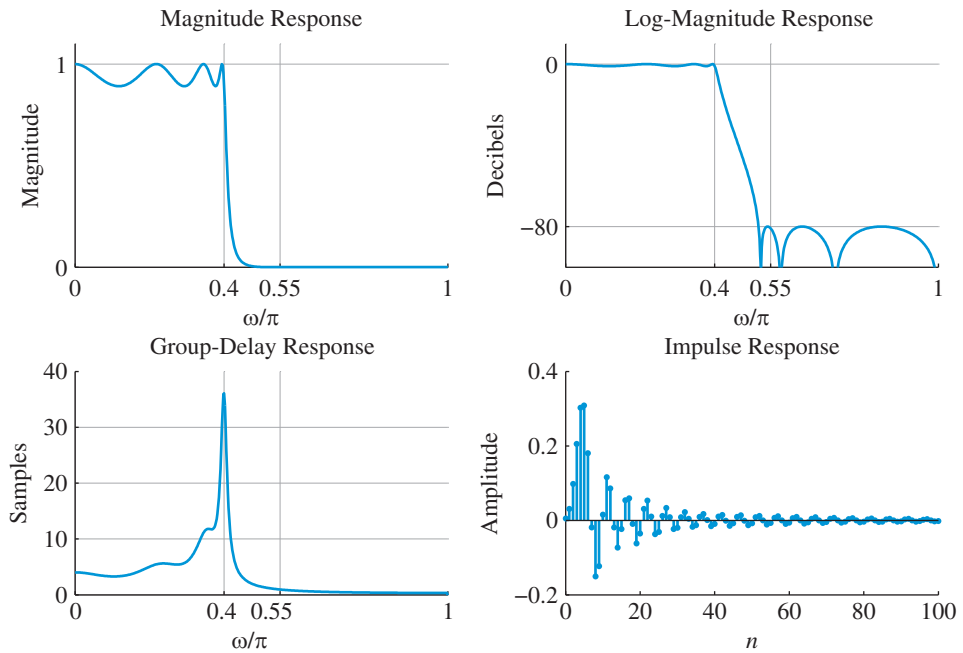
Figure 11.31 Design plots for the seventh-order digital elliptic lowpass filter in Example 11.19 with specifications $\omega_p = 0.4\pi$, $\omega_s = 0.55\pi$, $A_p = 1$ dB, and $A_s = 80$ dB.

- If we use the bilinear transformation to convert a continuous-time to a discrete-time filter, the frequency transformation may be performed either before or after the bilinear transformation, whichever is more convenient.
- If we use the impulse-invariance transformation, the frequency transformation should be performed after we have obtained the discrete-time lowpass filter. This approach avoids the aliasing caused by application of impulse-invariance transformation to highpass and bandstop filters.

We focus on discrete-time frequency transformations because they apply in both cases; however, we provide a brief description of continuous-time frequency transformations for completeness.

11.5.1   Continuous-time frequency transformations

Frequency transformations for continuous-time filters have been studied extensively in analog filter design textbooks, see for example Guillemin (1957) and Lam (1979). Table 11.1 summarizes a typical set of frequency transformations, which transform a normalized prototype lowpass filter $H_c(s)$ with cutoff frequency $\Omega_c = 1$ rad/s to filters with cutoff frequencies Ω_c, Ω_1, and Ω_2, as indicated in the right hand column. We note that the lowpass to bandpass, and lowpass to bandstop transformations double the order of the filter. In MATLAB, these transformations are implemented by the functions

Table 11.1 Continuous-time frequency transformations of lowpass prototype filters with normalized cutoff frequency $\Omega_c = 1$ rad/s.

Filter type	Transformation	Design parameters
Lowpass	$s \to \dfrac{s}{\Omega_c}$	$\Omega_c = $ cutoff frequency
Highpass	$s \to \dfrac{\Omega_c}{s}$	$\Omega_c = $ cutoff frequency
Bandpass	$s \to \dfrac{s^2 + \Omega_1 \Omega_2}{s(\Omega_2 - \Omega_1)}$	$\Omega_1 = $ lower cutoff frequency
		$\Omega_2 = $ upper cutoff frequency
Bandstop	$s \to \dfrac{s(\Omega_2 - \Omega_1)}{s^2 + \Omega_1 \Omega_2}$	$\Omega_1 = $ lower cutoff frequency
		$\Omega_2 = $ upper cutoff frequency

`lp2lp`, `lp2hp`, `lp2bp`, and `lp2bs`. In Example 11.20 we illustrate the use of the `lp2bp` function.

Example 11.20   LP to BP

Let us say that we want to design an analog bandpass filter with lower passband edge of $F_{p1} = 20$ Hz, upper passband edge of $F_{p2} = 60$ Hz with passband ripple of $A_p = 1$ dB and stopband attenuation of $A_s = 40$ dB. The frequency-band transformation function `lp2bl` is invoked by

$$[\texttt{B,A}] = \texttt{lp2bp(C,D,Omega0,BW)},$$

which obtains the numerator and denominator polynomial coefficients of the bandpass filter system function in arrays `B` and `A` respectively, given the numerator and denominator coefficients of a *unity bandwidth* analog lowpass filter in arrays `C` and `D`, the center frequency of the bandpass filter in `Omega0` and bandwidth in `BW`.

Thus we have first to design a lowpass filter with a passband edge of 1 rad/s satisfying the given magnitude specifications. Since stopband edge frequencies of the bandpass filter are not given, we are free to choose the stopband edge of the lowpass filter. In practice, when upper and lower stopband edge frequencies are available we use the mapping function from Table 11.1 to determine the lowpass edge frequency. Hence for this example we choose $\Omega_s = 1.5$ rad/s to obtain a reasonable order. We also choose the elliptic approximation for the lowest order. The required design steps are provided in the following MATLAB script:

```
>> Fp1 = 20; Fp2 = 60; Ap = 1; As = 40; % Given Specifications
>> % Design a Unity Bandwidth Lowpass Elliptic Filter
>> Omegap = 1; Omegas = 1.5;
>> [N,Omegac] = ellipord(Omegap,Omegas,Ap,As,'s'); N
N =
    5
```
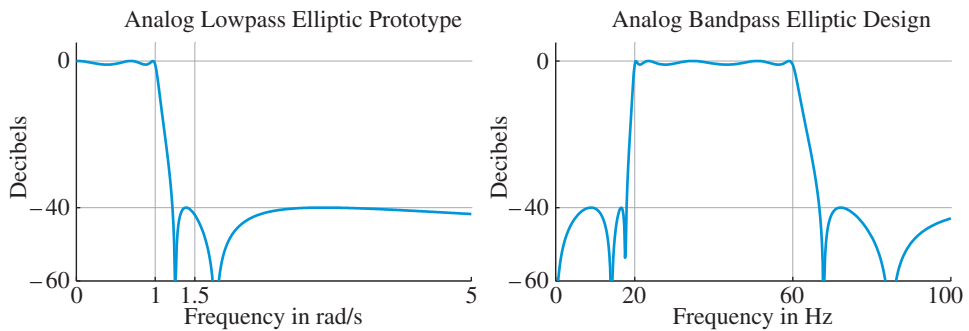
Figure 11.32 Log-magnitude plots for the fifth-order analog elliptic lowpass filter prototype and the transformed tenth-order analog elliptic bandpass filter in Example 11.20.

```
>> % Lowpass to Bandpass Transformation using lp2bp
>> Omega0 = sqrt(Omegap1*Omegap2); BW = Omegap2-Omegap1;
>> [B,A] = lp2bp(C,D,Omega0,BW); Nbp = length(A)-1
Nbp =
    10
```

Note that the designed lowpass filter is fifth-order while the transformed bandpass filter is tenth-order. Also note that the center frequency `Omega0` of the bandpass filter is obtained by taking the geometric mean of the two passband edge frequencies. Figure 11.32 shows log-magnitude response plots in dB for the two analog filters. The band transformation function `lp2bp` has obtained the desired analog bandpass filter. The exact stopband edge frequency resulting from the lowpass elliptic design is 1.22 rad/s (from the plot) which is then transformed to the lower stopband edge of 18 Hz and the upper stopband edge of 66.7 Hz for the bandpass filter. ∎

This example illustrates the design philosophy required to perform frequency band transformations in the analog domain. In the next section we provide more details on this approach using digital frequency band transformations.

11.5.2   Discrete-time frequency transformations

Suppose that we want to convert a prototype discrete-time lowpass filter $H_{\text{lp}}(w)$ with cutoff frequency θ_c into a highpass, bandpass, or bandstop filter $H(z)$ by using a frequency band transformation in the complex plane. To avoid confusion we denote by $w = q e^{j\theta}$ the independent variable in $H_{\text{lp}}(w)$ (design domain) and by $z = r e^{j\omega}$ the independent variable in $H(z)$ (target domain). We seek an algebraic transformation

$$w^{-1} = G(z^{-1}) \tag{11.98}$$

that replaces w^{-1} everywhere in $H_{\text{lp}}(w)$ by the function $G(z^{-1})$ to obtain a new system function given by

$$H(z) = H_{\text{lp}}(w)\Big|_{w^{-1}=G(z^{-1})}. \tag{11.99}$$

A useful frequency transformation should satisfy the following constraints:

1. $G(z^{-1})$ should be a rational function of z^{-1}. This ensures that a rational $H_{\text{lp}}(w)$ will be transformed into a rational $H(z)$.
2. The interior of the unit circle in the w-plane must map to the interior of the unit circle in the z-plane. This requirement preserves stability.
3. The unit circle of the w-plane must map onto the unit circle of the z-plane.

The last condition requires that

$$e^{-j\theta} = |G(e^{-j\omega})|e^{j\angle G(e^{-j\omega})}, \tag{11.100}$$

which implies that $G(z^{-1})$ should be an allpass function, that is,

$$|G(e^{-j\omega})| = 1. \tag{11.101}$$

The relationship between the design and target domain frequency variables is

$$-\theta = \angle G(e^{-j\omega}). \tag{11.102}$$

For filters with real coefficients, Constantinides (1970) showed that the most general allpass system (see Section 5.9) that satisfies the desired constraints is

$$w^{-1} = G(z^{-1}) = \pm \prod_{k=1}^{N} \frac{z^{-1} - \alpha_k^*}{1 - \alpha_k z^{-1}}, \tag{11.103}$$

where $|\alpha_k| < 1$ (see Problem 36). By properly choosing the values of N and α_k, we can obtain a variety of mappings. The most useful frequency transformations, derived by Constantinides (1970), are shown in Table 11.2. We note that the design parameters are functions only of the frequency variables corresponding to the various cutoff frequencies. These transformations can be performed either by direct substitution of $w = G(z^{-1})$ into $H_{\text{lp}}(w)$ or by calculating the target pole and zero locations (see Problem 62).

To illustrate the process we consider the lowpass to highpass frequency transformation. For this case, from Table 11.2 we have

$$w^{-1} = G(z^{-1}) = -\frac{z^{-1} + \alpha}{1 + \alpha z^{-1}}, \tag{11.104}$$

where $-1 < \alpha < 1$. Substituting $w = e^{j\theta}$ and $z = e^{j\omega}$, we obtain

$$e^{-j\theta} = -\frac{e^{-j\omega} + \alpha}{1 + \alpha e^{-j\omega}}, \tag{11.105}$$

which yields

$$\omega = \tan^{-1}\left[-\frac{(1-\alpha^2)\sin\theta}{2\alpha + (1+\alpha^2)\cos\theta}\right], \tag{11.106a}$$

$$\theta = \tan^{-1}\left[-\frac{(1-\alpha^2)\sin\omega}{2\alpha + (1+\alpha^2)\cos\omega}\right]. \tag{11.106b}$$

Table 11.2 Transformations from a discrete-time lowpass filter protoptype with cutoff frequency θ_c to highpass, bandpass, and bandstop filters.

Filter type	Transformation	Design parameters
Lowpass	$z^{-1} \rightarrow \dfrac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$	$\omega_c = \text{cutoff frequency}$ $\alpha = \dfrac{\sin[(\theta_c - \omega_c)/2]}{\sin[(\theta_c + \omega_c)/2]}$
Highpass	$z^{-1} \rightarrow -\dfrac{z^{-1} + \alpha}{1 + \alpha z^{-1}}$	$\omega_c = \text{cutoff frequency}$ $\alpha = -\dfrac{\cos[(\theta_c + \omega_c)/2]}{\cos[(\theta_c - \omega_c)/2]}$
Bandpass	$z^{-1} \rightarrow -\dfrac{z^{-2} - \alpha_1 z^{-1} + \alpha_2}{\alpha_2 z^{-2} - \alpha_1 z^{-1} + 1}$	$\omega_1 = \text{lower cutoff frequency}$ $\omega_2 = \text{upper cutoff frequency}$ $\alpha = \dfrac{\cos[(\omega_2 + \omega_1)/2]}{\cos[(\omega_2 - \omega_1)/2]}$ $K = \cot[(\omega_2 - \omega_1)/2]\tan(\theta_c/2)$ $\alpha_1 = 2\alpha K/(K + 1)$ $\alpha_2 = (K - 1)/(K + 1)$
Bandstop	$z^{-1} \rightarrow \dfrac{z^{-2} - \alpha_1 z^{-1} + \alpha_2}{\alpha_2 z^{-2} - \alpha_1 z^{-1} + 1}$	$\omega_1 = \text{lower cutoff frequency}$ $\omega_2 = \text{upper cutoff frequency}$ $\alpha = \dfrac{\cos[(\omega_2 + \omega_1)/2]}{\cos[(\omega_2 - \omega_1)/2]}$ $K = \tan[(\omega_2 - \omega_1)/2]\tan(\theta_c/2)$ $\alpha_1 = 2\alpha/(K + 1)$ $\alpha_2 = (1 - K)/(1 + K)$

This relationship, which is plotted in Figure 11.33, shows that the lowpass to highpass transformation causes a nonlinear warping of the frequency scale for $\alpha \neq 0$. As in the bilinear transformation case, this warping does not distort piecewise-constant frequency response characteristics. Solving (11.105) for α, we obtain

$$\alpha = -\frac{\cos[(\theta_c + \omega_c)/2]}{\cos[(\theta_c - \omega_c)/2]}, \tag{11.107}$$

which provides the value of α required to perform the lowpass to highpass frequency transformation given the lowpass and highpass cutoff frequencies, θ_c and ω_c, respectively.

Having developed the basic theory we do not need to continue using two variables w and z. Thus, given a prototype lowpass filter $H_{lp}(z)$ we can directly apply the substitution $z^{-1} \rightarrow G(z^{-1})$ to obtain the desired filter $H(z)$. This suggests the possibility of replacing all delays in a filter structure by appropriate allpass filters to create filters with tunable cutoff frequencies. This process has some pitfalls that are discussed in Problem 37. The application of the lowpass to highpass transformation discussed above is illustrated in the following example.
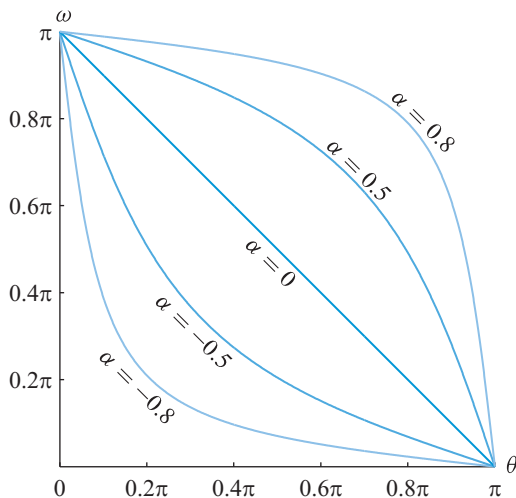
Figure 11.33 Warping of frequency scale in lowpass-to-highpass transformation.

Example 11.21   Lowpass to highpass filter transformation

We want to design a digital highpass filter that satisfies the following specifications:

$$\text{Stopband edge: } \omega_s = 0.45\pi \text{ rad,} \qquad \text{Stopband attenuation: } A_p = 15 \text{ dB,}$$

$$\text{Passband edge: } \omega_p = 0.6\pi \text{ rad,} \qquad \text{Passband ripple: } A_s = 1 \text{ dB.}$$

We use the Chebyshev I approximation. To use the highpass transformation function in Table 11.2 we first need to design an appropriate digital lowpass prototype filter. As we did for the analog frequency-band transformation in Example 11.20, we design the prototype filter with passband cutoff $\theta_p = 1$ radian that can be mapped to $\omega_p = 0.6\pi$ through band transformation. Setting $\theta_c = \theta_p$ and $\omega_c = \omega_p$ in (11.107), we obtain $\alpha = -0.1416$ for the transformation function (11.104). We also need the stopband edge θ_s for the lowpass prototype so as to design it to correspond to ω_s of the highpass filter. Substituting $\omega = \omega_s = 0.45\pi$ in (11.106b) we obtain $\theta_s = 1.4437$ radians. Now we design a lowpass prototype filter with specifications $\theta_p = 1$, $A_p = 1$, $\theta_s = 1.4437$, and $A_s = 15$ using the following MATLAB script:

```
>> omegas = 0.45*pi; omegap = 0.6*pi; Ap = 1; As = 15;
>> thetap = 1; % Lowpass prototype passband cutoff
>> al = -(cos((thetap+omegap)/2))/(cos((thetap-omegap)/2))
>> thetas = atan(-(1-al\^2)*sin(omegas)/...
     (2*al+(1+al\^2)*cos(omegas)));
>> % Lowpass Prototype Filter Design
>> [N,omegac] = cheb1ord(thetap/pi,thetas/pi,Ap,As);
>> [Bp,Ap] = cheby1(N,Ap,omegac)
Bp =
    0.0403    0.1208    0.1208    0.0403
```
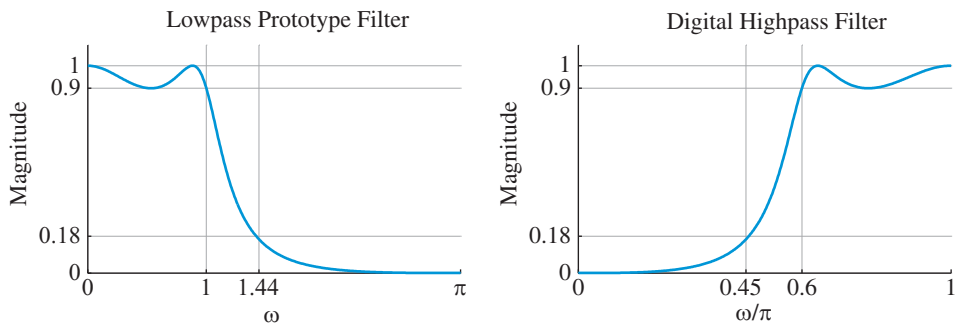
Figure 11.34 Magnitude plots of the third-order digital lowpass prototype filter and the transformed third-order digital passband filter in Example 11.21.

```
Ap =
    1.0000   -1.4726    1.1715   -0.3767
```

Thus the lowpass prototype filter is third-order given by

$$H_{\text{lp}}(z) = \frac{0.0403 + 0.1208z^{-1} + 0.1208z^{-2} + 0.0403z^{-3}}{1 - 1.4726z^{-1} + 1.1715z^{-2} - 0.3767z^{-3}}. \tag{11.108}$$

Finally, we transform the lowpass prototype into the desired highpass digital filter using the function (11.104) with $\alpha = -0.1416$. Hence substituting

$$-\frac{z^{-1} - 0.1416}{1 - 0.1416z^{-1}}$$

for z^{-1} in (11.108) we obtain the desired highpass filter

$$H(z) = \frac{0.0736 - 0.2208z^{-1} + 0.2208z^{-2} - 0.0736z^{-3}}{1 + 0.9761z^{-1} + 0.8568z^{-2} + 0.2919z^{-3}}, \tag{11.109}$$

which completes the design. The book toolbox function z2z is available for the above conversion. Figure 11.34 shows the magnitude responses of the prototype lowpass and the designed highpass digital filters. ∎

The design procedure illustrated in Example 11.21 can also be extended to other frequency-band transformations to obtain bandpass and bandstop and lowpass filters. These procedures are incorporated in MATLAB's familiar digital filter design functions as we shall study in the next section.

11.6   Design examples of IIR filters using MATLAB

As explained in Section 11.4 for lowpass filter designs, MATLAB also provides a two step approach to designing general frequency selective filters using the bilinear

transformation method. Using additional parameters in the invocation of the order determining and designing filters we can design lowpass, highpass, bandpass, or bandstop IIR filters using any one of the analog prototypes. We illustrate this two-step approach using the elliptic prototype since it uses the maximum number of parameters.

Order determination Given digital filter specifications, the `ellipord` function determines filter order according to the following syntax:

$$[\text{N,omegac}]=\text{ellipord(omegap,omegas,Ap,As)},$$

where the frequency parameters, in units of π, have the following interpretations:

- for lowpass filters `omegap < omegas`,
- for highpass filters `omegap > omegas`,
- for bandpass filters `omegap` and `omegas` are two-element vectors given by `omegap=[omegapL,omegapH]` and `omegap=[omegapL,omegapH]` such that `omegasL < omegapL < omegapH < omegasH`,
- for bandstop filters `omegap` and `omegas` are also two-element vectors given by `omegap=[omegapL,omegapH]` and `omegap=[omegapL,omegapH]` such that `omegapL < omegasL < omegasH < omegapH`.

The parameter `omegac` provides the appropriate cutoff frequency (or frequencies) in each case.

Filter design Using the above order determining function, the following invocations of the `ellip` function with varying parameters complete the IIR filter design:

- `[B,A]=ellip(N,Ap,As,omegac)` designs an `N`th-order *lowpass* elliptic filter with `omegac` equal to ω_p/π.
- `[B,A]=ellip(N,Ap,As,omegac,'high')` designs an `N`th-order *highpass* elliptic filter with `omegac` equal to ω_p/π.
- `[B,A]=ellip(N,Ap,As,omegac)` designs a `2N`th-order *bandpass* filter if `omegac=[omegapL,omegapH]` is a two-element vector in units of π.
- `[B,A]=ellip(N,Ap,As,omegac),'stop'` designs a `2N`th-order *bandstop* filter if `omegac=[omegapL,omegapH]` is a two-element vector in units of π.

Using these and similar functions for the Butterworth, Chebyshev I and Chebyshev II filters one can now design any standard frequency selective IIR filter. We provide some representative examples below.

Example 11.22   Butterworth highpass filter design
Consider the following specifications for the design of Butterworth highpass filter:

$$\text{Stopband edge: } \omega_s = 0.5\pi \text{ rad}, \quad \text{Stopband attenuation: } A_p = 40 \text{ dB},$$

$$\text{Passband edge: } \omega_p = 0.7\pi \text{ rad}, \quad \text{Passband ripple: } A_s = 1 \text{ dB}.$$
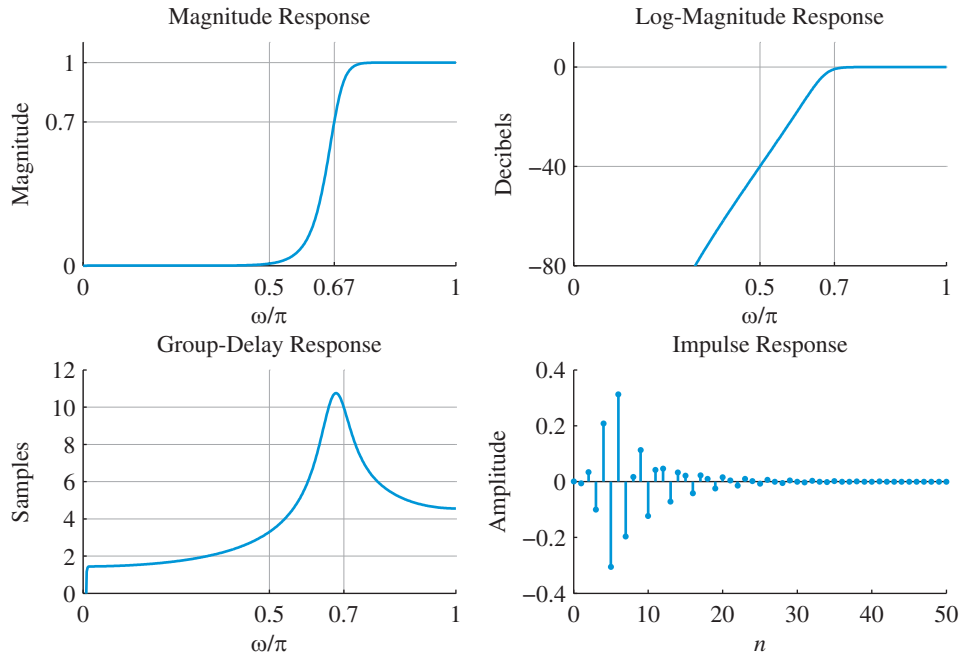
Figure 11.35 Filter design plots of the eighth-order digital Butterworth highpass filter in Example 11.22.

Using the MATLAB script:

```
>> omegap = 0.7; Ap = 1; omegas = 0.5; As = 40;
>> [N,omegac] = buttord(omegap,omegas,Ap,As)
N =
    8
omegac =
    0.6739
>> [B,A] = butter(N,omegac,'high');
```

we obtain an eighth-order filter with 3 dB cutoff frequency of 0.6739 radians. Figure 11.35 shows filter response plots of the designed filter.   ∎

Example 11.23   Chebyshev I bandpass filter design

Consider design of a Chebyshev I bandpass filter using specifications:

$$
\begin{array}{ll}
\text{Lower stopband: } [0, 0.25\pi], & \text{Attenuation: } 40 \text{ dB}, \\
\text{Passband: } [0.3\pi, 0.5\pi], & \text{Ripple: } 0.5 \text{ dB}, \\
\text{Upper stopband: } [0.6\pi, \pi], & \text{Attenuation: } 50 \text{ dB}.
\end{array}
$$
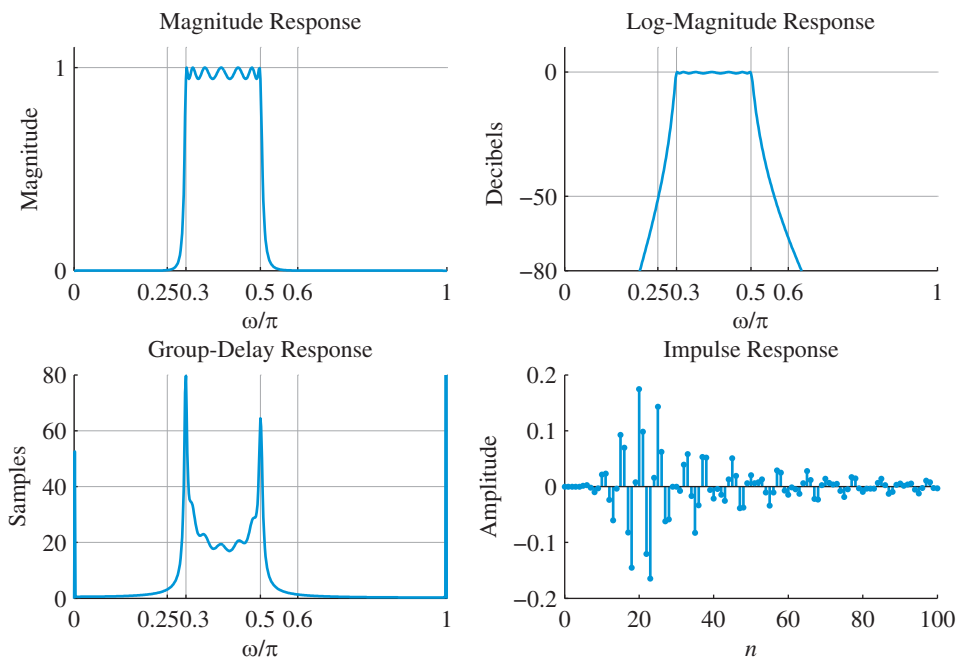
Figure 11.36 Filter design plots of the 14th-order digital Chebyshev I bandpass filter in Example 11.23.

There are two stopbands with two different attenuation values. Since the bandpass filter will be designed by transforming a lowpass filter, the design can accommodate only one attenuation. We design for the higher value. Using the MATLAB script:

```
>> omegap = [0.3,0.5]; omegas = [0.25,0.6]; Ap = 0.5; As = 50;
>> [N,omegac] = cheb1ord(omegap,omegas,Ap,As); N
N =
     7
>> [B,A] = cheby1(N,Ap,omegac);
```

we obtain the desired filter. Note that the `cheb1ord` function reports the order $N = 7$ for the lowpass prototype which means that the resulting bandpass filter is 14th-order. Figure 11.36 shows filter response plots of the bandpass filter. From the log-magnitude response plot, observe that the exact upper and lower stopband edges at which the respective attenuation values are met are 0.265π and 0.563π, respectively. This is due to the integer value of N. The filter has a longer group delay, which means that the impulse response is still significant after 100 samples. ∎

Example 11.24   Chebyshev II highpass bandstop design
The following specifications are for a bandstop filter which we want to design using a Chebyshev II approximation:
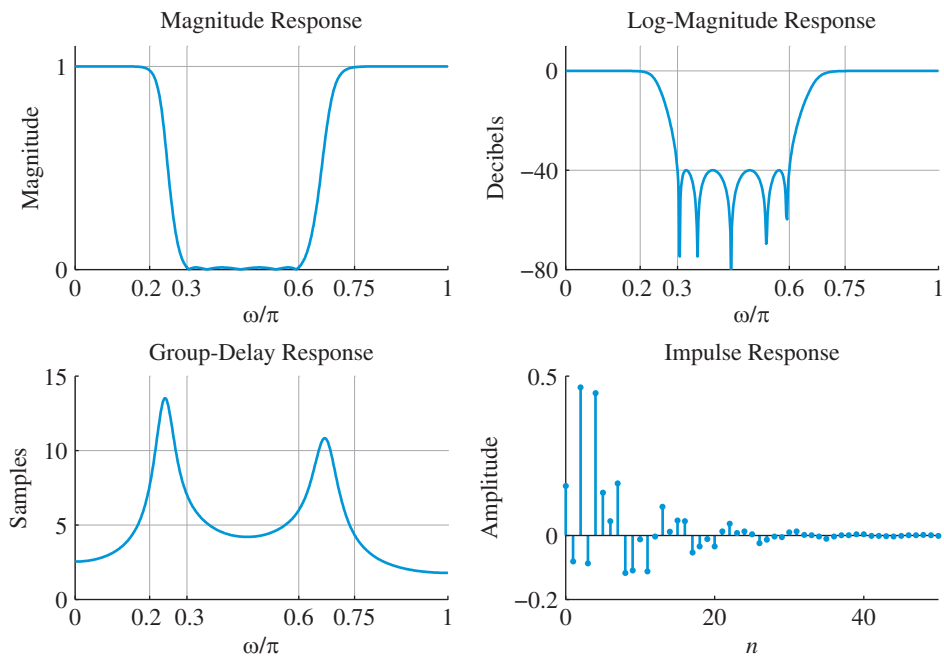
Figure 11.37 Filter design plots of the tenth-order digital Chebyshev II bandstop filter in Example 11.24.

$$\text{Lower passband: } [0, 0.2\pi], \qquad\qquad \text{Ripple: 0.5 dB,}$$
$$\text{Stopband: } [0.3\pi, 0.6\pi], \qquad \text{Attenuation: 40 dB,}$$
$$\text{Upper passband: } [0.75\pi, \pi], \qquad\qquad \text{Ripple: 0.5 dB.}$$

Using MATLAB script:

```
>> omegap = [0.2,0.75]; omegas = [0.3,0.6]; Ap = 0.5; As = 40;
>> [N,omegac] = cheb2ord(omegap,omegas,Ap,As); N
N =
     5
>> [B,A] = cheby2(N,As,omegac,'stop');
```

we obtain a tenth-order bandstop filter with equiripple behavior in the stopband. The filter response plots are shown in Figure 11.37. The exact upper and lower stopband-edge frequencies are 0.21π and 0.71π, respectively. ∎

Example 11.25 Elliptic bandpass filter design

Consider the design of a bandpass filter using an elliptic approximation to satisfy the specifications:
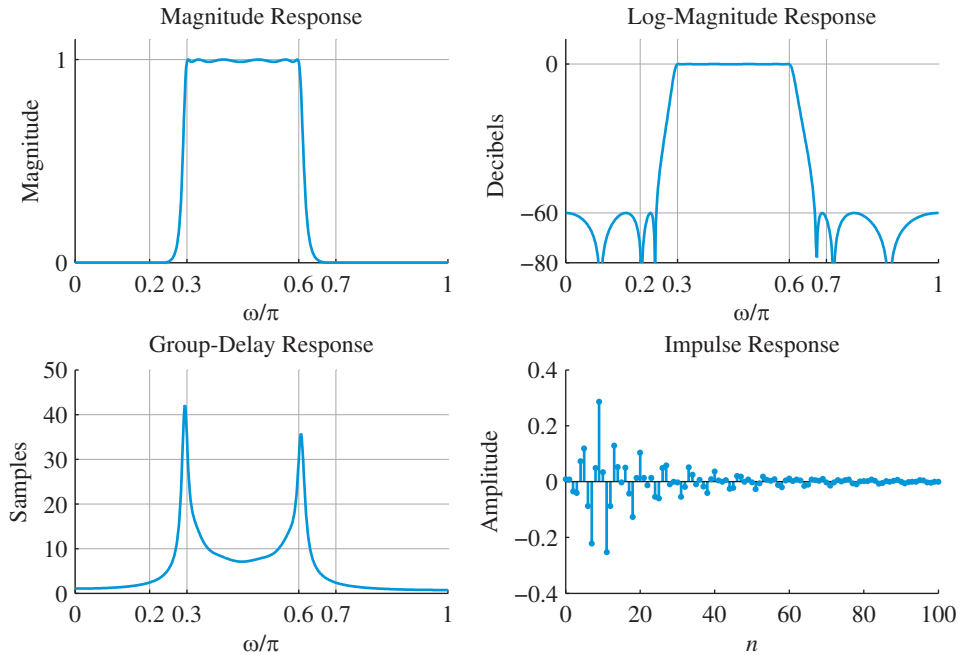
Figure 11.38 Filter design plots of the 12th-order digital elliptic bandpass filter in Example 11.25.

Lower stopband: $[0, 0.2\pi]$,	Attenuation: 60 dB,
Passband: $[0.3\pi, 0.6\pi]$,	Ripple: 0.1 dB,
Upper stopband: $[0.7\pi, \pi]$,	Attenuation: 60 dB.

Using MATLAB script:

```
>> omegap = [0.3,0.6]; omegas = [0.2,0.7]; Ap = 0.1; As = 60;
>> [N,omegac] = ellipord(omegap,omegas,Ap,As); N
N =
     6
>> [B,A] = ellip(N,Ap,As,omegac);
```

we obtain a 12th-order bandpass filter with equiripple behavior in all bands. The filter response plots are shown in Figure 11.38. The exact upper and lower stopband-edge frequencies are 0.243π and 0.668π, respectively.　　　■

FDATool for IIR filter design Finally, to complete our discussion on IIR filter design, we provide brief details on use of the filter analysis and design tool from the SP toolbox which is invoked by the function `fdatool`. The main panel selection-areas of its graphical user interface were described in Chapter 10. Figure 11.39 shows the user interface
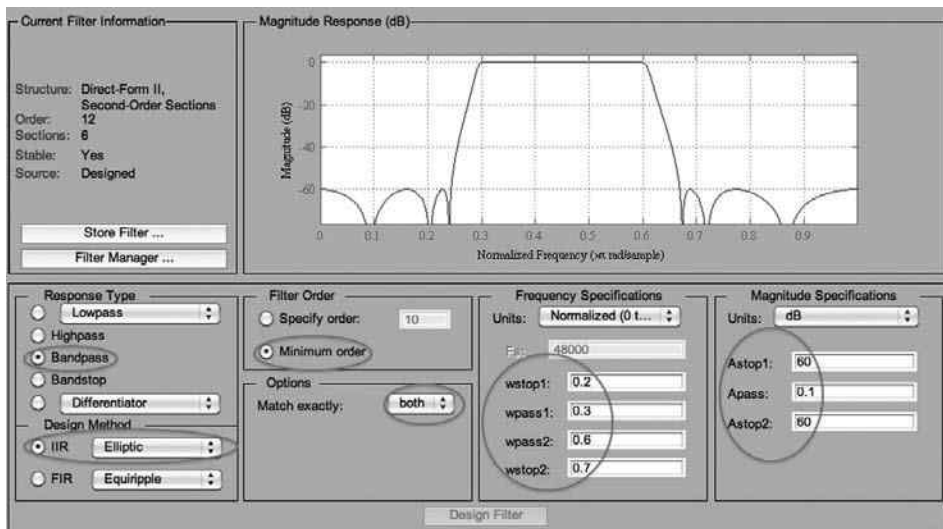
Figure 11.39 FDATool user interface for the design of a digital elliptic bandpass filter used in Example 11.25.

needed to design the bandpass elliptic given in Example 11.25, with several key parameter areas highlighted. Using appropriate radio buttons, drop-down list items, and the required parameter input areas properly entered, we can design and analyze any frequency selective IIR filter using one of the four approximations, which certainly eliminates the command window approach and makes the design exercise very convenient.

In conclusion, by using detailed theoretical explanations and through extensive worked and MATLAB-based design examples it is hoped that we have demystified the intricacies involved in the design of analog and digital frequency-selective filters.

Learning summary

- Practically realizable IIR filters, that is, causal and stable filters with rational system functions have a nonlinear phase response, which complicates filter design using optimization techniques.

- The IIR filter design problem discussed in this chapter consists of obtaining a causal and stable filter with a rational system function, whose frequency response best approximates the desired ideal magnitude responses within specified tolerances while the phase response is left unspecified.

- The most popular techniques for design of IIR filters start with the design of a continuous-time prototype lowpass filter (Butterworth, Chebyshev, or elliptic), which is subsequently converted to a discrete-time filter (lowpass, highpass, bandpass, or bandstop) using an appropriate set of transformations.

- The Butterworth approximation has maximally flat response in both passband and stopband and always produces, for the given specifications, the largest-order filters but having small group-delays across the passband. The Chebyshev approximation has equiripple response in passband (type I) or stopband (type II) filters and hence produces filters with smaller orders than the Butterworth approximation. However, the group-delay response is worse for type I while for type II it is comparable to Butterworth group-delay. The elliptic approximation has equiripple response in both bands and produces the smallest-order filters but with the worst group-delay response.

- The impulse invariance and bilinear mappings are two most popular transformations that convert analog into digital filters. The better and more versatile of the two is bilinear mapping.

- Frequency transformations of the allpass type are used to obtain a general frequency-selective filter from a prototype lowpass filter in the analog as well as the digital domain.

- The most useful techniques for filter design have been implemented on various computational environments, like MATLAB, and they are widely available in the form of filter design packages.

TERMS AND CONCEPTS

Bilinear transformation A one-to-one analog to digital filter transformation that maps analog complex frequency s into digital complex frequency z according to (11.83). It is a versatile mapping that can be used for any constant-magnitude multiband filters and approximations.

Butterworth approximation An analog filter system function which is a Taylor series approximation of the ideal response at a single frequency. The resulting filter has maximally flat response in passband and stopband.

Butterworth filter See Butterworth approximation.

Cauer filter Another name for an elliptic filter which was first introduced by W. Cauer in 1931.

Chebyshev approximation An analog filter system function which is optimum in the minimax sense either over a passband (type I) or stopband (type II) and is based on Chebyshev polynomials. The resulting filter has an equiripple passband (stopband) and monotone stopband (passband).

Chebyshev filter See Chebyshev approximation.

Elliptic approximation An analog filter system function which is optimum in the minimax sense over both passband and stopband and is based on rational Chebyshev functions. The resulting filter has equiripple passband and stopband responses.

Elliptic filter See Elliptic approximation.

Frequency (band) transformation Conversion of lowpass to highpass, bandpass, or bandstop filters which can be performed in either analog or digital domain.

Frequency warping The nonlinear relationship (11.90) between digital frequency ω and the analog frequency Ω in the bilinear transformation.

Impulse-invariance transformation An analog to digital filter transformation that preserves the shape of the analog filter impulse response. Suffers from the frequency-domain aliasing problem, hence suitable for lowpass or bandpass filters using Butterworth or Chebyshev I approximations.

Prewarping Computation of the analog frequency Ω from the digital frequency ω using the frequency warping formula so that the frequency-distortion in bilinear transformation is compensated.

Spectral factorization Separation of the product $H_c(s)H_c(-s)$ into a causal and stable factor $H_c(s)$. A unique solution for a rational product function is a minimum-phase function.

Zero-phase filtering An IIR filtering operation in which a causal and stable filter is applied to the input signal in forward and backward direction to obtain a response with zero delay.

MATLAB functions and scripts

Name	Description	Page
bilinear	Analog to digital filter transformation – bilinear mapping	662
buttord	Butterworth analog/digital filter order computation	632/669
butter	Butterworth analog/digital filter design	632/669
cheb1ord	Chebyshev I analog/digital filter order computation	640/669
cheby1	Chebyshev I analog/digital filter design	641/669
cheb2ord	Chebyshev II analog/digital filter order computation	645/670
cheby2	Chebyshev II analog/digital filter design	645/670
ellipord	Elliptic analog/digital filter order computation	650/670
ellip	Elliptic analog/digital filter design	650/670
ellipke	Complete elliptic integral of the first kind	649
fdatool	GUI-based filter design and analysis tool	685
filtfilt	Zero-phase IIR filtering	627
impinvar	Analog to digital filter transformation – impulse invariance	656
lp2bp	Analog lowpass to bandpass filter transformation	675
lp2bs	Analog lowpass to bandstop filter transformation	675
lp2hp	Analog lowpass to highpass filter transformation	675
lp2lp	Analog lowpass to lowpass filter transformation	675
z2z*	z-plane to z-plane digital filter transformation	680

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A detailed treatment of discrete-time filter design, at the same level as in this book, is given in Oppenheim and Schafer (2010), Proakis and Manolakis (2007), Mitra (2006), and Rabiner and Gold (1975).

2. The books by Parks and Burrus (1987) and Antoniou (2006) place more emphasis on filter design and provide more details regarding the design of FIR (Parks–McClellan) and IIR (elliptic) filters with equiripple responses.

3. An extensive review of filter design techniques, beyond those discussed in this chapter, is provided by Karam *et al.* (2009) and Saramaki (1993).

4. The approximation problem for continuous-time (analog) filters is thoroughly discussed in several textbooks, including Guillemin (1957), Zverev (1967), Daniels (1974), Weinberg (1975), and Lam (1979).

Review questions

1. Describe the common approach that uses three steps to design IIR frequency selective filters.

2. Explain clearly why design approaches used for IIR filters are different from those of the FIR approaches studied in Chapter 10.

3. Which similarities between the digital and analog filter system functions are used in IIR filter design?

4. Which two approaches are used in designing standard frequency selective IIR filters given an analog lowpass filter?

5. IIR filters cannot have linear or zero phase responses. Do you agree or disagree? Explain.

6. Describe advantages of using FIR filters over IIR filters.

7. Describe advantages of using IIR filters over FIR filters.

8. What is a zero-phase IIR filter and how is it implemented?

9. A new start-up company has come out with a product that uses a real-time zero-phase IIR filter. Would you invest in this company?

10. A causal IIR filter generally has a nonlinear phase response. What approach is used to make the phase response linear?

11. Explain the magnitude-squared specifications for a lowpass analog filter.

12. How does one determine the rational system function $H_c(s)$ given its magnitude-squared response?

13. Describe the frequency-domain characteristics of the Butterworth approximation.

14. Why are Butterworth filters called maximally-flat magnitude filters?

15. Explain how poles of the Butterworth filter are distributed in the s-plane and how we determine them.

16. Describe the behavior of the Chebyshev polynomial $T_N(x)$ over $0 \leq |x| \leq 1$ and $|x| > 1$ intervals for a given N.

17. Explain the frequency-domain characteristics of the Chebyshev I and Chebyshev II approximations.
18. Describe how poles of the Chebyshev I approximation are distributed in the *s*-plane and how we compute them.
19. Why does the Chebyshev I approximation produce a worse group-delay response than the Chebyshev II approximation?
20. How is a Chebyshev II approximation obtained from Chebyshev I?
21. Poles of the Chebyshev II approximation are distributed along an ellipse in the *s*-plane. Do you agree or disagree? Explain.
22. Describe the frequency-domain characteristics of the elliptic approximation.
23. State desirable conditions needed to transform an analog into a digital filter and explain the reasons behind each of them.
24. What is the basic principle behind the impulse-invariance transformation?
25. How are analog and digital frequencies as well as the corresponding frequency responses related through the impulse-invariance transformation?
26. How does the *s*-plane get mapped into the *z*-plane under the impulse-invariance transformation?
27. How is the impulse-invariance transformation different from the matched *z* transformation?
28. Can we transform an analog Butterworth highpass filter into a digital highpass filter using the impulse-invariance transformation? Explain.
29. We can obtain a digital lowpass filter from an analog Chebyshev lowpass filter through the impulse-invariance mapping. True or false?
30. Can we transform an analog elliptic lowpass filter into a digital lowpass filter using the impulse-invariance transformation? Explain.
31. What is the basic principle behind the bilinear transformation?
32. How are analog and digital frequencies as well as the corresponding frequency responses related through the bilinear transformation?
33. What is prewarping and why is it needed?
34. Can we transform an analog highpass of any approximation filter into a digital highpass using the bilinear transformation? Explain.
35. How does the *s*-plane gets mapped into the *z*-plane under bilinear mapping?
36. Why is bilinear mapping superior to the impulse invariance transformation.
37. Explain why one should consider discrete-time frequency transformation over continuous-time frequency transformation.
38. What constraints are imposed on discrete-time frequency transformation? Explain the reasons behind them.
39. What types of discrete-time system satisfy the desired constraints on the frequency transformations?
40. Provide the step-by-step approach to obtain a highpass digital filter given its specifications.

Problems

Tutorial problems

1. Consider the noncausal filter given by

$$H(z) = \frac{z}{(z - 0.8)(1 - 0.8z)}. \quad 0.8 < |z| < 1.25$$

   (a) Determine the frequency response of the filter and show that it is a zero-phase response.
   (b) Determine analytically the impulse response of the filter.
   (c) Using the `filtfilt` function compute the impulse response of the filter and compare it with your answer in (b).

2. Consider a seventh-order analog Butterworth lowpass filter $H_c(s)$ with 3 dB cutoff frequency of 20 Hz.
   (a) Determine and graph the pole locations of $H_c(s)$.
   (b) Plot the magnitude and log-magnitude responses over [0, 100] Hz range.
   (c) Determine frequencies at which the attenuation is 20 dB, 30 dB, and 40 dB.

3. Design an analog Butterworth lowpass filter with specifications: $F_p = 50$ Hz, $A_p = 0.5$ dB, $F_s = 80$ Hz, and $A_s = 45$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Also provide the zero-pole plot.

4. Using the definition of Chebyshev polynomials, show that for $|x| > 1$, $T_N(x)$ is given by hyperbolic functions (11.29).

5. Design an analog Chebyshev I lowpass filter with specifications: $\Omega_p = 20$ rad, $A_p = 0.2$ dB, $\Omega_s = 30$ rad, and $A_s = 40$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Also provide the zero-pole plot.

6. Design an analog Chebyshev II lowpass filter with specifications: $F_p = 15$ Hz, $A_p = 0.1$ dB, $F_s = 20$ Hz, and $A_s = 40$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact passband edge. Also provide the zero-pole plot.

7. Design an analog elliptic lowpass filter with specifications: $F_p = 10$ kHz, $A_p = 1$ dB, $F_s = 15$ kHz, and $A_s = 50$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact passband and stopband edges. Also provide the zero-pole plot.

8. Consider the following specifications for an analog highpass filter:

$$\Omega_s = 10 \tfrac{\text{rad}}{\text{sec}}, \quad A_s = 40 \text{ dB}, \quad \Omega_p = 15; \tfrac{\text{rad}}{\text{sec}}, \quad A_p = 1 \text{ dB}.$$

   (a) Design an analog Butterworth highpass filter using the `buttord` and `butter` functions and plot its magnitude response.
   (b) Using impulse-invariance transformation and $T_d = 1$ design a digital highpass filter and plots its magnitude response.
   (c) Comment on the feasibility of using impulse-invariance for transforming an analog highpass filter to a digital highpass filter.

9. Consider the specifications of a digital lowpass filter given below:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.4\pi \text{ radians}, \quad A_s = 40 \text{ dB}.$$

(a) Design the digital filter using elliptic approximation and impulse-invariance transformation with $T_d = 1$. Plot the magnitude and log-magnitude responses and comment on the efficacy of the design.

(b) Repeat (a) using $A_s = 60$ dB. Is this design any better?

10. Consider the design of a digital lowpass filter using impulse invariance transformation with specifications:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_s = 50 \text{ dB}, \qquad \omega_s = 0.4\pi \text{ radians}, \quad A_p = 1 \text{ dB}.$$

(a) Using a Butterworth prototype and $T_d = 1$ s, obtain the lowpass digital filter. Plot its magnitude and lag-magnitude responses. Also plot the impulse response of the digital filter superimposed on the impulse response of the analog prototype filter.

(b) Repeat (a) using $T_d = 0.1$ s.

(c) Repeat (a) using $T_d = 0.01$ s.

(d) Comment on the effect of T_d on frequency responses in the impulse-invariance design.

11. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \qquad \omega_s = 0.35\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Obtain a system function $H(z)$ in the rational function form that satisfies the above specifications so that the response is equiripple in the passband and monotone in the stopband. Use an impulse invariance approach.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

12. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \qquad \omega_s = 0.35\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using the bilinear transformation approach and the Butterworth approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

13. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.2\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \qquad \omega_s = 0.3\pi \text{ radians}, \quad A_s = 60 \text{ dB}.$$

(a) Using bilinear transformation and the Chebyshev I approximation approach obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

14. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.5\pi \text{ radians}, \quad A_p = 2 \text{ dB}, \qquad \omega_s = 0.6\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using bilinear transformation and the Chebyshev II approximation approach obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

15. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.1\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}, \quad \omega_s = 0.2\pi \text{ radians}, \quad A_s = 45 \text{ dB}.$$

(a) Using bilinear transformation and the elliptic approximation approach obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

16. A first-order lowpass continuous-time filter $H_c(s) = 10/(s + 1)$ is to be transformed into a digital lowpass filter using the analog frequency transformation given in Table 11.1 followed by bilinear mapping.

(a) Determine and plot pole and zero locations for the analog lowpass filter with cutoff frequency of $\Omega_c = 10$ rad.

(b) Determine and plot pole and zero locations for the digital filter with $T_d = 2$.

(c) Plot the magnitude response of the digital filter.

17. A highpass digital filter's specifications are given by:

$$\omega_s = 0.6\pi \text{ radians}, \quad A_s = 60 \text{ dB}, \quad \omega_p = 0.8\pi \text{ radians}, \quad A_p = 1 \text{ dB}.$$

(a) Using the Chebyshev II approximation obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

18. A digital filter is specified by the following band parameters:

$$\begin{aligned}
\text{Band-1: } & [0, 0.2\pi], & \text{Attn.} &= 40 \text{ dB}, \\
\text{Band-2: } & [0.3\pi, 0.6\pi], & \text{Attn.} &= 0.5 \text{ dB}, \\
\text{Band-3: } & [0.7\pi, \pi], & \text{Attn.} &= 50 \text{ dB}.
\end{aligned}$$

(a) Using the Butterworth approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

19. A digital filter is specified by the following band parameters:

$$\begin{aligned}
\text{Band-1: } & [0, 0.4\pi], & \text{Attn.} &= 1 \text{ dB}, \\
\text{Band-2: } & [0.55\pi, 0.65\pi], & \text{Attn.} &= 50 \text{ dB}, \\
\text{Band-3: } & [0.75\pi, \pi], & \text{Attn.} &= 1 \text{ dB}.
\end{aligned}$$

(a) Using the Chebyshev I approximation obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

Basic problems

20. Another approach for zero-phase IIR filtering is as follows. Let $x[n]$ be an input signal and $h[n]$ denote the causal and stable IIR system. First, $x[n]$ is filtered through $h[n]$ to obtain output $y_1[n]$. Next, the flipped signal $x[-n]$ is filtered through $h[n]$ to obtain $y_2[n]$. Finally, the zero-phase output $y[n]$ is given by $y[n] = y_1[n] + y_2[n]$.

(a) Let $h_{zp}[n]$ be the impulse response of the filter with input $x[n]$ and output $y[n]$. Determine $h_{zp}[n]$ in terms of $h[n]$.

(b) Determine the frequency response $H_{zp}(e^{j\omega})$ and show that its phase response is zero.

(c) Let $x[n] = u[n] - u[n-10]$ and $H(z) = 1/(1 - 0.9z^{-1})$. Determine the zero-phase response $y[n]$ and verify using the `filtfilt` function.

21. Consider a ninth-order analog Butterworth lowpass filter $H_c(s)$ with 3 dB cutoff frequency of 10 Hz.

(a) Determine and graph the pole locations of $H_c(s)$.

(b) Plot the magnitude and log-magnitude responses over $[0, 100]$ Hz range.

(c) Determine frequencies at which the attenuation is 30 dB, 40 dB, and 50 dB.

22. Design an analog Butterworth lowpass filter with specifications: $\Omega_p = 10$ rad, $A_p = 0.1$ dB, $\Omega_s = 15$ rad, and $A_s = 40$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Also provide the zero-pole plot.

23. Design an analog Chebyshev I lowpass filter with specifications: $F_p = 2$ kHz, $A_p = 1$ dB, $F_s = 3.5$ kHz, and $A_s = 50$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact stopband edge. Also provide the zero-pole plot.

24. Following an approach similar to the one used in solving (11.34), show that the zeros $\{\zeta_k\}$ of the Chebyshev II prototype are given by (11.54). Furthermore, from (11.55) show that the poles are given by (11.65).

25. Design an analog Chebyshev II lowpass filter with specifications: $\Omega_p = 20$ rad, $A_p = 0.1$ dB, $\Omega_s = 30$ rad, and $A_s = 35$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact passband edge. Also provide the zero-pole plot.

26. Design an analog elliptic lowpass filter with specifications: $F_p = 50$ Hz, $A_p = 1$ dB, $F_s = 60$ Hz, and $A_s = 30$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact passband and stopband edges. Also provide the zero-pole plot.

27. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.2\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}, \quad \omega_s = 0.35\pi \text{ radians}, \quad A_s = 45 \text{ dB}.$$

(a) Obtain a system function $H(z)$ in the rational function form that satisfies the above specifications with monotonic passband and stopband. Use an impulse invariance approach.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

28. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.15\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.3\pi \text{ radians}, \quad A_s = 45 \text{ dB}.$$

(a) Using an impulse invariance approach obtain a system function $H(z)$ in the cascade form that satisfies the above specifications with equiripple passband and monotone stopband.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

29. Consider the specifications of a digital lowpass filter given below:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.4\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Design the digital filter using the Chebyshev II approximation and impulse-invariance transformation with $T_d = 1$. Plot the magnitude and log-magnitude responses and comment on the feasibility of the design.

(b) Repeat (a) using $A_s = 70$ dB. Is this design any better?

30. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.45\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using bilinear transformation and the Butterworth approximation approach obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

31. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.4\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}, \quad \omega_s = 0.55\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using bilinear transformation and the Chebyshev I approximation approach obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

32. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.2\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.4\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using bilinear transformation and the Chebyshev II approximation approach obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

33. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.55\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}, \qquad \omega_s = 0.7\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using bilinear transformation and the elliptic approximation approach obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

34. In the matched z-transformation method, zeros and poles of $H_c(s)$ are mapped into zeros and poles of $H(z)$ using an exponential function. Thus if $s = \alpha$ is one of the roots (zero or pole) of $H_c(s)$, then the corresponding root of $H(z)$ is $z = e^{\alpha T_d}$, where T_d is a design parameter. Let

$$H_c(s) = \frac{s(s + 10)}{s^2 + 2s + 101}.$$

(a) Using $T_d = 1$, obtain the system function $H(z)$.

(b) Plot the magnitude of the frequency responses of $H_{c(s)}$ and $H(z)$ and compare them.

(c) Plot impulse responses $h_c(t)$ and $h[n]$ on the same axis and compare them.

35. A first-order lowpass continuous-time filter $H_c(s) = 10/(s + 1)$ is to be transformed into a digital bandpass filter using analog frequency transformation given in Table 11.1 followed by the bilinear mapping.

(a) Determine and plot pole and zero locations for the analog bandpass filter with cutoff frequencies of $\Omega_{c1} = 50$ rad and $\Omega_2 = 100$ rad.

(b) Determine and plot pole and zero locations for the digital filter with $T_d = 2$.

(c) Plot the magnitude response of the digital filter.

36. Show that the most general allpass system given in (11.103) satisfies the three constraints required of every frequency transformation.

37. Consider a simple first-order lowpass digital filter given by $H_{lp}(z) = (z + 1)/(z - a)$ where $0 < a < 1$. We want to transform it into a highpass filter $H_{hp}(z)$ using the mapping $z^{-1} \rightarrow G(z^{-1}) = -(z^{-1} + \alpha)/(1 + \alpha z^{-1})$.

(a) Draw the signal flow graphs of $H_{lp}(z)$ and $G(z^{-1})$.

(b) Replace the branch z^{-1} in $H_{lp}(z)$ by the signal flow graph of $G(z^{-1})$ and simplify to create the signal flow graph for $H_{hp}(z)$. Can such a signal flow graph be implemented? Explain.

(c) Obtain the rational function $H_{hp}(z)$ using the frequency transformation $G(z^{-1})$ and draw its signal flow graph. Is this implementation realizable? Why?

38. A highpass digital filter's specifications are given by:

$$\omega_s = 0.6\pi \text{ radians}, \quad A_s = 40 \text{ dB}, \quad \omega_p = 0.8\pi \text{ radians}, \quad A_p = 1 \text{ dB}.$$

(a) Using the Butterworth approximation obtain a system function $H(z)$ in the cascade function form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

39. A highpass digital filter's specifications are given by:

$$\omega_s = 0.55\pi \text{ radians}, \quad A_s = 50 \text{ dB}, \quad \omega_p = 0.7\pi \text{ radians}, \quad A_p = 1 \text{ dB}.$$

(a) Using the Chebyshev II approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

40. A highpass digital filter's specifications are given by:

$$\omega_s = 0.45\pi \text{ radians}, \quad A_s = 60 \text{ dB}, \quad \omega_p = 0.55\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}.$$

(a) Using the elliptic approximation obtain a system function $H(z)$ in the parallel function form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

41. A digital filter is specified by the following band parameters:

$$\begin{aligned} \text{Band-1: } [0, 0.25\pi], \quad & \text{Attn.} = 40 \text{ dB}, \\ \text{Band-2: } [0.3\pi, 0.6\pi], \quad & \text{Attn.} = 1 \text{ dB}, \\ \text{Band-3: } [0.65\pi, \pi], \quad & \text{Attn.} = 50 \text{ dB}. \end{aligned}$$

(a) Using the Butterworth approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

42. A digital filter is specified by the following band parameters:

$$\begin{aligned} \text{Band-1: } [0, 0.2\pi], \quad & \text{Attn.} = 0.5 \text{ dB}, \\ \text{Band-2: } [0.3\pi, 0.55\pi], \quad & \text{Attn.} = 50 \text{ dB}, \\ \text{Band-3: } [0.7\pi, \pi], \quad & \text{Attn.} = 1 \text{ dB}. \end{aligned}$$

(a) Using the elliptic approximation obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

43. A digital filter is specified by the following band parameters:

$$
\begin{array}{ll}
\text{Band-1: } [0, 0.3\pi], & \text{Attn. } = 50 \text{ dB}, \\
\text{Band-2: } [0.4\pi, 0.5\pi], & \text{Attn. } = 1 \text{ dB}, \\
\text{Band-3: } [0.6\pi, \pi], & \text{Attn. } = 50 \text{ dB}.
\end{array}
$$

(a) Using the Chebyshev II approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

44. A digital filter is specified by the following band parameters:

$$
\begin{array}{ll}
\text{Band-1: } [0, 0.2\pi], & \text{Attn. } = 40 \text{ dB}, \\
\text{Band-2: } [0.3\pi, 0.5\pi], & \text{Attn. } = 1 \text{ dB}, \\
\text{Band-3: } [0.6\pi, \pi], & \text{Attn. } = 50 \text{ dB}.
\end{array}
$$

(a) Using the Chebyshev I approximation obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

45. A digital filter is specified by the following band parameters:

$$
\begin{array}{ll}
\text{Band-1: } [0, 0.5\pi], & \text{Attn. } = 0.5 \text{ dB}, \\
\text{Band-2: } [0.6\pi, 0.7\pi], & \text{Attn. } = 50 \text{ dB}, \\
\text{Band-3: } [0.75\pi, \pi], & \text{Attn. } = 0.5 \text{ dB}.
\end{array}
$$

(a) Using the Butterworth approximation obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

Assessment problems

46. Design an analog Butterworth lowpass filter with specifications: $F_p = 5$ kHz, $A_p = 1$ dB, $F_s = 7$ kHz, and $A_s = 50$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Also provide the zero-pole plot.

47. Consider a fifth-order analog Chebyshev I lowpass filter $H_c(s)$ with passband edge of 10 Hz and attenuation of 1 dB.

(a) Determine and graph the pole locations of $H_c(s)$.

(b) Plot the magnitude and log-magnitude responses over [0, 100] Hz range.

(c) Determine frequencies at which the attenuation is 30 dB, 40 dB, and 50 dB.

48. Design an analog Chebyshev I lowpass filter with specifications: $\Omega_p = 4$ rad, $A_p = 1$ dB, $\Omega_s = 5$ rad, and $A_s = 40$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact stopband edge. Also provide the zero-pole plot.

49. Consider a sixth-order analog Chebyshev II lowpass filter $H_c(s)$ with stopband edge of 20 Hz and attenuation of 40 dB.

(a) Determine and graph the pole locations of $H_c(s)$.

(b) Plot the magnitude and log-magnitude responses over $[0, 50]$ Hz range.

(c) Determine frequencies at which the attenuation is 0.1 dB, 0.5 dB, and 1 dB.

50. Design an analog Chebyshev II lowpass filter with specifications: $F_p = 25$ kHz, $A_p = 1$ dB, $F_s = 35$ rad, and $A_s = 40$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact passband edge. Also provide the zero-pole plot.

51. Design an analog elliptic lowpass filter with specifications: $\Omega_p = 20$ rad, $A_p = 1$ dB, $\Omega_s = 30$ rad, and $A_s = 60$ dB. Provide plots of the magnitude, log-magnitude, group-delay, and impulse responses. Determine the exact passband and stopband edges. Also provide the zero-pole plot.

52. Assume that a stable and causal analog filter has a double pole at $s = \alpha$, that is, $H_c(s) = A/(s - \alpha)^2$.

(a) Determine the impulse response $h_c(t)$ and sample it at $t = nT_d$ to obtain $h[n]$.

(b) Determine $H(z)$ and comment on its structure in terms of the double pole of $H_c(s)$ at $s = \alpha$.

(c) Generalize this result for the repeated pole of $H_c(s)$ at $s = \alpha$ with multiplicity of r.

53. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.4\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}, \quad \omega_s = 0.5\pi \text{ radians}, \quad A_s = 40 \text{ dB}.$$

(a) Using the impulse invariance approach obtain a system function $H(z)$ in the cascade form that satisfies the above specifications with monotonic passband and stopband.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

54. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.25\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}, \quad \omega_s = 0.35\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using the impulse invariance approach obtain a system function $H(z)$ in the parallel form that satisfies the above specifications with equiripple passband and monotone stopband.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

55. Consider the specifications of a digital lowpass filter given below:

$$\omega_p = 0.3\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.4\pi \text{ radians}, \quad A_s = 30 \text{ dB}.$$

(a) Design the digital filter using elliptic approximation and impulse-invariance transformation with $T_d = 2$. Plot the magnitude and log-magnitude responses and comment on the feasibility of the design.

(b) Repeat (a) using $A_s = 60$ dB. Is this design any better?

56. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.3\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.5\pi \text{ radians}, \quad A_s = 40 \text{ dB}.$$

(a) Using bilinear transformation and the Butterworth approximation approach obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

57. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.2\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.4\pi \text{ radians}, \quad A_s = 50 \text{ dB}.$$

(a) Using bilinear transformation and the Chebyshev I approximation approach obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

58. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.45\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.55\pi \text{ radians}, \quad A_s = 45 \text{ dB}.$$

(a) Using bilinear transformation and the Chebyshev II approximation approach obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

59. A lowpass digital filter's specifications are given by:

$$\omega_p = 0.2\pi \text{ radians}, \quad A_p = 1 \text{ dB}, \quad \omega_s = 0.35\pi \text{ radians}, \quad A_s = 60 \text{ dB}.$$

(a) Using bilinear transformation and the elliptic approximation approach obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

60. A first-order lowpass continuous-time filter $H_c(s) = 10/(s + 1)$ is to be transformed into a digital highpass filter using the analog frequency transformation given in Table 11.1 followed by bilinear mapping.

(a) Determine and plot pole and zero locations for the analog highpass filter with cutoff frequency of $\Omega_c = 100$ rad.

(b) Determine and plot pole and zero locations for the digital filter with $T_d = 2$.

(c) Plot the magnitude response of the digital filter.

61. A first-order lowpass continuous-time filter $H_c(s) = 10/(s + 1)$ is to be transformed into a digital bandstop filter using the analog frequency transformation given in Table 11.1 followed by bilinear mapping.

    (a) Determine and plot pole and zero locations for the analog bandstop filter with cutoff frequencies of $\Omega_{c1} = 50$ rad and $\Omega_2 = 100$ rad.

    (b) Determine and plot pole and zero locations for the digital filter with $T_d = 2$.

    (c) Plot the magnitude response of the digital filter.

62. Consider a simple first-order lowpass digital filter given by $H_{lp}(z) = (z + 1)/(z - a)$ where $0 < a < 1$. We want to transform it into a highpass filter $H_{hp}(z)$ using the mapping $z^{-1} \rightarrow -(z^{-1} + \alpha)/(1 + \alpha z^{-1})$ or equivalently, $z \rightarrow -(z + \alpha)/(1 + \alpha)$.

    (a) By considering zero and pole of $H_{lp}(z)$ separately, determine the resulting zero and pole of $H_{hp}(z)$ by solving the transformation equation

$$r_{lp} = -\frac{r_{hp} + \alpha}{1 + \alpha r_{hp}},$$

    where r_{lp} and r_{hp} are roots (pole or zero) of the respective system functions.

    (b) By direct substitution of the transformation into $H_{lp}(z)$ obtain the rational function $H_{hp}(z)$ and compute its pole-zero. Compare your results. Which method is better?

63. A highpass digital filter's specifications are given by:

$$\omega_s = 0.7\pi \text{ radians}, \quad A_s = 40 \text{ dB}, \qquad \omega_p = 0.9\pi \text{ radians}, \quad A_p = 0.5 \text{ dB}.$$

    (a) Using the Butterworth approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

    (b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

    (c) Determine the exact band-edge frequencies for the given attenuation.

64. A highpass digital filter's specifications are given by:

$$\omega_s = 0.65\pi \text{ radians}, \quad A_s = 50 \text{ dB}, \qquad \omega_p = 0.8\pi \text{ radians}, \quad A_p = 1 \text{ dB}.$$

    (a) Using the Chebyshev I approximation obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

    (b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

    (c) Determine the exact band-edge frequencies for the given attenuation.

65. A highpass digital filter's specifications are given by:

$$\omega_s = 0.7\pi \text{ radians}, \quad A_s = 60 \text{ dB}, \qquad \omega_p = 0.8\pi \text{ radians}, \quad A_p = 1 \text{ dB}.$$

    (a) Using the elliptic approximation obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

    (b) Provide design plots in the form of log-magnitude, phase, group-delay, and impulse responses.

    (c) Determine the exact band-edge frequencies for the given attenuation.

66. A digital filter is specified by the following band parameters:

$$\begin{aligned}
\text{Band-1: } &[0, 0.2\pi], &\text{Attn. } &= 1 \text{ dB}, \\
\text{Band-2: } &[0.35\pi, 0.5\pi], &\text{Attn. } &= 50 \text{ dB}, \\
\text{Band-3: } &[0.65\pi, \pi], &\text{Attn. } &= 1 \text{ dB}.
\end{aligned}$$

(a) Using the Butterworth approximation obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

67. A digital filter is specified by the following band parameters:

$$
\begin{aligned}
&\text{Band-1: } [0, 0.4\pi], && \text{Attn. } = 40 \text{ dB}, \\
&\text{Band-2: } [0.45\pi, 0.55\pi], && \text{Attn. } = 1 \text{ dB}, \\
&\text{Band-3: } [0.65\pi, \pi], && \text{Attn. } = 50 \text{ dB}.
\end{aligned}
$$

(a) Using the Chebyshev I approximation obtain a system function $H(z)$ in the rational function form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

68. A digital filter is specified by the following band parameters:

$$
\begin{aligned}
&\text{Band-1: } [0, 0.2\pi], && \text{Attn. } = 1 \text{ dB}, \\
&\text{Band-2: } [0.35\pi, 0.5\pi], && \text{Attn. } = 60 \text{ dB}, \\
&\text{Band-3: } [0.65\pi, \pi], && \text{Attn. } = 0.5 \text{ dB}.
\end{aligned}
$$

(a) Using the elliptic approximation obtain a system function $H(z)$ in the cascade form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

69. A digital filter is specified by the following band parameters:

$$
\begin{aligned}
&\text{Band-1: } [0, 0.4\pi], && \text{Attn. } = 40 \text{ dB}, \\
&\text{Band-2: } [0.45\pi, 0.55\pi], && \text{Attn. } = 0.5 \text{ dB}, \\
&\text{Band-3: } [0.65\pi, \pi], && \text{Attn. } = 50 \text{ dB}.
\end{aligned}
$$

(a) Using the Chebyshev II approximation obtain a system function $H(z)$ in the parallel form that satisfies the above specifications.

(b) Provide design plots in the form of magnitude, log-magnitude, group-delay, and impulse responses.

(c) Determine the exact band-edge frequencies for the given attenuation.

Review problems

70. An analog signal

$$
x_c(t) = 5\sin(2\pi\,250t) + 10\sin(2\pi\,300t)
$$

is to be processed using the effective continuous-time system of Figure 6.18 in which the sampling frequency is 1 kHz.

(a) Design a minimum-order IIR digital filter that will suppress the 300 Hz component down to 50 dB while pass the 250 Hz component with attenuation of less than 1 dB. The digital filter should have an equiripple passband and stopband.

Determine the system function of the filter and plot its log-magnitude response in dB.

(b) Process the signal $x_c(t)$ through the effective analog system. Generate sufficient samples so the output response $y_c(t)$ goes into steady state. Plot the steady state $y_{ss}(t)$ and comment on the filtering result.

(c) Repeat parts (a) and (b) by designing an equiripple FIR filter. Compare the orders of the two filters and their filtering results.

71. Consider the following bandpass digital filter specifications:

$$\text{Stopband-1: } [0, 0.4\pi], \qquad \text{Attn.} = 40 \text{ dB},$$
$$\text{Passband: } [0.45\pi, 0.55\pi], \qquad \text{Attn.} = 0.5 \text{ dB},$$
$$\text{Stopband-2: } [0.65\pi, \pi], \qquad \text{Attn.} = 50 \text{ dB}.$$

(a) Design a minimum order FIR filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(b) Design a minimum order IIR filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses. From your plots determine the exact band-edge frequencies.

(e) Compare the two filter designs in terms of their responses and orders.

72. Consider the following multiband digital filter specifications:

$$\text{Band-1: } [0, 0.1\pi], \qquad 0 \le |H(e^{j\omega})| \le 0.01,$$
$$\text{Band-2: } [0.2\pi, 0.5\pi], \qquad 0.45 \le |H(e^{j\omega})| \le 0.5,$$
$$\text{Band-3: } [0.6\pi, 0.7\pi], \qquad 0 \le |H(e^{j\omega})| \le 0.01,$$
$$\text{Band-4: } [0.8\pi, \pi], \qquad 0.95 \le |H(e^{j\omega})| \le 1.$$

(a) Design a minimum order FIR filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(b) Design a minimum order IIR filter to satisfy the above specifications. The filter should also have equiripple responses in bands 2 and 4 and monotone responses in bands 1 and 3. Plot its magnitude, log-magnitude, and group-delay responses. From your design obtain the order of the filter and from plots determine the exact band-edge frequencies. (Hint: Consider a parallel of a bandpass and a highpass filter.)

(c) Compare the two filter designs in terms of their responses and orders.

73. Consider the following multiband digital filter specifications:

$$\text{Band-1: } [0, 0.2\pi], \qquad 0.9 \le |H(e^{j\omega})| \le 1,$$
$$\text{Band-2: } [0.3\pi, 0.5\pi], \qquad 0.6 \le |H(e^{j\omega})| \le 0.7,$$
$$\text{Band-3: } [0.6\pi, 0.7\pi], \qquad 0.3 \le |H(e^{j\omega})| \le 0.4,$$
$$\text{Band-4: } [0.8\pi, \pi], \qquad 0 \le |H(e^{j\omega})| \le 0.01.$$

(a) Design a minimum order FIR filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(b) Design a minimum order IIR filter to satisfy the above specifications. The filter should also have equiripple responses in bands 2 and 4 and monotone responses

in bands 1 and 3. Plot its magnitude, log-magnitude, and group-delay responses. From your design obtain the order of the filter and from plots determine the exact band-edge frequencies. (Hint: Consider a parallel of frequency-selective filters.)

(c) Compare the two filter designs in terms of their responses and orders.

74. Consider the following bandstop digital filter specifications:

$$\text{Passband-1: } [0, 0.4\pi], \qquad \text{Attn.} = 40 \text{ dB,}$$
$$\text{Stopband: } [0.45\pi, 0.55\pi], \qquad \text{Attn.} = 0.5 \text{ dB,}$$
$$\text{Passband-2: } [0.65\pi, \pi], \qquad \text{Attn.} = 50 \text{ dB.}$$

(a) Design a minimum order Butterworth filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(b) Design a minimum order Chebyshev I filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(c) Design a minimum order Chebyshev II filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(d) Design a minimum order elliptic filter to satisfy the above specifications. Plot its magnitude, log-magnitude, and group-delay responses.

(e) From your designs and plots compare the following: (i) Filter order, (ii) Exact band-edge frequencies, and (iii) Group-delay responses.

12 Multirate signal processing

A key feature of the discrete-time systems discussed so far is that the signals at the input, output, and every internal node have the same sampling rate. However, there are many practical applications that either require or can be implemented more efficiently by processing signals at different sampling rates. Discrete-time systems with different sampling rates at various parts of the system are called *multirate systems*. The practical implementation of multirate systems requires changing the sampling rate of a signal using discrete-time operations, that is, without reconstructing and resampling a continuous-time signal. The fundamental operations for changing the sampling rate are decimation and interpolation. The subject of this chapter is the analysis, design, and efficient implementation of decimation and interpolation systems, and their application to two important areas of multirate signal processing: sampling rate conversion and multirate filter banks.

Study objectives

After studying this chapter you should be able to:

- Understand the operations of decimation, interpolation, and arbitrary sampling rate change in the time and frequency domains.
- Understand the efficient implementation of discrete-time systems for sampling rate conversion using polyphase structures.
- Design a special type of filter (Nyquist filters), which are widely used for the efficient implementation of multirate filters and filter banks.
- Understand the operation, properties, and design of two-channel filter banks with perfect reconstruction analysis and synthesis capabilities.

12.1 Sampling rate conversion

The need for sampling rate conversion arises in many practical applications, including digital audio, communication systems, image processing, and high-definition television. Conceptually, the sampling rate conversion process can be regarded as a two-step operation. First, the discrete-time signal is reconstructed into a continuous-time signal; then, it is resampled at a different sampling rate. We emphasize that the above steps are only a mental picture for illustrating the underlying principle. In practice, the conversion is implemented using discrete-time signal processing without actual reconstruction of any continuous-time signal. However, the idea of an "underlying" continuous-time signal (even a fictional one) is very useful for understanding sampling rate conversion operations.

Given a discrete-time signal $x[n]$ we can determine a continuous-time signal $x_c(t)$ such that $x[n] = x_c(nT)$ by using the bandlimited interpolation (6.25)

$$x_c(t) = \sum_{k=-\infty}^{\infty} x[n] \frac{\sin[\pi(t-nT)/T]}{\pi(t-nT)/T}. \tag{12.1}$$

Because of the way it was formed, the signal $x_c(t)$ has Fourier transform $X_c(j\Omega) = 0$ for $|\Omega| > \pi/T$. Furthermore, the spectra of $x[n]$ and $x_c(t)$ are related by (6.12)

$$X(e^{j\Omega T}) = \frac{1}{T} \sum_{k=-\infty}^{\infty} X_c(j(\Omega - k\Omega_s)), \tag{12.2}$$

where $\Omega_s = 2\pi/T$ is the sampling frequency. For future reference, we recall that the time and frequency variables are related by $t = nT$ and $\omega = \Omega T$.

The computation of a sequence $x_0[n] \triangleq x_c(nT_0)$ from the known sequence $x[n] = x_c(nT)$ for $T_0 \neq T$, without reconstructing $x_c(t)$, is called *resampling* or *sampling rate change*. In this section we discuss the discrete-time implementation of three important resampling operations: (a) $T_0 = DT$, (b) $T_0 = T/I$, and (c) $T_0 = T(D/I)$, where D and I are integers.

12.1.1 Sampling rate decrease by an integer factor

Suppose that we sample a continuous-time signal $x_c(t)$ with sampling periods T and $T_D \triangleq DT$, where D is an integer. The corresponding discrete-time signals $x[n] = x_c(nT)$ and $x_D[n] = x_c(nT_D)$ are related by

$$x_D[n] \triangleq \mathcal{D}_D\{x[n]\} \triangleq x[nD] \triangleq x_{\downarrow D}[n]. \tag{12.3}$$

The system defined by (12.3) is called a *downsampler* or *sampling rate compressor (SRC)* or simply *compressor*, because it reduces the sampling rate of a discrete-time signal

Sampling Rate Compressor
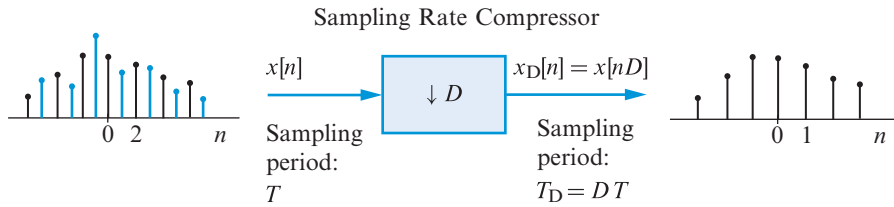
Figure 12.1 Block diagram representation of sampling rate compressor system.

(*downsampling*) by an integer factor D. This system is essentially a discrete-time sampler which retains only one out of each group of D consecutive input samples into its output. The compressor, which is represented as shown in Figure 12.1, can be implemented in MATLAB by the function

$$\text{function y=src(x,D)}$$

$$\text{y=x(1:D:length(x)).} \tag{12.4}$$

We note that the samples of $x[n]$ for $n \neq mD$ are lost in the downsampling. The MATLAB SP toolbox provides the function

$$\text{y=downsample(x,D,k)} \tag{12.5}$$

to implement the operation $y[n] = x[nD + k]$, $k = 0, 1, \ldots, D - 1$.

The spectrum of the "sampled" discrete-time signal $x_D[m]$ is given by

$$X_D\left(e^{j\Omega T_D}\right) = \frac{1}{T_D} \sum_{\ell=-\infty}^{\infty} X_c\left(j\left(\Omega - \ell\Omega_D\right)\right), \tag{12.6a}$$

$$= \frac{1}{DT} \sum_{\ell=-\infty}^{\infty} X_c\left(j\left(\Omega - \ell\frac{2\pi}{DT}\right)\right), \tag{12.6b}$$

where $\Omega_D = 2\pi/T_D = \Omega_s/D$ is the reduced sampling rate. This process is illustrated in Figure 12.2 for $D = 3$. We note that increasing the sampling period from T to $T_D = DT$ brings the replicas of $X_c(j\Omega)$ closer. To avoid aliasing, the reduced sampling rate should satisfy the condition $\Omega_D = \Omega_s/D \geq 2\Omega_{max}$. Alternatively, if T is fixed, we should band-limit $x_c(t)$ to $\Omega_{max} \leq \Omega_s/(2D)$ before we sample at a lower rate.

Careful inspection of Figure 12.2 shows that the spectrum $X_D\left(e^{j\Omega T_D}\right)$ of the decimated sequence can be obtained directly by putting replicas of the spectrum $X\left(e^{j\Omega T}\right)$ of the original sequence at $\Omega = m(\Omega_s/D)$ for $m = 0, 1, \ldots, D - 1$, adding them and then scaling the sum by $1/D$. This result can be obtained analytically by expressing the summation index ℓ in (12.6b) as follows:

$$\ell = m + kD. \quad 0 \leq m \leq D - 1 \tag{12.7}$$
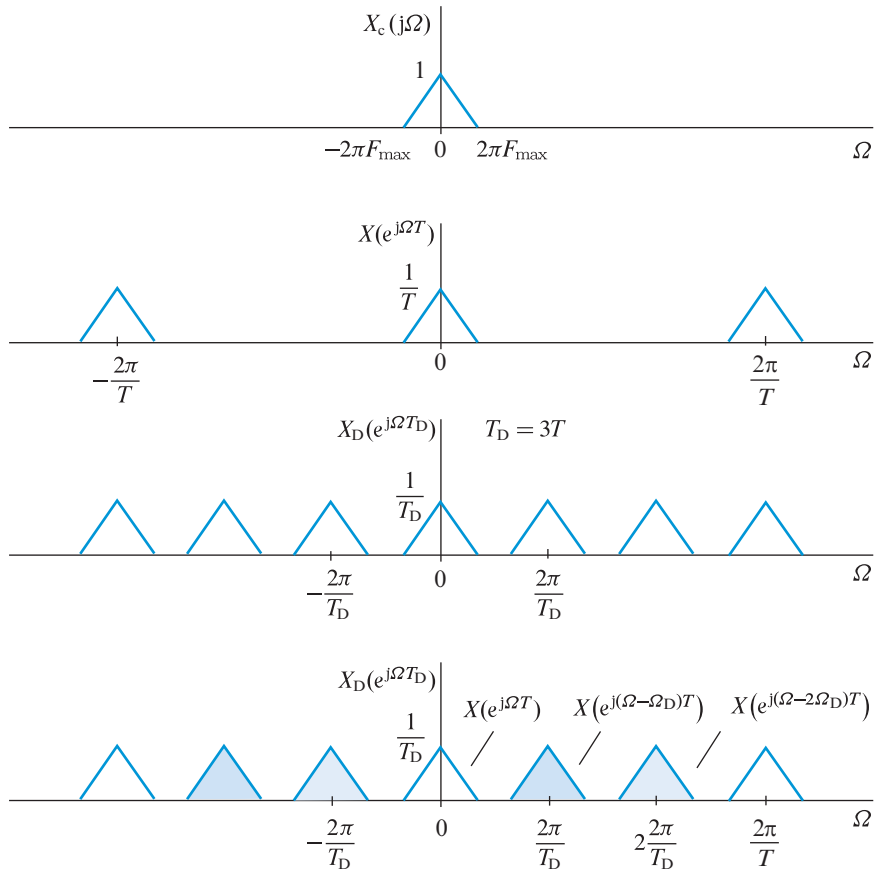
Figure 12.2 Frequency-domain illustration of sampling rate reduction.

Basically, this indexing "breaks" the infinite summation with index ℓ into a sum of D infinite summations with index k. Indeed, replacing ℓ by $m + kD$ yields

$$X_D(e^{j\Omega T_D}) = \frac{1}{D}\sum_{m=0}^{D-1}\left[\frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left(j\left(\Omega - m\frac{2\pi}{DT} - k\frac{2\pi}{T}\right)\right)\right].\tag{12.8}$$

Comparing the term within the square brackets to (12.2), we obtain

$$\frac{1}{T}\sum_{k=-\infty}^{\infty} X_c\left(j\left(\Omega - m\frac{2\pi}{DT} - k\frac{2\pi}{T}\right)\right) = X\left(e^{j(\Omega - 2\pi m/(DT))T}\right).\tag{12.9}$$

Substituting (12.9) into (12.8) and using $\Omega_D = 2\pi/T_D$, we obtain the desired result

$$X_D\left(e^{j\Omega T_D}\right) = \frac{1}{D}\sum_{m=0}^{D-1} X\left(e^{j(\Omega - m\Omega_D)T}\right).\tag{12.10}$$

This equation, which relates directly the spectra of $x[n]$ and $x_D[n]$, is similar to (12.2) relating the spectra of $x[n]$ and $x_c(t)$. However, there are two important differences. First, we note that the summation in (12.10) involves a finite number of replicas. This is explained by the fact that discrete-time signals have a finite frequency range. Second, the left hand side of (12.10) depends on T_D and the right hand side on T. This does not lead into any confusion as long as we retain this information and we plot the corresponding signal and spectra with respect to the physical time and frequency variables t and Ω. However, we should be careful when we work with the normalized variables n and ω. If we define the normalized frequency variable as

$$\omega \triangleq \Omega T_D, \tag{12.11}$$

and note that $\Omega T = \Omega T_D/D = \omega/D$, we can express (12.10) as follows:

$$X_D(e^{j\omega}) = \frac{1}{D} \sum_{m=0}^{D-1} X\left(e^{j(\omega - 2\pi m)/D}\right). \tag{12.12}$$

If we had defined $\omega \triangleq \Omega T$, the result would have been another version of (12.12). To understand the operation of downsampling in the discrete-time frequency domain, we discuss the effects of sampling on a discrete-time sinusoid.

Example 12.1 Spectrum expansion

Consider the discrete-time sinusoidal signal $x[n] = \cos(\omega_0 n)$. If $\omega_0 = 2\pi/16$, this signal has fundamental period $N = 16$ samples. Downsampling $x[n]$ by a factor $D = 2$ yields the sequence

$$x_D[n] = \cos[2\pi \omega_0(nD)] = \cos[2\pi(D\omega_0)n], \tag{12.13}$$

which is a cosine with frequency $\omega = D\omega_0 = 2\pi/8$ and fundamental period $N = 8$ samples (see Figure 12.3). Therefore, decreasing the period by a factor D increases the
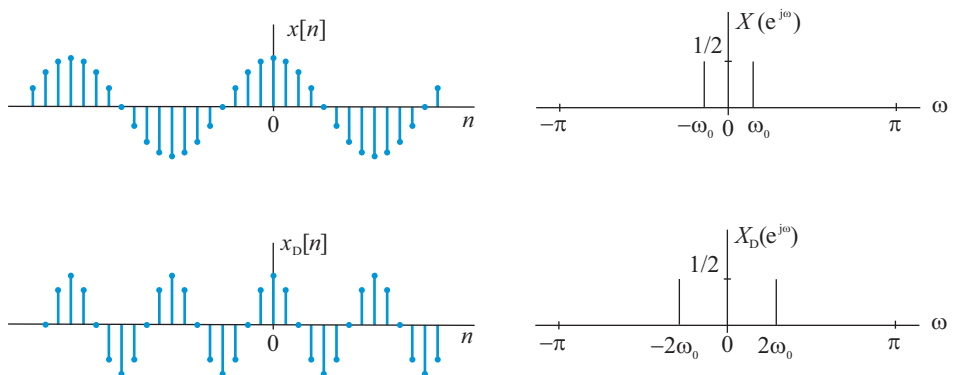
Figure 12.3 Spectrum (not axis) expansion during downsampling. The signal $x[n] = \cos(\omega_0 n)$ becomes $x_D[n] = \cos[(D\omega_0)n]$ after downsampling.

fundamental frequency by the same factor. The result is a scaling of the position (*not* the frequency axis) of frequency components by a factor D. In this example, the downsampled frequency $D\omega_0$ is inside the fundamental interval, that is, $D\omega_0 \leq \pi$; therefore, there is no aliasing. If $D\omega_0 > \pi$, the lines at $\pm D\omega_0$ will be folded back about $\pm\pi$ at locations $\pm(2\pi - D\omega_0)$. ∎

In the next example we illustrate the effects of downsampling on the spectrum of a discrete-time time signal with a continuous spectrum.

Example 12.2   Downsampling by a factor of two

Consider a sequence $x[n]$ with Fourier transform $X(e^{j\omega})$

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}. \tag{12.14}$$

We next downsample $x[n]$ and $x[n+1]$ by a factor $D = 2$ to obtain the sequences

$$x[n] = \{\ldots, x[-2], x[-1], x[0], x[1], x[2], \ldots\}$$
$$\mathcal{D}_2\{x[n]\} = \{\ldots, x[-4], x[-2], x[0], x[2], x[4], \ldots\} = x_D[n]$$
$$\mathcal{D}_2\{x[n+1]\} = \{\ldots, x[-3], x[-1], x[1], x[3], x[5], \ldots\}$$

Since $\mathcal{D}_2\{x[n+1]\} \neq x_D[n+1]$, the downsampler is a time-varying system; however, we can easily show that downsampling is a linear operation.

The Fourier transform of the downsampled sequence $x_D[n]$ is given by

$$X_D(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x_D[n]e^{-j\omega n} = \sum_{n=-\infty}^{\infty} x[2n]e^{-j\omega n}. \tag{12.15}$$

Relating $X_D(e^{j\omega})$ to $X(e^{j\omega})$ is not trivial because the index of the sequence and the index in the exponent are different. To avoid this obstacle, we note that if we knew the transform

$$X_u(e^{j\omega}) \triangleq \sum_{n=-\infty}^{\infty} x[2n]e^{-j\omega 2n} \tag{12.16}$$

we could obtain the Fourier transform of $x_D[n] = x[2n]$ by the simple substitution

$$X_D(e^{j\omega}) = X_u(e^{j\omega/2}). \tag{12.17}$$

It remains now to relate $X_\mathrm{u}(e^{j\omega})$ to $X(e^{j\omega})$. To this end, we expand (12.16) as

$$X_\mathrm{u}(e^{j\omega}) = \cdots + x[-2]e^{j2\omega} + x[0] + x[2]e^{-j2\omega} + x[-4]e^{-j4\omega} + \cdots \qquad (12.18)$$

This is the Fourier transform of a sequence obtained by replacing the odd-indexed samples of $x[n]$ by zeros. Thus, if we define the sequence

$$x_\mathrm{u}[n] = \{\ldots, x[-2], 0, x[0], 0, x[2], \ldots\},$$

we note that $x_\mathrm{D}[n] = x[2n] = x_\mathrm{u}[2n]$. The sequence $x_\mathrm{u}[n]$ can be written as

$$x_\mathrm{u}[n] = \frac{1}{2}\big(x[n] + (-1)^n x[n]\big) = \frac{1}{2}\big(x[n] + e^{j\pi n} x[n]\big).$$

Taking the Fourier transform of both sides yields

$$X_\mathrm{u}(e^{j\omega}) = \frac{1}{2}\left[X(e^{j\omega}) + X(e^{j(\omega-\pi)})\right]. \qquad (12.19)$$

From (12.17) and (12.19) we finally obtain the desired relationship

$$X_\mathrm{D}(e^{j\omega}) = \frac{1}{2}X(e^{j\omega/2}) + \frac{1}{2}X(e^{j(\omega/2-\pi)}). \qquad (12.20)$$

To understand the meaning of this relation, we write (12.20) for $D = 2$. The result is

$$X_\mathrm{D}(e^{j\Omega T_\mathrm{D}}) = \frac{1}{2}X(e^{j\Omega T}) + \frac{1}{2}X(e^{j(\Omega T-\pi)}). \qquad (12.21)$$

If we define the relative frequency by $\omega \triangleq \Omega T_\mathrm{D}$, relations (12.12) and (12.21) are equivalent. ∎

We now turn our attention to the graphical construction of $X_\mathrm{D}(e^{j\omega})$ directly from $X(e^{j\omega})$ according to (12.12). This process involves the following steps:

1. Stretch $X(e^{j\omega})$ by a factor D to obtain $X(e^{j\omega/D})$; we note that the highest frequency ω_H is "repositioned" to frequency $\omega = \omega_\mathrm{H}D$.
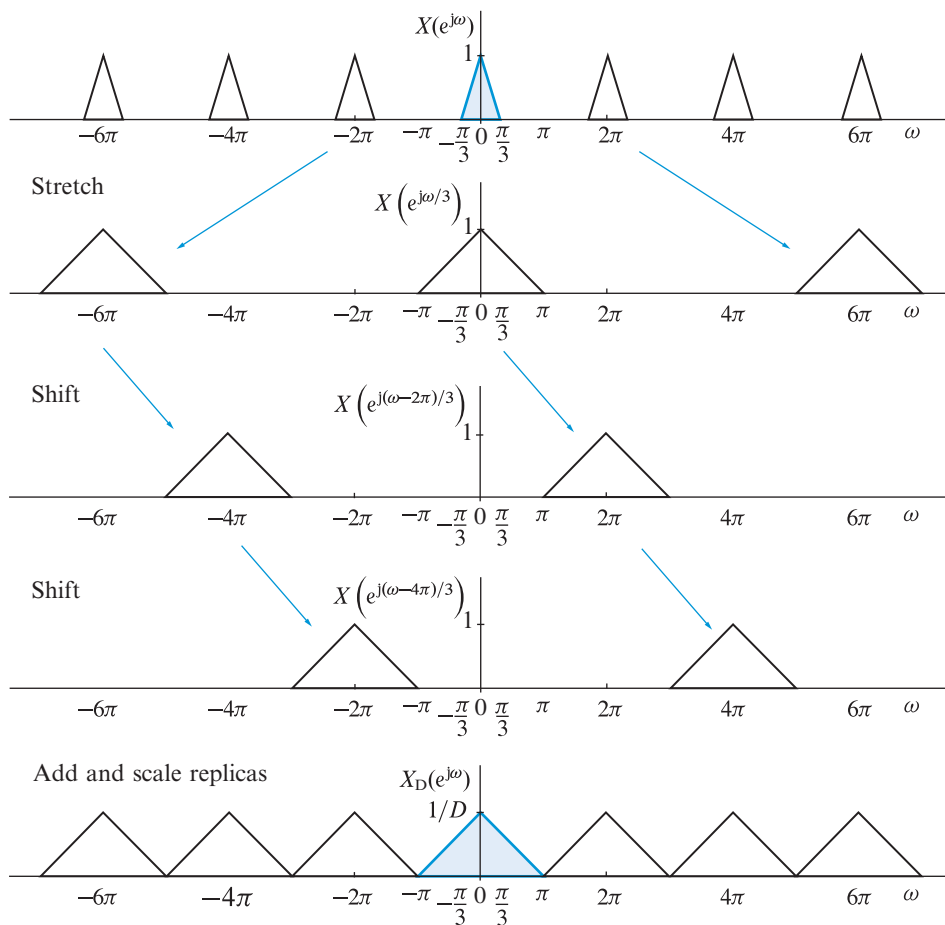
Figure 12.4 Frequency-domain interpretation of downsampling for $D = 3$.

2. Create and put D copies of $X(e^{j\omega/D})$ at the frequencies $\omega = 2\pi m$ (integer multiples of 2π) for $m = 0, 1, \ldots, D - 1$.
3. Add the D stretched and shifted replicas and then divide by D to obtain the spectrum $X_D(e^{j\omega})$ of the downsampled sequence $x_D[n] = x[nD]$.

This process is illustrated in Figure 12.4 for $D = 3$ and $\omega_H = \pi/3$. We emphasize the importance of first stretching the spectrum $X(e^{j\omega})$ by a factor D to obtain $X(e^{j\omega/D})$ and then shifting the result at multiples of 2π (*not* multiples of $2\pi/D$).

   To establish a closer analogy between continuous-time and discrete-time sampling, we define the *discrete-time sampling rate*

$$f_s \triangleq \frac{1}{D} \quad \text{or} \quad \omega_s = \frac{2\pi}{D}, \tag{12.22}$$

Decimator
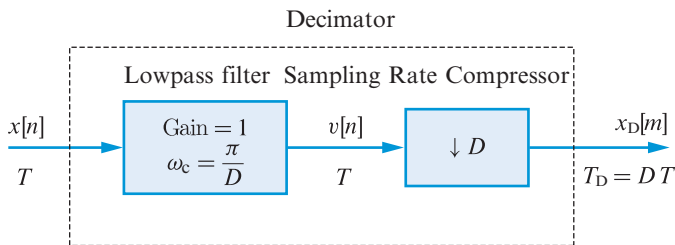
Figure 12.5 Representation of a system for sampling rate decrease by an integer factor D.

which simply means "keep one out of every D samples." Then, the condition that should be satisfied to avoid aliasing can be stated as

$$\text{If } X\left(e^{j\omega}\right) = 0, \quad \omega_H \le |\omega| \le \pi \quad \text{then} \quad \omega_s = \frac{2\pi}{D} \ge 2\omega_H, \qquad (12.23)$$

or equivalently $f_s = 1/D \ge 2f_H$, where $\omega_H = 2\pi f_H$. Downsampling without aliasing is illustrated in Figure 12.4. If the spectrum of $x[n]$ does *not* satisfy (12.23), we will have aliasing distortion. To avoid potential aliasing distortion we precede the sample rate compressor by a lowpass antialiasing filter with cutoff frequency $\omega_c = \pi/D$. The combined system, shown in Figure 12.5, is called a *decimator* and the corresponding process is known as *decimation*. A frequency-domain interpretation of this approach is given in Figure 12.6.

If we use an FIR lowpass filter of order M, the output of the decimator is

$$x_D[m] = v[mD] = \sum_{k=0}^{M} h[k]x[mD - k]. \qquad (12.24)$$

Since only every Dth output of the filter is needed, the overall computational cost is reduced by a factor of D. The samples of the input signal are shifted in memory every T seconds, but the filter computes the output only every T/D seconds; that is, the filter computes one output sample every D input samples (see Tutorial Problem 1). The fundamental difference between (12.24) and convolution is that the input sequence (or equivalently the impulse response) is shifted by D samples, instead of one sample, when the output sequence index changes by one. This approach is used by the book toolbox function `firdec` given in Figure 12.7.

MATLAB also provides two functions to perform decimation. The first one, defined by MATLAB functions for decimation

$$\texttt{y=upfirdn(x,h,1,D)}, \qquad (12.25)$$

provides the same functionality as is given in (12.24). The second function:

$$\texttt{y=decimate(x,D,M,'fir')}, \qquad (12.26)$$

designs and uses an Mth-order lowpass FIR filter with cutoff frequency $\omega_c = \pi/D$ using function `fir1` (see Section 10.3).
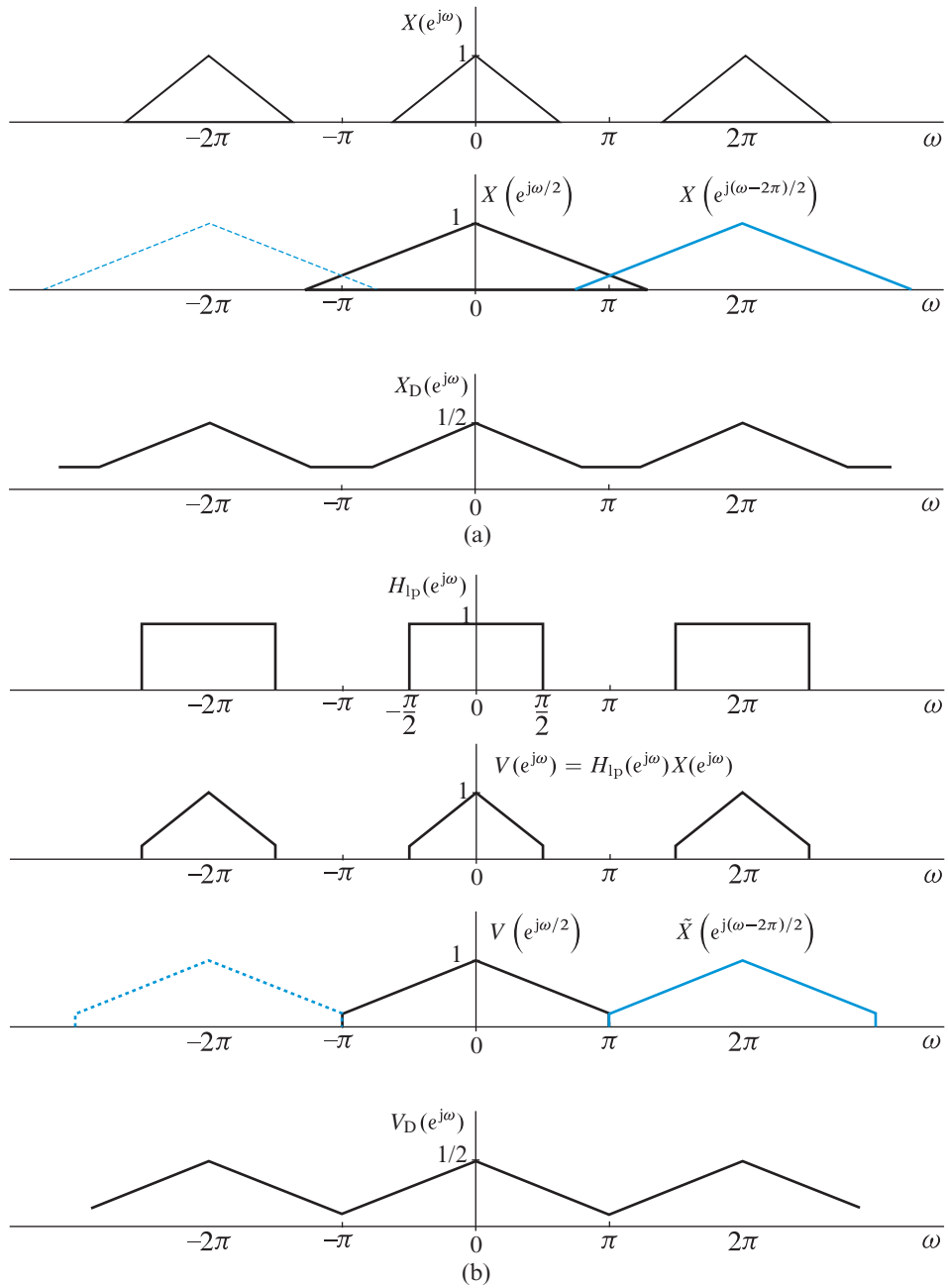
Figure 12.6 (a) Downsampling ($D = 2$) with aliasing. (b) Downsampling with lowpass prefiltering (decimation) to avoid aliasing. The distortion caused by lowpass filtering is less damaging than aliasing distortion.

```
function y = firdec(h,x,D)
% FIR decimation by a factor D
M=length(h)-1; x=[zeros(1,M) x]; L=length(x); h=fliplr(h);
for n=1:floor(L/D-M);
    m=(n-1)*D;
    y(n)=dot(h,x(m+1:m+M+1));
end
```

Figure 12.7 MATLAB function for implementation of an FIR decimator.

The continuous-time signal $x_c(t)$ can be perfectly reconstructed from the samples $x_D[m]$ using the ideal bandlimited interpolation formula

$$x_c(t) = \sum_{m=-\infty}^{\infty} x_D[m] \frac{\sin[\pi(t - mDT)/DT]}{\pi(t - mDT)/DT}. \tag{12.27}$$

If we evaluate (12.27) at $t = nT$, and recall that $x[n] = x_c(nT)$, we obtain

$$x[n] = \sum_{m=-\infty}^{\infty} x_D[m] \frac{\sin[\pi(n - mD)/D]}{\pi(n - mD)/D}, \tag{12.28}$$

which provides perfect reconstruction of $x[n]$ directly from $x_D[m]$. Clearly, the ideal discrete-time interpolator (12.28) is not practically realizable. However, practical approximations obtained by windowing the ideal interpolation sequence and using the DFT are possible (see Problem 29).

12.1.2　Sampling rate increase by an integer factor

Increasing the sampling rate of a discrete-time signal $x[n]$ by an integer factor I (*upsampling*) requires the insertion of $(I - 1)$ samples between consecutive samples of $x[n]$. Thus, upsampling is an information preserving operation. If we consider the underlying continuous-time signal $x_c(t)$, the ideal goal is to obtain a sequence of samples

$$x_I[n] \triangleq x_c(nT_I) = x_c(nT/I), \tag{12.29}$$

where $T_I = T/I$, from the samples of the discrete-time signal

$$x[n] = x_c(nT). \tag{12.30}$$

This can be done using the interpolation formula (12.28), simply replacing D by I. Thus, the interpolated signal $x_I[n]$ is obtained using the formula

$$x_I[n] = \sum_{m=-\infty}^{\infty} x[m] \frac{\sin[\pi(n - mI)/I]}{\pi(n - mI)/I}. \tag{12.31}$$

The ideal interpolation kernel in (12.31) and its Fourier transform are given by

$$g_{bl}[n] \triangleq \frac{\sin(\pi n/I)}{\pi n/I} \xleftrightarrow{\text{DTFT}} G_{bl}(e^{j\omega}) = \begin{cases} I, & 0 \le |\omega| \le \pi/I \\ 0. & \pi/I < |\omega| \le \pi \end{cases} \qquad (12.32)$$

In general, the interpolation process is described by the formula

$$x_I[n] = \sum_{m=-\infty}^{\infty} x[m]g_r[n - mI], \qquad (12.33)$$

which yields the ideal interpolator (12.31) if $g_r[n] = g_{bl}[n]$. Since $g_{bl}[0] = 1$ and $g_{bl}[nI] = 0$, for $n \ne 0$, the ideal interpolator does not alter the original samples. Thus, we usually require $g_r[n]$ to have the following property

$$g_r[n] = \begin{cases} 1, & n = 0 \\ 0. & n = \pm I, \pm 2I, \dots \end{cases} \qquad (12.34)$$

To understand the operation described by (12.33), we evaluate the Fourier transform of the interpolated sequence $x_I[n]$. The result is given by

$$X_I(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x[m]G_r(e^{j\omega})e^{-jmI\omega}, \qquad (12.35a)$$

$$= G_r(e^{j\omega}) \sum_{m=-\infty}^{\infty} x[m]e^{-jmI\omega}. \qquad (12.35b)$$

Since the spectrum of the original sequence $x[n]$ is given by

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-j\omega n}, \qquad (12.36)$$

we can express relation (12.35b) as

$$X_I(e^{j\omega}) = G_r(e^{j\omega})X(e^{j\omega I}). \qquad (12.37)$$

We note that, for $I = 1$, the operation specified by (12.33) is a convolution sum; in this case, formula (12.37) is the frequency-domain description of an LTI system. To understand the meaning of (12.37) for $I \ne 1$, we explicitly write $X(e^{j\omega I})$ as follows:

$$X(e^{j\omega I}) = \sum_{m=-\infty}^{\infty} x[m]e^{-j\omega I m}, \qquad (12.38a)$$

$$= \cdots + x[0] + x[1]e^{-jI\omega} + x[2]e^{-j2I\omega} + \cdots \qquad (12.38b)$$

Careful inspection of (12.38) shows that the terms $e^{-jk\omega}$ are missing when k is not an integer multiple of I. Therefore, if we define a sequence $x_u[n]$ by inserting $(I-1)$ zeros between consecutive samples of $x[n]$, that is, by

$$x_u[n] \triangleq \mathcal{U}_I\{x[n]\} \triangleq x_{\uparrow I} x[n] \triangleq \begin{cases} x[n/I], & n \text{ is a multiple of } I \\ 0, & \text{otherwise} \end{cases} \qquad (12.39)$$

we have in the frequency domain (see discussion in Example 12.2)

$$X_u(e^{j\omega}) = X(e^{j\omega I}). \qquad (12.40)$$

The system described by (12.39) is called *upsampler* or *sampling rate expander* (SRE) and is represented by the block diagram shown in Figure 12.8. The SRE is implemented in MATLAB by the function

```
function y=sre(x,I)
y=zeros(1:I*length(x));          (12.41)
y(1:I:length(y))=x;
```

MATLAB provides the more general function

```
y=upsample(x,I,k)                (12.42)
```

which expands the input by a factor of I and then shifts the obtained sequence by inserting k zeros, $k = 0, 1, \ldots, I - 1$, in the beginning.

We can now use the SRE to obtain an alternative implementation of the interpolation operation (12.31). Indeed, substitution of (12.40) into (12.37) gives

$$X_I(e^{j\omega}) = G_r(e^{j\omega})X_u(e^{j\omega}), \qquad (12.43)$$
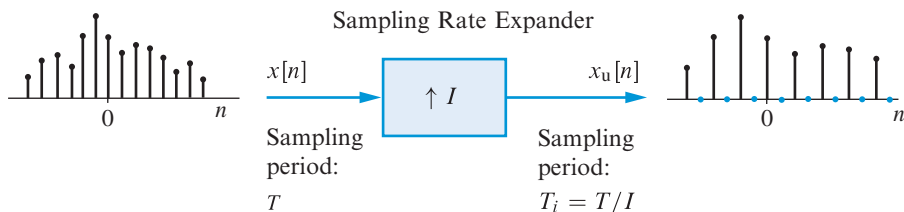
Figure 12.8 Representation of a sampling rate expander.
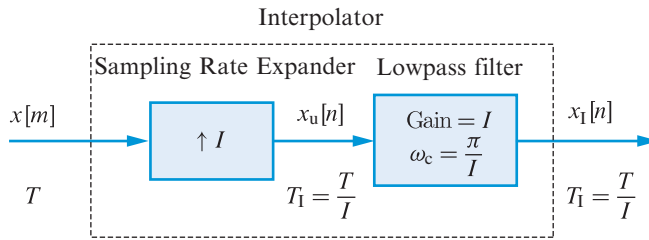
Figure 12.9 Discrete-time system for sampling rate increase by an integer factor I using ideal bandlimited interpolation.

which corresponds to an LTI system described by the convolution summation

$$x_I[n] = \sum_{k=-\infty}^{\infty} x_u[k] g_r[n-k]. \tag{12.44}$$

From the preceding discussion, we conclude that a general system (12.31) for increasing the sampling rate by an integer factor I can be implemented by a SRE followed by a filter with impulse response $g_r[n]$. If we use an ideal lowpass filter with gain I and cutoff frequency $\omega_c = \pi/I$ (see Figure 12.9) we obtain the ideal interpolator described by (12.31). The sequences $g_r[n]$ used in (12.33) and (12.44) are identical; however, the interpretation is different. In the former case, $g_r[n]$ is the kernel or characteristic sequence of a linear time-varying interpolation system; in the latter case, $g_r[n]$ is the impulse response of a lowpass filter.

Frequency-domain interpretations To understand the interpolation process in the frequency-domain, we start with the upsampler, whose operation is determined by (12.39). Since $X_u(e^{\pm j\pi}) = X(e^{\pm j\pi I})$, we conclude that I periods of $X(e^{j\omega})$ are compressed and form one period of $X_u(e^{j\omega})$. This is illustrated in Figure 12.10 for $I = 3$. The extra $(I-1)$ copies of the compressed spectrum introduced by upsampling are called *images*. In this sense, we say that the upsampler creates an imaging effect. The interpolation filter $G_{bl}(e^{j\omega})$ removes all these images and scales the spectrum by I to compensate for the $1/I$ reduction in signal bandwidth. If the interpolation filter has a cutoff frequency larger than π/I or we use a non-ideal filter, energy from the images remains in the interpolated signal. This type of distortion is known as "*post-aliasing*" in the computer graphics literature; see Mitchell and Netravali (1988).

Figure 12.11 provides a frequency-domain interpretation of decimation and interpolation operations in terms of the physical variables t and Ω. We note that increasing the sampling period by an integer factor D inserts $(D-1)$ replicas of $X_c(j\Omega)$ in the original Nyquist range and then reduces its width by a factor D. The SRE increases the Nyquist range by a factor I by including $(D-1)$ additional copies of $X_c(j\Omega)$. If we use the normalized time index $n = t/T$, downsampling increases the rate of change of the output signal (see Figure 12.3). However, because we use the normalized frequency variable $\omega = \Omega T$, the width of the Nyquist range is always equal to 2π. Therefore, downsampling "stretches" the input signal spectrum by a factor D, and upsampling compresses the input spectrum by
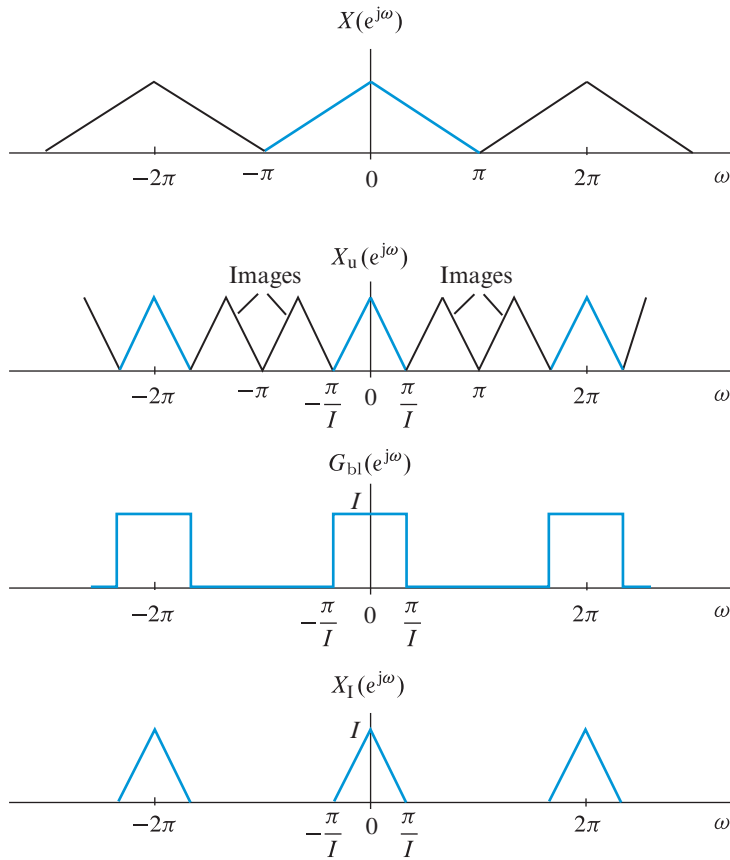
Figure 12.10 Frequency-domain interpretation of interpolation for $I = 3$.

a factor I (see Figure 12.12). This is essentially a consequence of the scaling theorem for the discrete-time Fourier transform.

MATLAB functions for interpolation There are two MATLAB functions that can be used to interpolate a sequence $x[n]$ by a factor I. The function

$$y=\text{interp}(x,I) \tag{12.45}$$

uses an internally designed lowpass filter based on a procedure described by Oetken *et al.* (1975). The function

$$y=\text{upfirdn}(x,h,I,1) \tag{12.46}$$

uses an FIR filter provided by the user (see Section 12.3 for the design of interpolation filters). At this point, we advise the reader to implement the interpolator in Figure 12.9 using the functions `sre` and `filter` (see Tutorial Problem 3). Efficient interpolator structures are discussed in Section 12.2.

Multirate signal processing
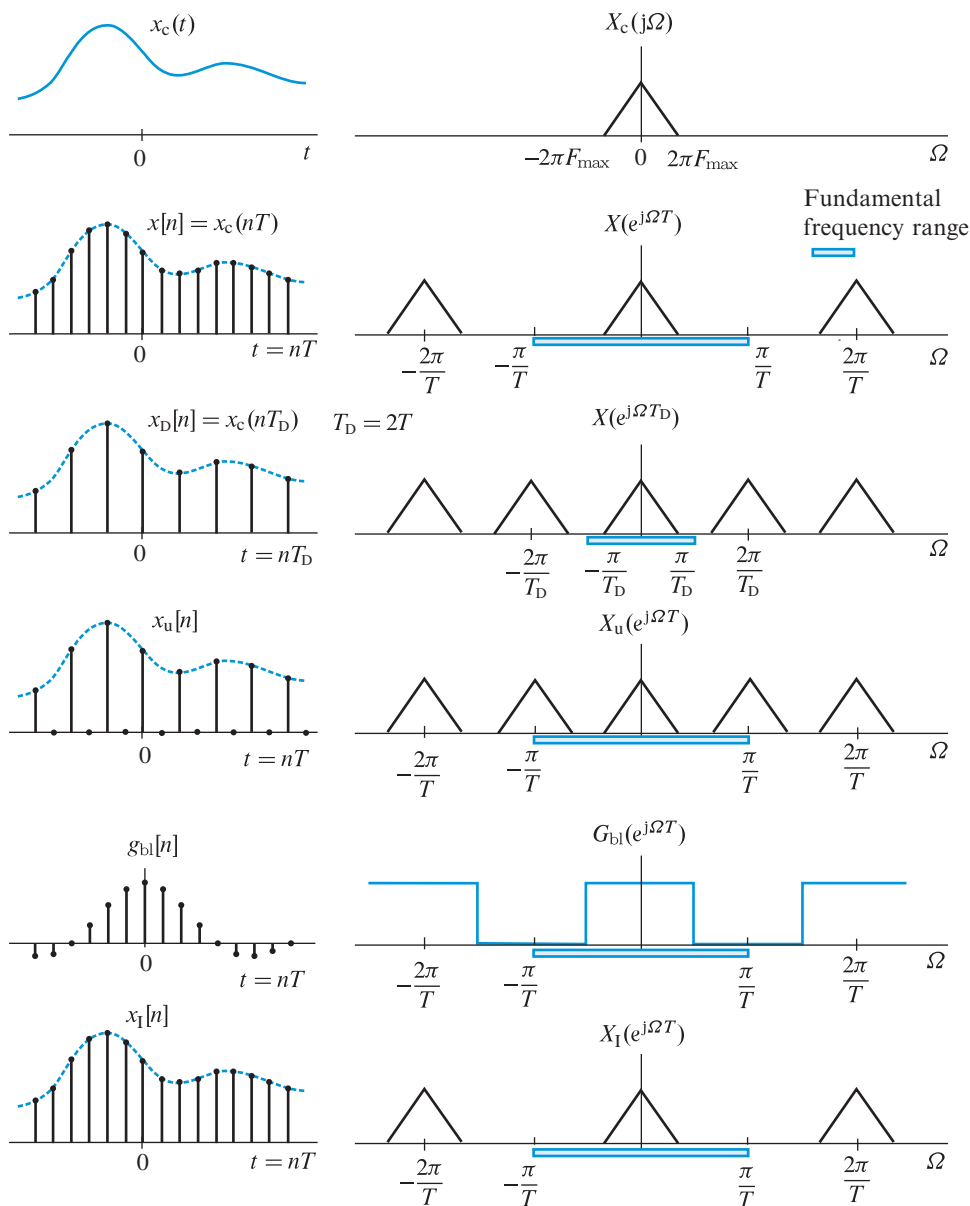
Figure 12.11 Decimation and interpolation operations in the time and frequency domains for $D = I = 2$ using the physical time (t) and the physical frequency (Ω) variables. Note the decrease of the fundamental frequency range from $-\pi/T < \Omega < \pi/T$ to $-\pi/T_D < \Omega < \pi/T_D$ during downsampling and vice versa during upsampling.
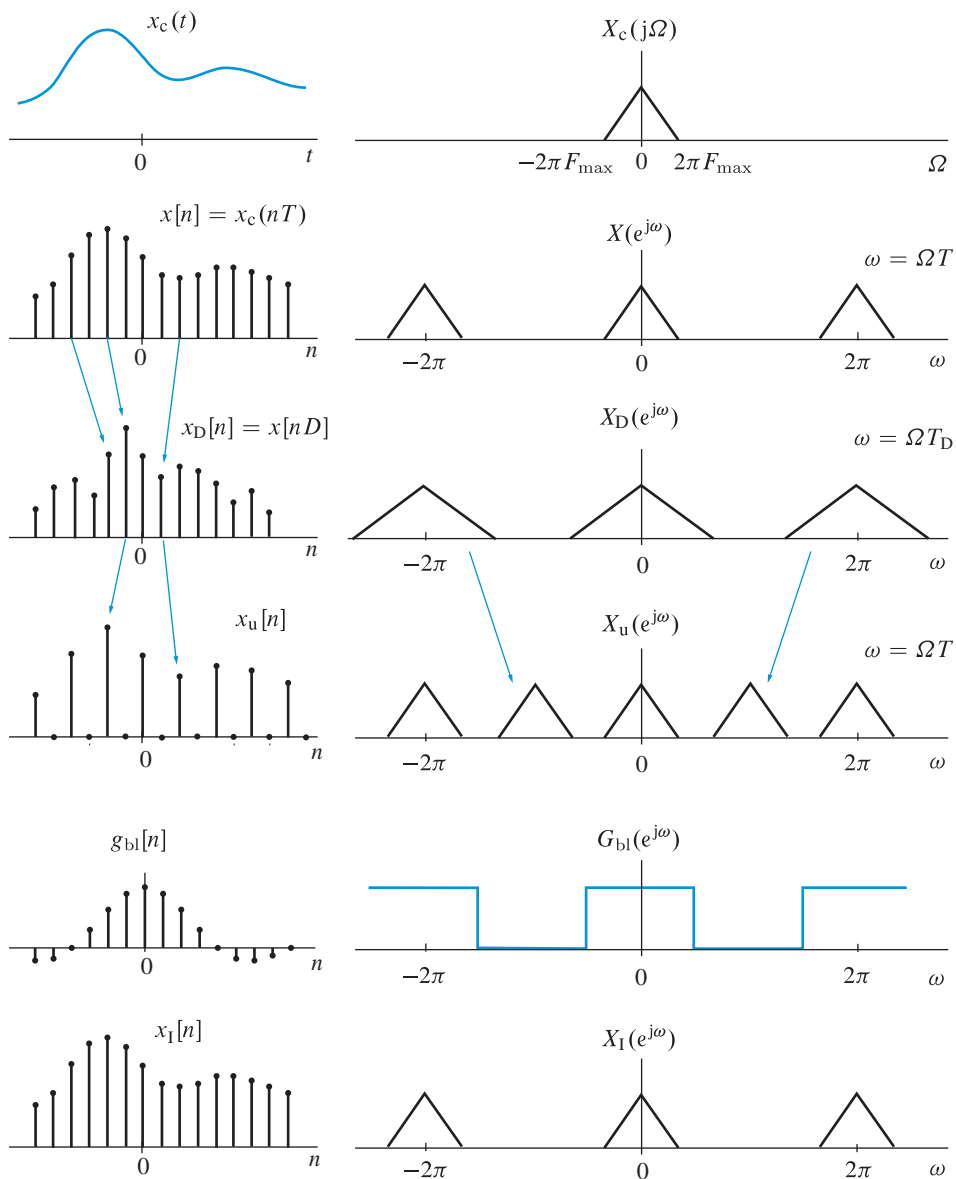
Figure 12.12 Decimation and interpolation operations in the time and frequency domains for $D = I = 2$ using the normalized time (n) and the normalized frequency (ω) variables. Note that the stretching of $X(e^{j\omega})$ may lead to aliasing in $X_D(e^{j\omega})$; however, the contraction of $X_D(e^{j\omega})$ always creates spectral images in $X_u(e^{j\omega})$ which must be removed by the interpolation lowpass filter.

Example 12.3   Linear interpolation

The most familiar of all practical interpolators is the linear interpolator, in which the inter-polated values between two consecutive samples lie on the straight line connecting these two samples (see Figure 12.13). To find the impulse response of the linear interpolator filter, we use the definition of linear interpolation

$$x_I[(m-1)I+k] = x[m-1] + (x[m]-x[m-1])\frac{k}{I} \tag{12.47a}$$

$$= x[m-1]\left(1-\frac{k}{I}\right) + x[m]\frac{k}{I} \tag{12.47b}$$

$$= x_u[(m-1)I]\left(1-\frac{k}{I}\right) + x_u[mI]\frac{k}{I} \tag{12.47c}$$

for $k = 0, 1, \ldots, I-1$ and $n = (m-1)I+k$. If we define the triangular sequence

$$g_{\text{lin}}[n] \triangleq \begin{cases} 1 - \dfrac{|n|}{I}, & -I < n < I \\ 0, & \text{otherwise} \end{cases} \tag{12.48}$$

(a)

(b)
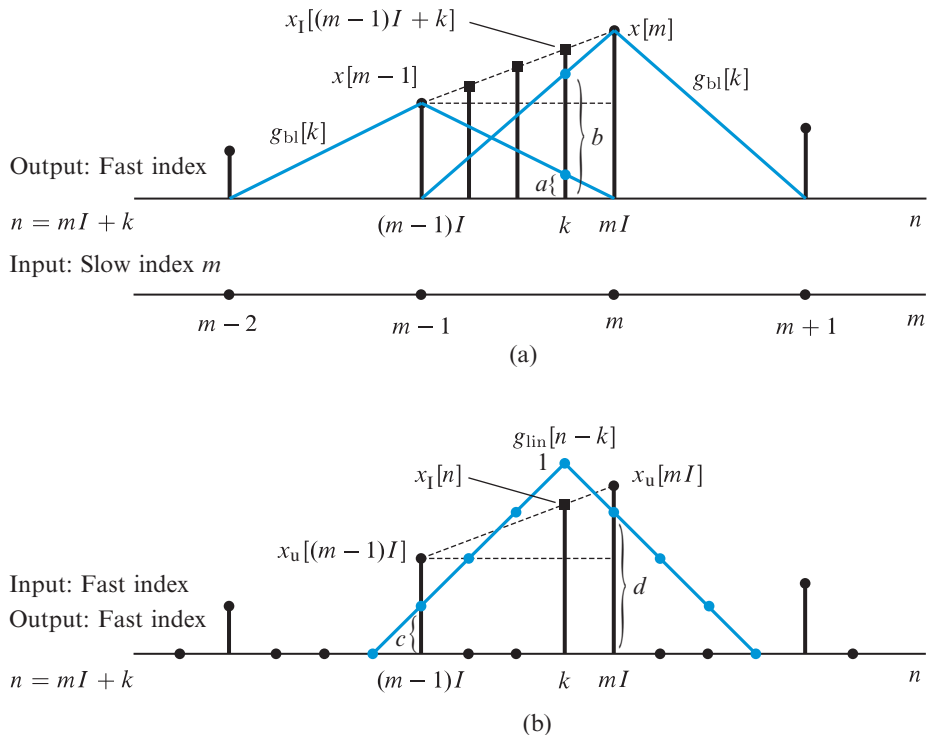
Figure 12.13 Operation of a linear interpolator: (a) the input $x[m]$ is indexed by the "slow" index m and the output by the "fast" index $n = Im + k$, (b) the input $x_u[n]$ and the output $x_I[n]$ are both indexed by the fast index n; in this case the operation is equivalent to a convolution summation.
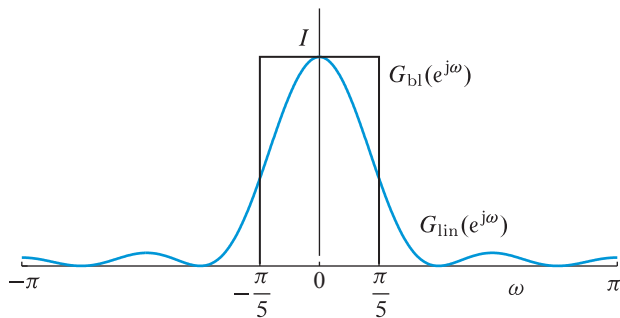
Figure 12.14 Frequency responses of the ideal bandlimited interpolator and a linear interpolator with $I = 5$.

we can express the linear interpolation formula (12.47) as a convolution summation (12.44); see Tutorial Problem 6 for details. Figure 12.13(a) expresses linear interpolation as a superposition of interpolation functions; Figure 12.13(b) expresses linear interpolation as a shift-weight-add convolution operation.

Since (12.48) is the impulse response of the linear interpolation filter, its frequency response is given by

$$G_{\text{lin}}\big(e^{j\omega}\big) = \frac{1}{I}\left[\frac{\sin(\omega I/2)}{\sin(\omega/2)}\right]^2. \tag{12.49}$$

Figure 12.14 shows the magnitude response of the ideal bandlimited interpolator and the linear interpolator with $I = 5$. We note that, unless the original signal being interpolated has bandwidth much smaller than π/I, the linear interpolator cannot sufficiently attenuate the images of the signal spectrum, which results in post-aliasing distortion. Therefore, as we intuitively expect, the linear interpolator provides reasonable performance for oversampled signals. Consider the decimation of the sequence $x[n] = \cos(2\pi 0.02n) + 3\sin(2\pi 0.0036n)$, $0 \le n \le 80$, by $D = 5$, followed by upsampling by $I = 5$, and linear interpolation implemented by linear filtering (see Tutorial Problem 9) as illustrated in the following MATLAB script:

```
N=80; n=(0:N-1); x=cos(2*pi*0.02*n)+3*sin(2*pi*0.0036*n);
D=5; xd=downsample(x,D); I = 5; xu=upsample(xd,I);
glin=(1:I-1); glin=[glin I fliplr(glin)]/I;
xi=filter(glin,1,xu).
```

Figure 12.15 shows the sequences resulting from downsampling, upsampling, and linear interpolation from the original sequence $x[n]$. Except for the initial and final four samples, the interpolation is acceptable. ■

Linear interpolation is the simplest and most widely used of the polynomial interpolation techniques studied extensively in numerical analysis; see for example Hamming (1973). Improved performance can be obtained by using higher-order Lagrange interpolation
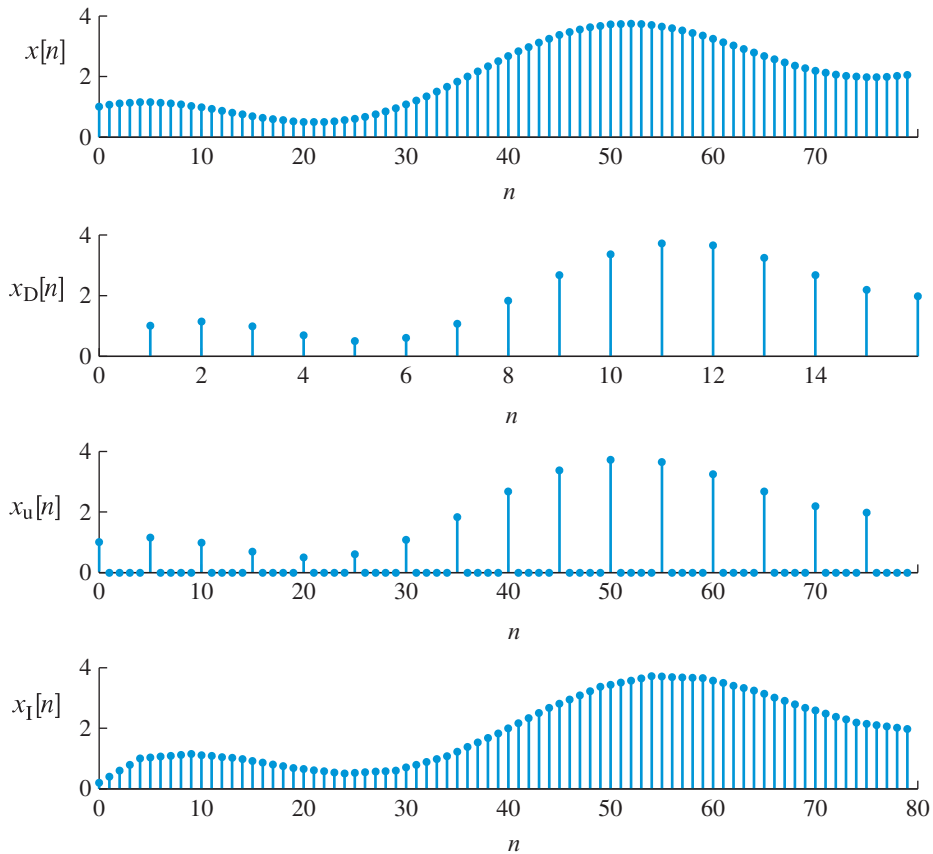
Figure 12.15 Decimation with $D = 5$ followed by upsampling by $I = 5$ and linear interpolation implemented by linear filtering.

and spline interpolation; these techniques can be formulated in the signal processing framework that we presented here and which is discussed in Schafer and Rabiner (1973) and Unser (1999). A simple spline technique known as cubic spline interpolation, which is widely used in image processing (see Keys (1981) and Pratt (2007)), is discussed in Problem 42. In the following example we discuss one application of interpolation and decimation that produces fractionally delayed samples of a bandlimited signal.

Example 12.4   Fractional delay

In many applications such as digital modems, music/audio signal processing, time-delay estimation, etc. it is important to know the exact sampling instances in addition to the sampling frequency. This necessitates delaying samples by a fractional amount which cannot be done by a simple (integer) delay operator as discussed in Chapter 9.

To understand the fractional delay, consider the underlying continuous-time signal $x_c(t)$. Using samples $x[n] = x_c(nT)$ of $x_c(t)$ we want to obtain samples $y[n]$ of the delayed signal

$y_c(t) = x_c(t - t_D)$. Thus we want

$$y[n] = y_c(nT) = x_c(nT - t_D) = x[n - \Delta], \tag{12.50}$$

where $\Delta \triangleq t_D/T$ is a fractional number. The frequency response of the discrete-time system in (12.50) is

$$H_{fd}(e^{j\omega}) = e^{-j\omega\Delta}, \tag{12.51}$$

which is an IIR allpass filter with impulse response

$$h[n] = \mathcal{F}^{-1}\left[e^{-j\Delta\omega}\right] = \frac{\sin[\pi(n - \Delta)]}{\pi(n - \Delta)}, \tag{12.52}$$

which is noncausal and unrealizable. Although it is possible to design a realizable FIR or IIR filter that approximates $h[n]$ in (12.52), sampling rate conversion provides another approach to implement fractional delays. Consider the signal

$$x_c(t) = \cos(2\pi\{40\}t) + 3\sin(2\pi\{144\}t), \tag{12.53}$$

which is sampled at $F_S = 1000$ Hz to obtain $x[n]$. We want to obtain $y[n] = x[n - \frac{1}{2}]$. To do this we first interpolate $x[n]$ using $I = 2$, then delay the resulting signal by one sample, and finally downsample the delayed interpolated signal using $D = 2$ to obtain $y[n]$. The following MATLAB script illustrates the details:

```
>> F1 = 40; F2 = 144; Fs = 1000;
>> N = 20; n = (-N:2*N); I = 2; D = I;
>> x = cos(2*pi*F1/Fs*n)+3*sin(2*pi*F2/Fs*n);
>> xI = interp(x,D); nd = -N*D:2*N*D; % Interpolate by I=2
>> yd = [0,xI(1:end-1)]; % Delay by one sample
>> y = downsample(yd,D); % Downsample by D=2
```

Figure 12.16 shows the original sampled signal $x[n]$, the interpolated signal $x_I[n]$, the delayed interpolated signal $y_d[n]$, and the final downsampled signal $y[n]$. The corresponding analog signal is also shown using a dashed line. It is evident that the technique using interpolation/delay/downsampling produces a fractionally delayed signal. A more general and efficient implementation is given in Laakso *et al.* (1996). ∎

12.1.3   Sampling rate change by a noninteger factor

We have shown how to decrease or increase the sampling rate by an integer factor. In this section, we show that we can combine interpolation and decimation to change the sampling rate by a rational factor I/D. Consider the cascade connection of an interpolator and a decimator shown in Figure 12.17(a). The interpolator increases the sampling rate by a
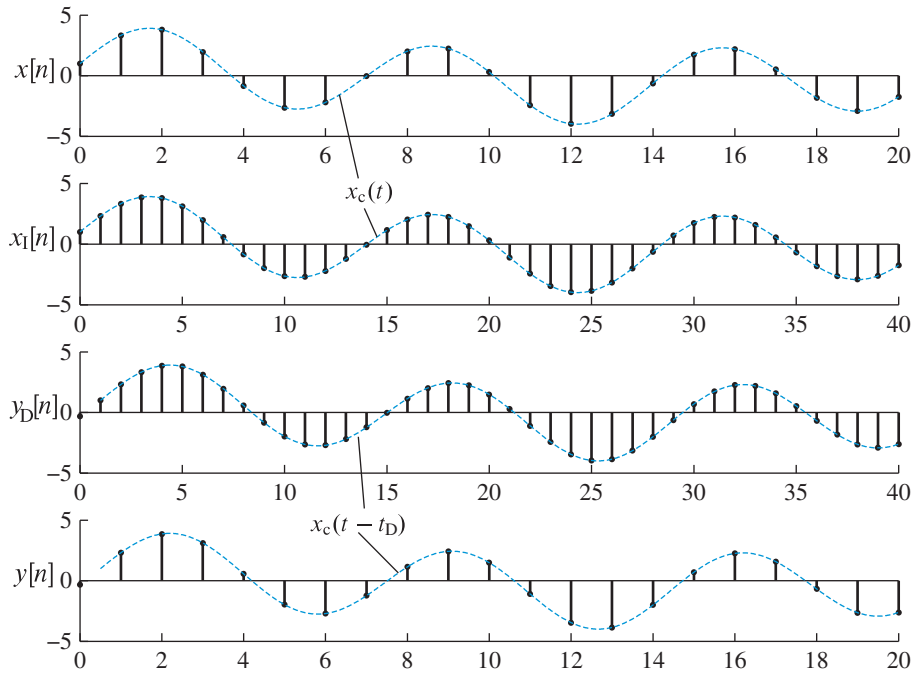
Figure 12.16 Fractional delay in Example 12.4.
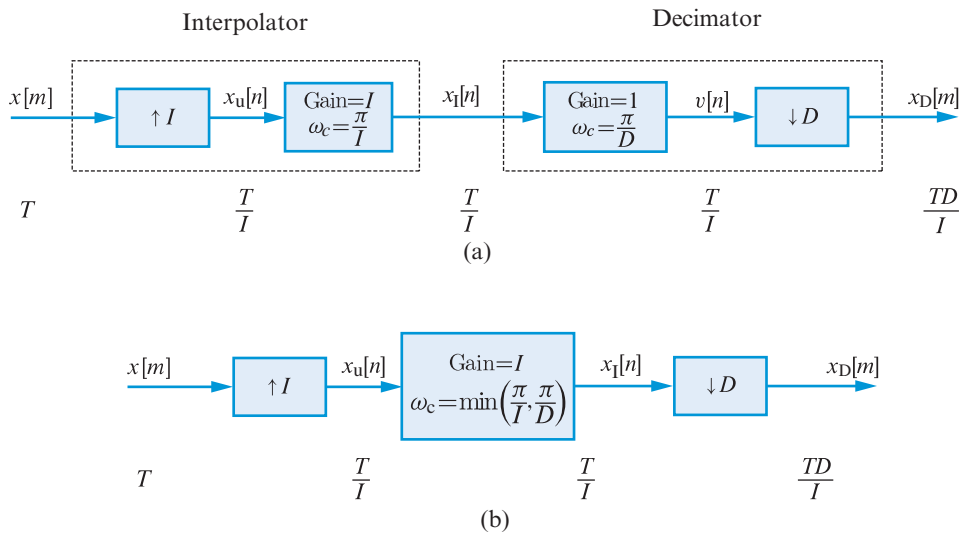
Figure 12.17 (a) System for changing the sampling rate by a rational factor. (b) Simplified system obtained by combining interpolation and decimation filters.

factor I and the decimator decreases the sampling rate by a factor D. The effective sampling rate of the output signal is $T_0 = (1/T)I/D$. Because decimation may cause aliasing, we always put the interpolator first to preserve the spectrum of $x[n]$. Since the interpolation and decimation lowpass filters are in cascade and operate at the same sampling rate, they can be replaced by a single lowpass filter as shown in Figure 12.17(b). This filter is defined by

$$
H(e^{j\omega}) = \begin{cases} I, & 0 \le |\omega| \le \min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \\ 0. & \min\left(\frac{\pi}{I}, \frac{\pi}{D}\right) \le |\omega| \le \pi \end{cases} \tag{12.54}
$$

When $I > D$ (sampling rate increase), the common filter acts as an anti-imaging postfilter to remove the images introduced by the upsampling operation; when $I < D$ (sampling rate decrease), the common filter acts as an antialiasing filter for the downsampling operation (if the original signal is not properly bandlimited). This choice is necessary to avoid aliasing due to downsampling or leftover imaging distortion due to interpolation. Thus, with a proper choice of I and D we can approximate any desired ratio of sampling periods to any degree of accuracy. This approach is practical when I and D are small integers. When the ratio I/D involves large values, for example 511/255, or T_0/T is not a ratio of integers, it may be preferable to use techniques developed to change the sampling rate by arbitrary factors. Details of such techniques can be found in Mitra (2006) and Proakis and Manolakis (2007).

12.2 Implementation of multirate systems

In this section we discuss the structures for efficient implementation of sampling rate conversion by a rational factor. The derived structures, besides their widespread applicability in multirate signal processing applications, have some interesting theoretical properties and find use in other areas of signal processing.

12.2.1 Sampling rate compressors and expanders

Since sampling rate compressors and expanders are linear but time-varying, they do not have a system function $H(z) = Y(z)/X(z)$. However, we can express $Y(z)$ as a function that involves some form of $X(z)$.

Sampling rate compressor We use a two-step process as in Example 12.2. We first sample $x[n]$ by multiplying with the periodic sampling train

$$
\delta_D[n] = \sum_{k=-\infty}^{\infty} \delta[n - kD] = \frac{1}{D} \sum_{k=0}^{D-1} W_D^{-kn}, \tag{12.55}
$$

where $W_D \triangleq \exp(-j2\pi/D)$ (see 7.24). The z-transform of the sampled sequence

$$v[n] \triangleq x[n]\delta_D[n] \tag{12.56}$$

is determined by the following steps:

$$V(z) = \sum_{k=-\infty}^{\infty} x[n]\delta_D[n]z^{-n} = \sum_{k=-\infty}^{\infty} x[n] \left[\frac{1}{D} \sum_{k=0}^{D-1} W_D^{-kn} \right] z^{-n}$$

$$= \frac{1}{D} \sum_{k=0}^{D-1} \sum_{k=-\infty}^{\infty} x[n] \left(W_D^k z \right)^{-n} = \frac{1}{D} \sum_{k=0}^{D-1} X(W_D^k z). \tag{12.57}$$

Since $y[n] = v[nD] = x[nD]$, the z-transform of $y[n]$ can be expressed as

$$Y(z) = \sum_{n=-\infty}^{\infty} v[nD]z^{-n} = \sum_{m=-\infty}^{\infty} v[m](z^{1/D})^{-m} = V(z^{1/D}). \tag{12.58}$$

Combining (12.57) and (12.58) we obtain the desired relation

$$y[n] = x[nD] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z) = \frac{1}{D} \sum_{k=0}^{D-1} X(W_D^k z^{1/D}). \tag{12.59}$$

We note that evaluation of (12.59) on the unit circle yields the discrete-time sampling theorem (12.12).

Sampling rate expander The sampling rate expander was defined by

$$y[n] = \begin{cases} x[n/I], & n = 0, \pm I, \pm 2I, \ldots \\ 0. & \text{otherwise} \end{cases} \tag{12.60}$$

The z-transform of the output is given by

$$Y(z) = \sum_{n=-\infty}^{\infty} x[n/I]z^{-n} = \sum_{m=-\infty}^{\infty} x[m]z^{-Im}, \tag{12.61}$$

which implies that

$$Y(z) = X(z^I). \tag{12.62}$$

Evaluating (12.62) on the unit circle yields the frequency-domain relationship given in (12.38a).

(a)   (b)

(c)   (d)
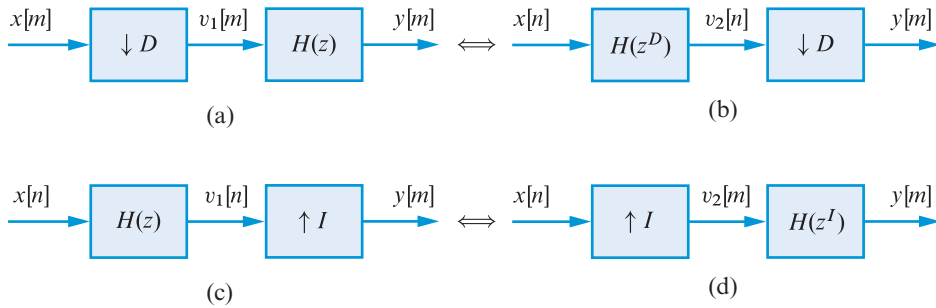
Figure 12.18 Multirate identities: two equivalent systems (a and b) for downsampling and two equivalent systems (c and d) for upsampling. It is possible to interchange the filter with a compressor or expander if we properly modify the filter.

12.2.2   The multirate identities

We next derive two identities, known as *multirate identities*, which are widely used to understand and simplify the operation of multirate systems.

Interchange of filtering with downsampling The output of the system in Figure 12.18(a) is described by

$$Y(z) = H(z)V_1(z) = H(z)\frac{1}{D}\sum_{k=0}^{D-1}X(z^{1/D}W_D^k). \tag{12.63}$$

For the system in Figure 12.18(b) we have

$$Y(z) = \frac{1}{D}\sum_{k=0}^{D-1}V_2(z^{1/D}W_D^k) = H(z)\frac{1}{D}\sum_{k=0}^{D-1}X(z^{1/D}W_D^k), \tag{12.64}$$

because $V_2(z) = H(z^D)X(z)$ and $W_D^{kD} = 1$. From (12.63) and (12.64) we conclude that the two systems are equivalent. Thus, we can interchange the order of downsampling and filtering if we upsample the impulse response of the filter.

Interchange of filtering with upsampling The output of the system in Figure 12.18(c) is given by

$$Y(z) = V_1(z^I) = H(z^I)X(z^I). \tag{12.65}$$

The output of the system in Figure 12.18(d) is

$$Y(z) = H(z^I)V_2(z) = H(z^I)X(z^I). \tag{12.66}$$

Comparison of (12.65) and (12.66) shows that the two structures are equivalent. Thus, we can interchange the order of filtering and upsampling if we upsample the impulse response of the filter.

Polyphase filter structures

Polyphase filter structures are widely used to simplify implementation of decimators and interpolators; however, they are useful in their own right. Consider an FIR filter with length $N = ML$. The polyphase decomposition breaks the impulse response into M subsequences of length L. The basic idea is best illustrated by means of a simple example. For $N = 6$ and $M = 2$, we group together even and odd terms as follows:

$$
\begin{aligned}
H(z) &= h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} + h[5]z^{-5} \\
&= \left(h[0] + h[2]z^{-2} + h[4]z^{-4} \right) + z^{-1} \left(h[1] + h[3]z^{-2} + h[5]z^{-4} \right).
\end{aligned} \tag{12.67}
$$

If we define the following subfilters

$$
P_0(z) \triangleq h[0] + h[2]z^{-1} + h[4]z^{-2}, \tag{12.68a}
$$
$$
P_1(z) \triangleq h[1] + h[3]z^{-1} + h[5]z^{-2}, \tag{12.68b}
$$

we can express $H(z)$ as follows:

$$
H(z) = P_0\left(z^2\right) + z^{-1}P_1\left(z^2\right). \tag{12.69}
$$

For $M = 3$ we obtain the following decomposition:

$$
H(z) = P_0\left(z^3\right) + z^{-1}P_1\left(z^3\right) + z^{-2}P_2\left(z^3\right), \tag{12.70}
$$

where the polyphase components are given by

$$
P_0(z) \triangleq h[0] + h[3]z^{-1}, \tag{12.71a}
$$
$$
P_1(z) \triangleq h[1] + h[4]z^{-1}, \tag{12.71b}
$$
$$
P_2(z) \triangleq h[2] + h[5]z^{-1}. \tag{12.71c}
$$

We note that when N, the length of the impulse response, is a composite number there are multiple polyphase decompositions. If there is possibility of confusion, we should include M into the notation.

In general, the impulse response of the kth subfilter is obtained by downsampling the shifted sequence $h[n + k]$ by a factor M, that is,

$$
p_k[n] \triangleq h[nM + k], \quad k = 0, 1, \ldots, M - 1. \tag{12.72}
$$

Therefore, we have

$$
H(z) = \sum_{k=0}^{M-1} z^{-k}P_k\left(z^M\right), \tag{12.73}
$$

where

$$P_k(z) = \sum_{n=0}^{L-1} p_k[n]z^{-n}. \tag{12.74}$$

In MATLAB the polyphase decomposition can easily be obtained using the `reshape` function. Assuming that N, the length of $h[n]$, is a multiple of M, we have

$$\text{P=reshape(h,M,length(h)/M)}. \tag{12.75}$$

If N is not a multiple of M, we append $h[n]$ with the proper number of zeros. The rows of the matrix P contain the polyphase components of the impulse response.

Using (12.70) for $M = 3$, we note that the z-transform of the output sequence $y[n]$ can be expressed into two equivalent forms as follows:

$$\begin{aligned} Y(z) &= H(z)X(z) \\ &= P_0(z^3)X(z) + z^{-1}P_1(z^3)X(z) + z^{-2}P_2(z^3)X(z), \tag{12.76a} \\ &= P_0(z^3)X(z) + z^{-1}\{P_1(z^3)X(z) + z^{-1}[P_2(z^3)X(z)]\}. \tag{12.76b} \end{aligned}$$

The first expression (12.76a) leads to the direct form polyphase structure shown in Figure 12.19(a); the second expression (12.76) leads to the transposed form polyphase structure shown in Figure 12.19(b). Polyphase decomposition for IIR filters is beyond the scope of this text; see Vaidyanathan (1993).

12.2.4 Polyphase structures for decimation and interpolation

The most important application of polyphase structures is in implementation of computationally efficient decimators and interpolators.

(a)

(b)
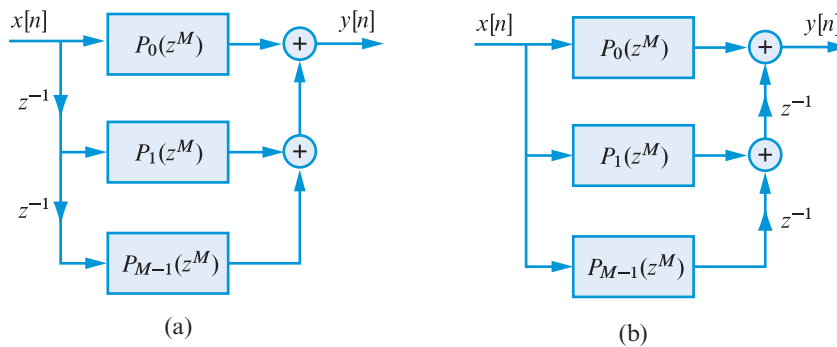
Figure 12.19 (a) Realization of the direct form polyphase structure, and (b) the transposed form polyphase structure; both structures are shown for $M = 3$.
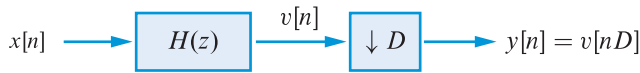
Figure 12.20 Decimation system.

Consider the decimation system shown in Figure 12.20. We first express the filter $H(z)$ using the polyphase structure (12.73), (12.74) with $M = D$. The result is as follows:

$$H(z) = \sum_{k=0}^{M-1} h[k]z^{-k} = \sum_{k=0}^{D-1} P_k(z)z^{-k}, \tag{12.77a}$$

$$P_k(z) = \sum_{n=0}^{L-1} p_k[n]z^{-n}, \tag{12.77b}$$

$$p_k[n] = h[nD + k], \quad k = 0, 1, \ldots, D - 1 \tag{12.77c}$$

where we have assumed that $M = LD$; otherwise, we pad $h[n]$ with zeros. If we replace $H(z)$ with the polyphase structure in Figure 12.19(a) and we recall that downsampling is a linear operation (hence downsampling commutes with addition), we obtain the system in Figure 12.21(a). Applying the multirate identity for downsampling to this system leads to the structure in Figure 12.21(b). The output of the polyphase decimator is

$$y[n] = \sum_{k=0}^{D-1} y_k[n] = \sum_{k=0}^{D-1} p_k[n] * x_k[n], \tag{12.78}$$

where the inputs to the parallel bank of polyphase filters are given by

$$x_k[n] = x[nD - k], \quad k = 0, 1, \ldots, D - 1 \tag{12.79}$$
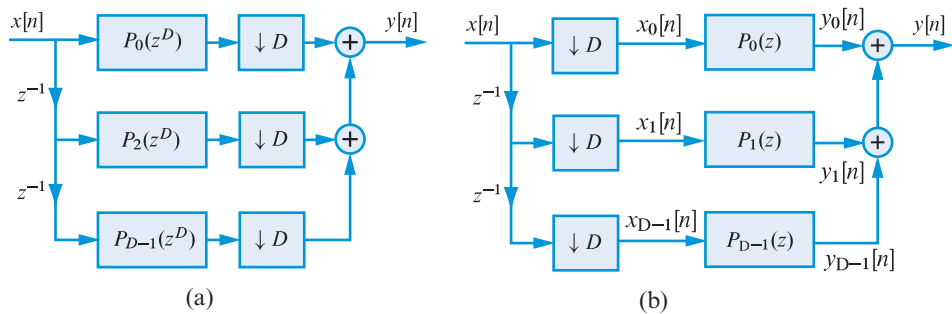
Figure 12.21 Polyphase implementation of a decimation system before applying the downsampling identity (a), and after applying the downsampling identity (b).

or more explicitly

$$x_0[n] = \{x[0], x[D], x[2D], x[4D], \ldots\}$$

$$x_1[n] = \{0, x[D-1], x[2D-1], x[4D-1], \ldots\}$$

$$\vdots \qquad\qquad\qquad \vdots$$

$$x_{D-1}[n] = \{0, x[1], x[D+1], x[2D+1], \ldots\}$$

From this analysis, we can easily see that the downsampling operations in Figure 12.21(b) can be replaced by the commutator structure shown in Figure 12.22. The commutator starts at $n = 0$ feeding the sample $x[0]$ to $P_0(z)$. The next sample $x[1]$, at time $n = 1$, goes to $P_{D-1}(z)$. The commutator continues its operation rotating counterclockwise at the input sampling rate. However, the polyphase filter bank operates at the lower output sampling rate. We emphasize that the commutator is a conceptual tool, which can easily be implemented in software. A MATLAB function of a polyphase FIR decimator, `ppdecim`, is given in Figure 12.23. The bank of polyphase filters requires the same number of computations per output sample as the direct-form filter; however, it operates at a rate $1/D$ lower than that of the input sequence.

A similar polyphase structure can be obtained for the interpolation system shown in Figure 12.24. Replacing the filter $H(z)$ by the polyphase structure in Figure 12.19(b) for $M = I$, yields the polyphase structure in Figure 12.25(a). If we next apply the multirate identity for upsampling, the structure in Figure 12.25(a) takes the form shown in in Figure 12.25(b). The output of each polyphase filter is given by

$$p_k[n] = h[nI + k], \quad k = 0, 1, \ldots, I-1 \tag{12.80a}$$

$$y_k[n] = p_k[n] * x[n], \tag{12.80b}$$

$$y[m] = y_k[n]. \ m = nI + k \tag{12.80c}$$

The upsampling and delay operations in Figure 12.25(b) can be replaced by the commutator structure shown in Figure 12.26. We note that while the filtering operation takes place at the lower input sampling rate, the commutator picks-up sequentially samples from the
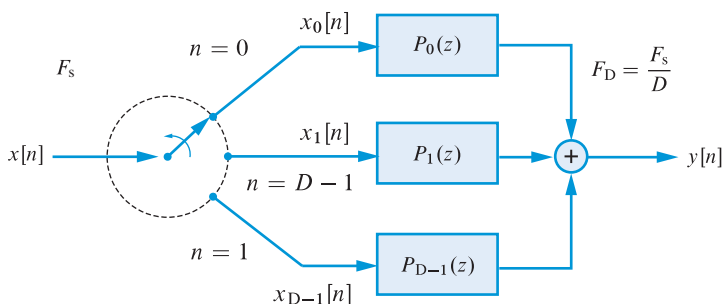
Figure 12.22 Polyphase decimator with a commutator instead of delays.

```
function y=ppdecim(h,x,D);
% Polyphase decomposition of h[n]
Lh=length(h); Lp=floor((Lh-1)/D)+1;
p=zeros(1,Lp*D); p(1:Lh)=h; p=reshape(p,D,Lp);
% Polyphase decomposition of x[n]
Lx=length(x); Ly=floor((Lx+Lh-2)/D)+1;
K=floor((Lx+D-2)/D)+1;
v=[zeros(1,D-1),reshape(x,1,Lx),zeros(1,D*K-Lx-D+1)];
v=flipud(reshape(v,D,K));
% Polyphase decimator implementation
y=zeros(1,K+Lp-1);
for m=1:D, y=y+conv(p(m,:),v(m,:)); end
y=y(1:Ly);
```

Figure 12.23 MATLAB implementation of a polyphase FIR decimator.
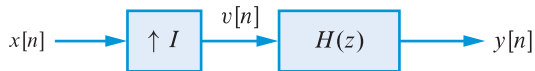
Figure 12.24 Interpolation system.
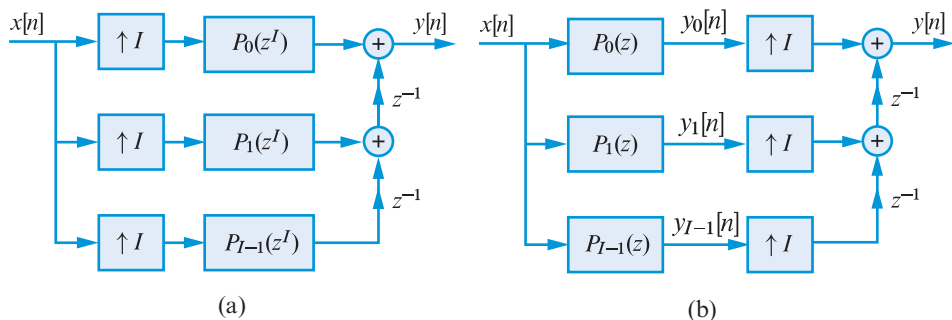
Figure 12.25 Polyphase interpolation structure before (a) and after (b) the application of the multirate identity for upsampling.

filtered sequences $y_0[n], y_1[n], \ldots, y_{I-1}[n]$, at the higher output sampling rate. The commutator starts with $y_0[n]$ and continues counterclockwise; for each input sample it does one full rotation to pick-up I interpolation samples from the outputs of the polyphase bank. The advantage of the polyphase interpolation structure is that the filter operates at the lower input sampling rate; the number of computations per input sample is the same as that for the direct form implementation (see Tutorial Problem 14). A MATLAB function for an FIR polyphase interpolator, ppinterp is given in Figure 12.27.

Efficient polyphase structures for sampling rate conversion by a rational factor, which is somewhat more involved, are discussed in Crochiere and Rabiner (1981), Hsiao (1987),
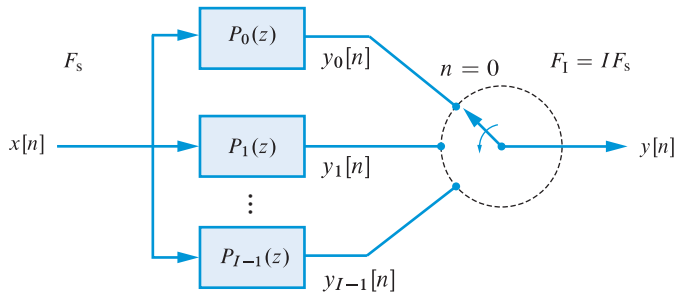
Figure 12.26 Polyphase interpolator with a commutator instead of delays.

```
function y=ppinterp(h,x,I)
% Polyphase decomposition of h[n]
Lh=length(h); Lp=floor((Lh-1)/I)+1;
p=flipud(reshape([reshape(h,1,Lh),zeros(1,Lp*I-Lh)],I,Lp));
% Polyphase interpolator implementation
Lx=length(x); Ly=Lx*I+Lh-1;
Lv=Lx+Lp;
v=zeros(I,Lv);
for i=1:I, v(i,1:Lv-1)=conv(p(i,:),x); end
y=reshape(flipud(v),1,I*Lv);
y=y(1:Ly);
```

Figure 12.27 MATLAB implementation of a polyphase FIR interpolator.

Vaidyanathan (1993), and Fliege (1994). A program for the approach described by Crochiere and Rabiner (1981) is provided in Crochiere (1979).

MATLAB functions for rational rate conversion The MATLAB SP toolbox provides two functions for rational sampling rate conversion. The function

```
y=resample(x,I,D)
```

resamples the signal in array `x` at `I/D` times the original sampling rate to obtain `y`. The resulting resampled array `y` has the length given by `ceil(I/D)*length(x)`. The function `resample` also has additional input and output parameters for which the SP toolbox manual should be consulted. The second function

```
y = upfirdn(x,h,I,D)
```

performs a rational rate conversion using three operations. It first upsamples the input signal array `x` by the factor `I`, then performs FIR filtering on the upsampled signal using the impulse response in `h`, and finally downsamples the result by a factor `D` to obtain the resampled signal in array `y`. Using a properly designed FIR filter as discussed in Section 12.3, we have complete control over the rational rate conversion.

12.3   Filter design for multirate systems

The ideal lowpass filters required for practical sampling rate conversion are usually approximated by FIR filters designed using the techniques discussed in Chapter 10. In this section we discuss some special filters that are particularly useful in multirate systems; however, these filters also have uses in other applications.

12.3.1   Half-band and Kth-band (Nyquist) FIR filters

Decimation and interpolation by a factor of two require lowpass filters with cutoff frequency $\omega_c = \pi/2$, that is, *ideal half-band filters*. The impulse response of this ideal filter is

$$h[n] = \frac{\omega_c}{\pi} \left. \frac{\sin \omega_c(n-\alpha)}{\omega_c(n-\alpha)} \right|_{\omega_c = \pi/2} = \begin{cases} 1/2, & n = \alpha \\ 0. & n-\alpha = \pm 2, \pm 4, \ldots \end{cases} \qquad (12.81)$$

To simplify the discussion we assume $\alpha = 0$, that is, we consider noncausal zero-phase filters with real frequency response function $H(e^{j\omega})$. In this case, we have

$$h[0] = 1/2, \quad h[2n] = 0. \quad n = \pm 1, \pm 2, \ldots \qquad (12.82)$$

Thus, the polyphase representation of (12.77a) is given by

$$H(z) = P_0(z^2) + z^{-1}P_1(z^2) = \tfrac{1}{2} + z^{-1}P_1(z^2), \qquad (12.83)$$

where $P_0(z) = \sum_n h[2n]z^{-n}$ and $P_1(z) = \sum_n h[2n+1]z^{-n}$. In general, any filter with $h[-n] = h[n]$ or $H(e^{-j\omega}) = H(e^{j\omega})$ that satisfies (12.82) is called a *half-band filter*. Using (12.83) we can easily derive the property

$$H(z) + H(-z) = 1. \qquad (12.84)$$

Evaluating this relationship on the unit circle yields

$$H(e^{j\omega}) + H(e^{j(\omega-\pi)}) = 1. \qquad (12.85)$$

If we change the variable ω to $\pi/2 + \theta$ we obtain the following relation:

$$H(e^{j(\pi/2+\theta)}) - \tfrac{1}{2} = -\left[H(e^{j(\pi/2-\theta)}) - \tfrac{1}{2}\right]. \qquad (12.86)$$

Figure 12.28 shows the impulse response and frequency response of an Mth-order FIR half-band filter. First, we note that the $M/2$ zeros in the impulse response reduce the amount of
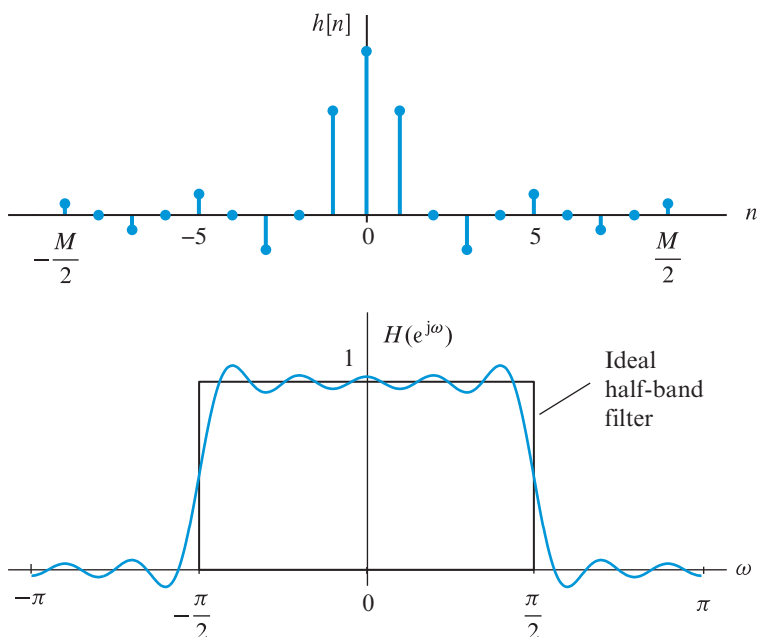
Figure 12.28 Impulse response and frequency response of a half-band FIR filter with order $M = 18$. Notice the odd symmetry of $H(e^{j\omega})$ about $\omega = \pm\pi/2$.

computations by almost one-half. Second, we note that $H(e^{j\omega})$ has odd symmetry about $\omega = \pm\pi/2$. The implications of this symmetry for a practical lowpass filter are: (a) the peak errors δ_1 and δ_2 are equal, and (b) the band-edges ω_p and ω_s are symmetric with respect to $\pi/2$. Hence, we have

$$\delta_p = \delta_s, \tag{12.87a}$$

$$\omega_p = \pi - \omega_s. \tag{12.87b}$$

From Figure 12.28 we can easily see that $M/2$ must be an odd number. Thus, $M/2 = 2p-1$ or $M = 4p-2$, where p is a positive integer. We can easily derive a causal filter by delaying $h[n]$ by $M/2$ samples.

The *Kth band* or *Nyquist filter*, which is a natural extension of the half-band filter, is defined as a zero-phase FIR whose impulse response satisfies the condition

$$h[n] = \begin{cases} 1/K, & n = 0 \\ 0. & n = \pm K, \pm 2K, \dots \end{cases} \tag{12.88}$$

The impulse response of the ideal Kth-band filter is given by (12.81) with $\omega_c = \pi/K$. The polyphase decomposition of a Nyquist filter is given by the generalization of (12.83)

$$H(z) = \frac{1}{K} + \sum_{k=1}^{K-1} z^{-k} P_k(z^K), \tag{12.89}$$

and similarly (12.84) can be generalized to (see Tutorial Problem 10)

$$\sum_{k=0}^{K-1} H(zW_K^k) = 1,$$ (12.90)

where $W_K = e^{-j2\pi/K}$. If we set $z = e^{j\omega}$, we obtain the relation

$$\sum_{k=0}^{K-1} H(e^{j(\omega-2\pi k/K)}) = 1.$$ (12.91)

Thus, the sum of K copies of the frequency response of a Nyquist filter, shifted successively by $2\pi/K$ rads, is equal to unity.

The design of Nyquist filters, which are Type I FIR filters, using windowing techniques is straightforward, as long as the window has even symmetry, odd length $N = M + 1$, and the center coefficient is one. We can also use frequency sampling with the smooth transition band approach discussed in Section 10.3.

We conclude by presenting an efficient approach to the design of equiripple half-band filters using the Parks–McClellan algorithm and a trick introduced by Vaidyanathan and Nguyen (1987). This approach consists of the following steps:

1. Given the specifications ω_s, A_p, and A_s of the half-band filter, obtain the parameters δ_p, δ_s, and ω_p so that they satisfy the design requirements and the constraints of the half-band filter. That is,

$$\delta \triangleq \min\left(\delta_p, \delta_s\right), \quad \omega_p = \pi - \omega_s.$$ (12.92)

2. Design a single band Type II FIR filter $G(z)$ of order $M/2 = 2p - 1$ (odd) with $\tilde{\omega}_p = 2\omega_p$, $\tilde{\omega}_s = \pi$, and $\tilde{\delta} = 2\delta$ using the Parks–McClellan algorithm. Since $G(z)$ is Type II, the frequency response $G(e^{j\omega})$ is equal to zero at $\omega = \pi$.

3. Scale the impulse response $g[n]$ by one-half, upsample the result by a factor of two, and set the middle coefficient to $1/2$. The result is an impulse response $h[n]$ with system function $H(z)$ given by

$$H(z) = \frac{1}{2}\left[z^{-M/2} + G(z^2)\right],$$ (12.93)

   which is a half-band filter with passband cutoff frequency ω_p.

This design procedure is illustrated in the following example.

Example 12.5 Half-band filter design

We want to design a half-band lowpass filter with stopband edge of $\omega_s = 0.55\pi$, passband ripple of 0.1 dB and stopband attenuation of 50 dB. We follow the above three-step approach.

Step-1 Obtain ripple parameters and passband edge. Using $A_p = 0.1$ db and $A_s = 50$ dB, the corresponding ripple values are 0.0058 and 0.0032, respectively. Hence we set

$\delta = \min(0.0058, 0.0032) = 0.0032$. The passband edge is set at $\omega_p = \pi - \omega_s = 0.45\pi$.

```
>> omegas = 0.55*pi; Ap = 0.1; As = 50;
>> deltap = (10\^(Ap/20)-1)/(10\^(Ap/20)+1);
>> deltas = (1+deltap)/(10^(As/20));
>> delta = min(deltap,deltas); omegap = pi - omegas;
```

Step-2 Design a lowpass filter using the Parks–McClellan method. Using $\tilde{\delta} = 2\delta$ and $\tilde{\omega}_p = 2\omega_p$, we obtain filter order M from the `firpmord` function and set it to $M = 4p - 2$ for the next smallest integer p as shown below:

```
>> f = [2*omegap,pi]/pi; A = [1,0]; dev = 2*[delta,delta];
>> [M,fo,Ao,W] = firpmord(f,A,dev); M = ceil((M+2)/4)*4-2;
>> M
M
   = 46
```

Thus $M = 46$. We now design a lowpass filter using $M/2$ and check for the maximum ripple.

```
>> [g,delta] = firpm(M/2,fo,Ao); delta
delta
   = 0.0069
>> M = M+4; [g,delta] = firpm(M/2,fo,Ao); delta
delta
   = 0.0055
```

The maximum ripple is more than $2\delta = 0.0064$, Hence we increase the order by 4 (so that $M = 4p-2$) to $M = 50$ and obtain the design with an acceptable maximum ripple of 0.0055 in the impulse response $g[n]$.

Step-3 Determine the half-band filter impulse response. Finally, we scale the impulse $g[n]$ by $1/2$, upsample by a factor of 2, and set the sample at $M/2$ to $1/2$.

```
>> h = upsample(0.5*g,2); h(M/2+1) = 0.5; h = h(1:M+1);
```

to obtain the desired impulse response $h[n]$.

Figure 12.29 shows the amplitude and log-magnitude responses of the designed filter. As expected, the amplitude response is odd with respect to the point $(\pi/2, 1/2)$ and the log-magnitude response down more than 50 dB at $\omega_s = 0.55\pi$. ∎

More information about the properties and design of Kth band filters can be found in Mintzer (1982).

12.3.2   Multistage decimation and interpolation

Decimation by an integer factor D requires an FIR filter with cutoff frequency $\omega_c = \pi/D$. For large D, the passband of the filter becomes very narrow, which requires an even
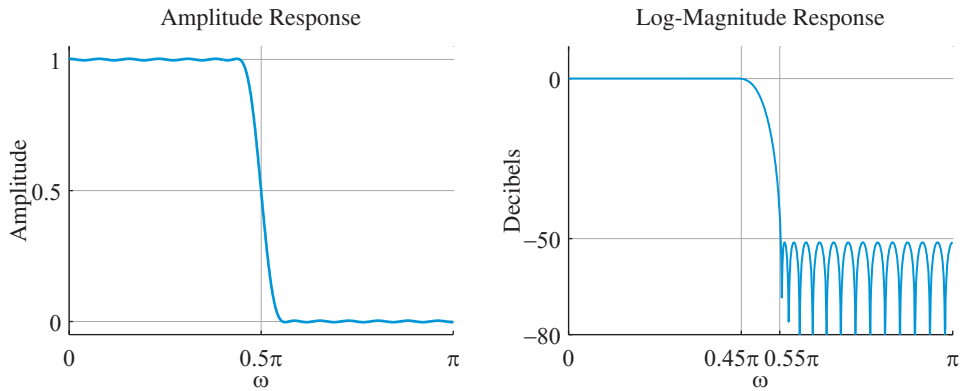
Figure 12.29 Frequency response plots of the half-band filter designed in Example 12.5.
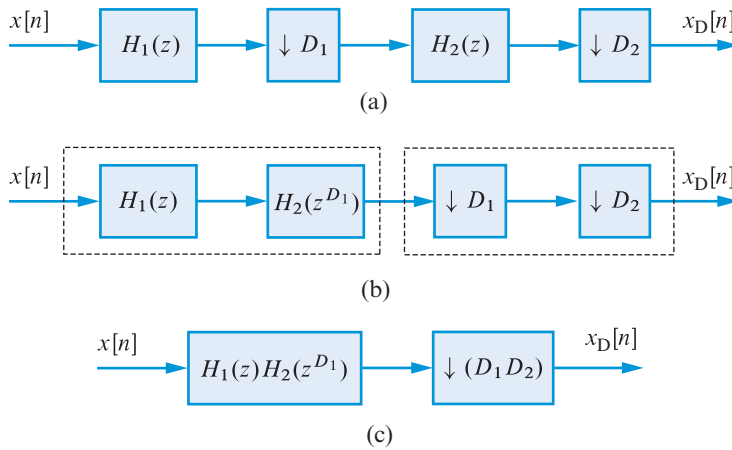
Figure 12.30 (a–c) Conversion of a two-stage decimation system to an equivalent one-stage system using the multirate identity for downsampling.

smaller transition band $\Delta\omega$. Since for FIR filters the order M increases with $1/\Delta\omega$, large decimation factors require long FIR filters with very short transition bands. Such filters are difficult to design and require a large number of computations. An efficient way to avoid these problems is to use a multistage approach. To this end, consider the two-stage decimation system in Figure 12.30(a) where the overall decimation factor is $D = D_1 D_2$. We note that the cutoff frequency $\omega_c = \pi/D$ for a single stage decimator is much smaller than the cutoff frequencies $\omega_c^{(1)} = \pi/D_1$ of $H_1(z)$ or $\omega_c^{(2)} = \pi/D_1$ of $H_2(z)$. Therefore, the filters $H_1(z)$ and $H_2(z)$ are easier to design and have shorter lengths.

If we interchange the order of downsampler by D_1 with the lowpass filter $H_2(z)$ using the the multirate identity for downsampling, we obtain the system in Figure 12.30(b). Combining the two filters and the two downsamplers yields the equivalent single-stage decimation system in Figure 12.30(c). The equivalent single-stage decimator has a factor $D = D_1 D_2$

and a lowpass filter with system function

$$H(z) = H_1(z)H_2(z^{D_1}).$$ (12.94)

These fundamental ideas are illustrated in the following example.

Example 12.6 Two-stage decimation

Consider a single-stage decimation system in which the high sampling rate of $F_H = 100$ Hz is to be reduced to the low rate of $F_L = 10$ Hz using $D = 10$. We will need a lowpass filter $H(z)$ with a cutoff frequency $\omega_c = \pi/D = 0.1\pi$, which is a narrowband filter requiring an even narrower transition band. Let the specifications for this filter be: $\omega_p = 0.09\pi$, $\omega_s = 0.1\pi$, $A_p = 0.1$ dB, and $A_s = 50$ dB. An equiripple lowpass filter, designed using the Parks–McClellan algorithm, has the order $M = 489$ and operates at a rate of 100 Hz. The computation complexity C_D of a decimation system can be defined as the number of multiplications per second. Using the polyphase implementation, this complexity is given by the product of the filter length and the sampling rate of the filter divided by D. Thus, for the single-stage decimation system of the problem, the computational complexity is given by $C_D = (M + 1)F_H/D = 4900$ mults/s.

To reduce the design and computational complexity we implement the decimation system using the two-stage approach of Figure 12.30(a) in which $D_1 = 5$ and $D_2 = 2$. Thus we have to design two filters, $H_1(z)$ and $H_2(z)$, in Figure 12.30(b). According to Figure 12.30(c), we have a cascade of $H_1(z)$ and the upsampled filter $H_2(z^5)$. In the worst case, the passband ripple of $H(z)$ is equal to the sum of the ripples of the cascaded filters. We divide A_p equally between the two filters. In the stopband, we want the overall ripple value to be at least as much as each of the $H_1(z)$ and $H_2(z^5)$. Hence we set, for both filters, the same stopband attenuation A_s.

We first design the filter $H_1(z)$ operating at $F_H = 100$ Hz. The specifications for $H_1(z)$ are given by: $\omega_{p_1} = \omega_p = 0.09\pi$, $\omega_{s_1} = \omega_s D_2 = 0.2\pi$, $A_{p_1} = A_p/2 = 0.05$ dB, and $A_s = 50$ dB. An equiripple lowpass filter designed using the Parks–McClellan algorithm has the order $M_1 = 49$. The computational complexity for $H_1(z)$ is given by $C_1 = (M_1 + 1)F_H/D_1 = 1000$.

Next, we design the filter $H_2(z)$ operating at $F_1 \triangleq F_H/D_1 = 20$ Hz. The specifications for $H_2(z)$ are given by: $\omega_{p_2} = \omega_p D_1 = 0.45\pi$, $\omega_{s_2} = \omega_s D_1 = 0.5\pi$, $A_{p_2} = A_p/2 = 0.05$ dB, and $A_s = 50$ dB. An equiripple lowpass filter designed using the Parks–McClellan algorithm has the order $M_2 = 107$ and is operating at a rate of 20 Hz. The computational complexity is given by $C_2 = (M_2 + 1)F_1/D_2 = 1080$ mults/s. Therefore, the total complexity for the two-stage system of Figure 12.30(a) is $C_1 + C_2 = 2080$ mults/s which is considerably less than $C_D = 4900$ mults/s of the single-stage decimation system.

It should be noted that if we implement the system of Figure 12.30(c) instead, then the equivalent filter length would be 591 and the corresponding complexity would be 5850 mults/s which is more than the original single-stage system. Using an implementation called interpolated FIR (IFIR) filter, the computational complexity can be further reduced. This is discussed in Section 12.3.3 and illustrated in Example 12.7.

Figure 12.31 shows log-magnitude responses of the designed decimation filters: (a) single-stage $H(z)$, (b) two-stage $H_1(z)$, (c) two-stage $H_2(z)$, and (d) the equivalent
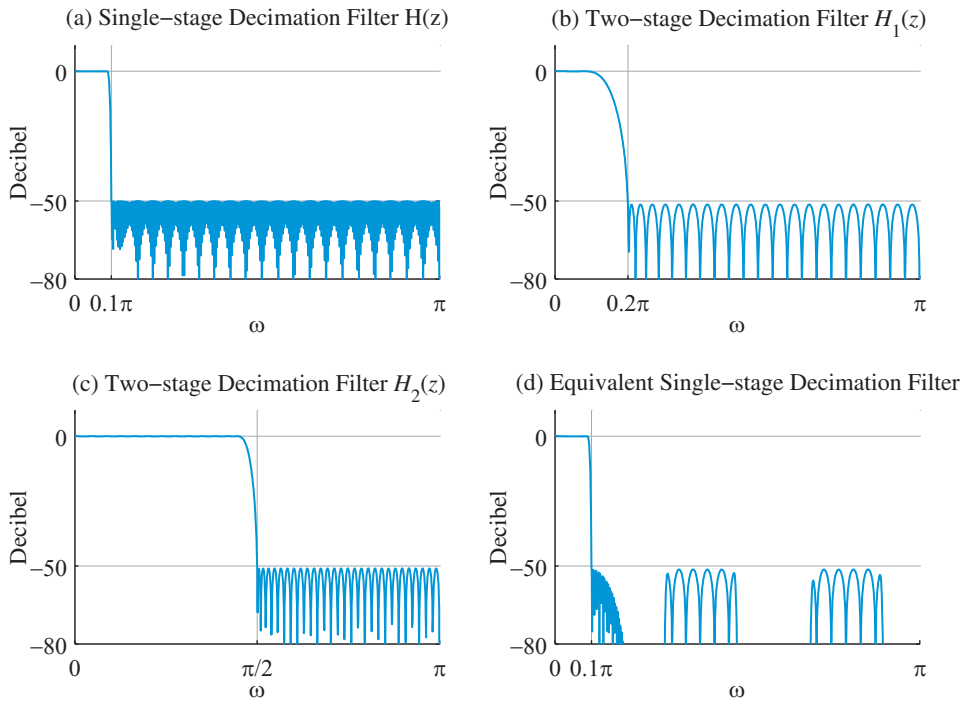
Figure 12.31 Single-stage and two-stage decimation filter response plots in Example 12.6.

single-stage $H_1(z)H_2(z^5)$. The two-stage filters satisfy all the desired specifications and yet require reduced computational complexity. ∎

The same ideas can be used, in conjunction with the multirate identity for upsampling, to derive multistage interpolators. The results, which are shown in Figure 12.32, are derived in Problem 11. Similar ideas can be used for sampling rate change by a rational factor. These ideas can be generalized for multiple stages. In this case, D and I are composite numbers which can be written as products of multiple factors. Since each decomposition leads to a different multistage structure, it is desirable to find a systematic procedure for the design of optimum multistage sampling rate converters. This problem is discussed in detail by Crochiere and Rabiner (1983).

12.3.3   Interpolated FIR (IFIR) filters

To understand the principle of IFIR filters consider a prototype lowpass filter $F(z)$ with cutoff frequency ω_c shown in Figure 12.33(a). If we replace z by z^L we obtain a filter $F_L(z) = F(z^L)$ such that

$$
f_L[n] = \begin{cases} f[n], & n = 0, \pm L, \pm 2L, \dots \\ 0, & \text{otherwise} \end{cases} \xrightarrow{\text{DTFT}} F_L(e^{j\omega}) = F(e^{j\omega L}). \qquad (12.95)
$$

12.3 Filter design for multirate systems
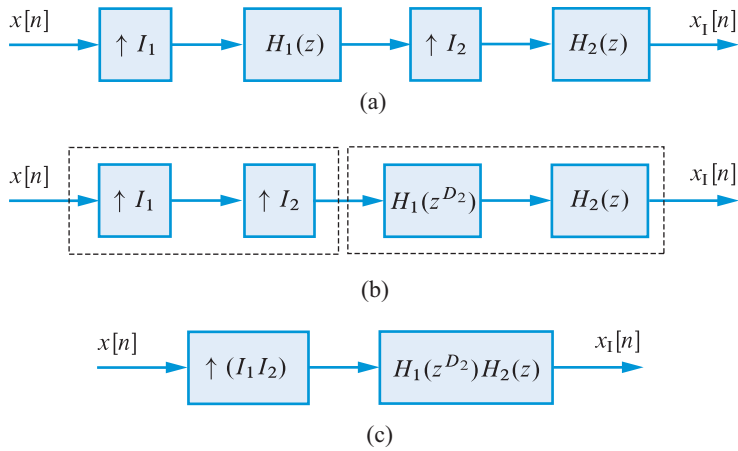
Figure 12.32 Conversion of a two-stage interpolation system to an equivalent one-stage system using the multirate identity for upsampling.
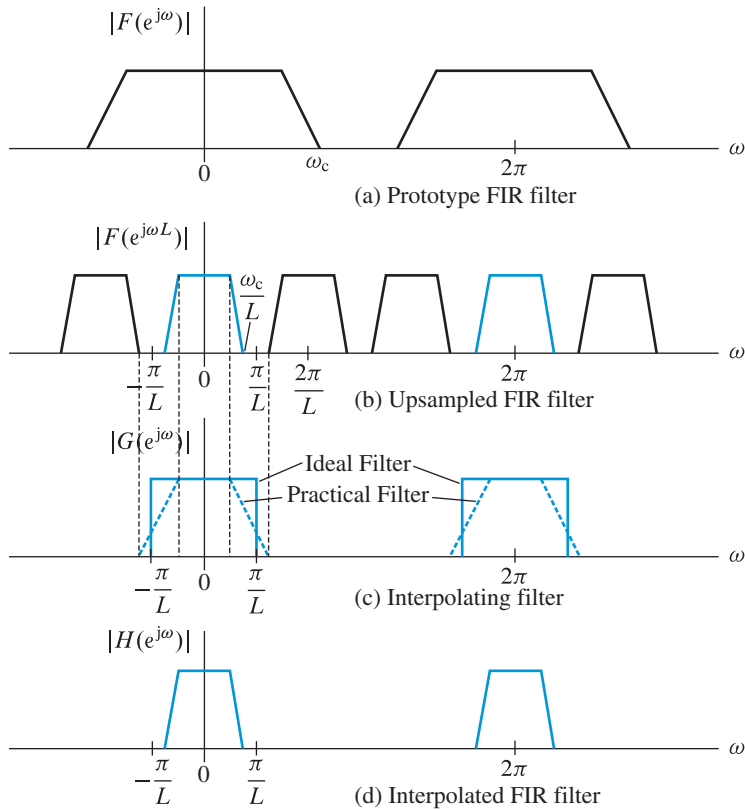
Figure 12.33 (a–d) The interpolated FIR (IFIR) filter concept.

We note that $f_L[n]$ is obtained by upsampling $f[n]$ by a factor L and $F_L(e^{j\omega})$ is constructed by squeezing L periods of $F(e^{j\omega})$ between 0 and 2π. A similar process was used to create comb filters in Section 5.7.3. The frequency response $F_L(e^{j\omega})$ contains $(L-1)$ compressed images of $F(e^{j\omega})$ as shown in Figure 12.33(b). These images can be eliminated by using an ideal interpolating lowpass filter $G(z)$ (or the practical filter as explained in Example 12.7) shown in Figure 12.33(c). The result is a lowpass filter with cutoff frequency $\omega_c \approx \pi/L$ and sharper transition band. The system function of the combined filter is given by

$$H(z) = F(z^L)G(z). \tag{12.96}$$

This process is equivalent to interpolation of the impulse response $f[n]$ using the interpolator in Figure 12.9. This interpretation is responsible for the term *interpolated FIR filter*; these filters were introduced by Neuvo *et al.* (1984). More information about using periodic sub-filters as building blocks to design FIR filters is given by Saramaki (1993).

Example 12.7   IFIR

We want to design a narrowband lowpass filter $H(z)$ with cutoff frequencies $\omega_p = 0.09\pi$ and $\omega_s = 0.1\pi$, passband ripple $A_p = 0.1$ dB and stopband attenuation $A_s = 50$ dB as shown in Figure 12.34(a) where ripples are not shown for clarity. An equiripple lowpass filter, designed using the Parks–McClellan algorithm, has length $L_H = 489$. The log-magnitude response of $H(z)$ is shown in Figure 12.35(a).
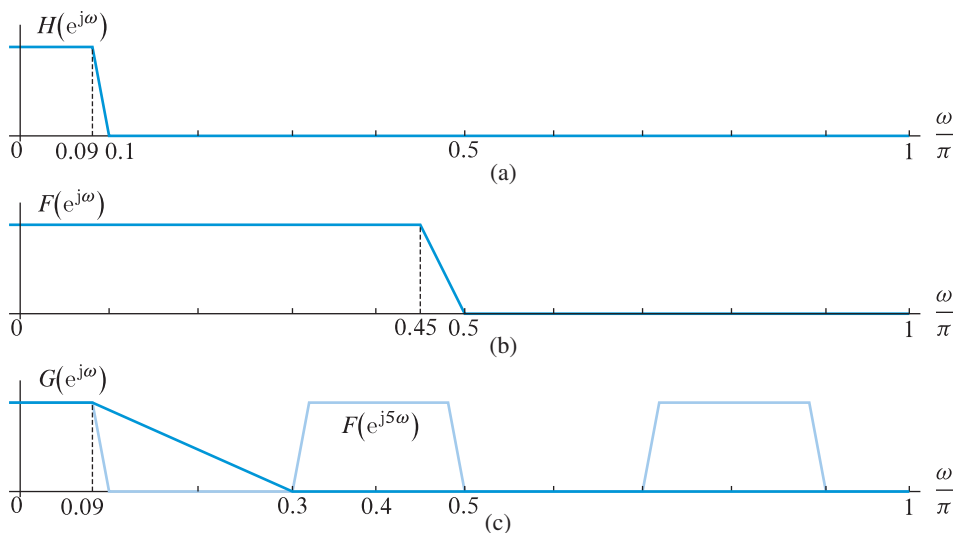
Figure 12.34 Spectra of filters in Example 12.7: (a) narrowband lowpass filter $H(z)$, (b) prototype filter $F(z)$, and (c) interpolating filter $G(z)$.

(a) Given Filter $H(z)$ Response

(b) Filter $G(z)$ Response

(c) Filter $F(z)$ Response

(d) IFIR Filter Response
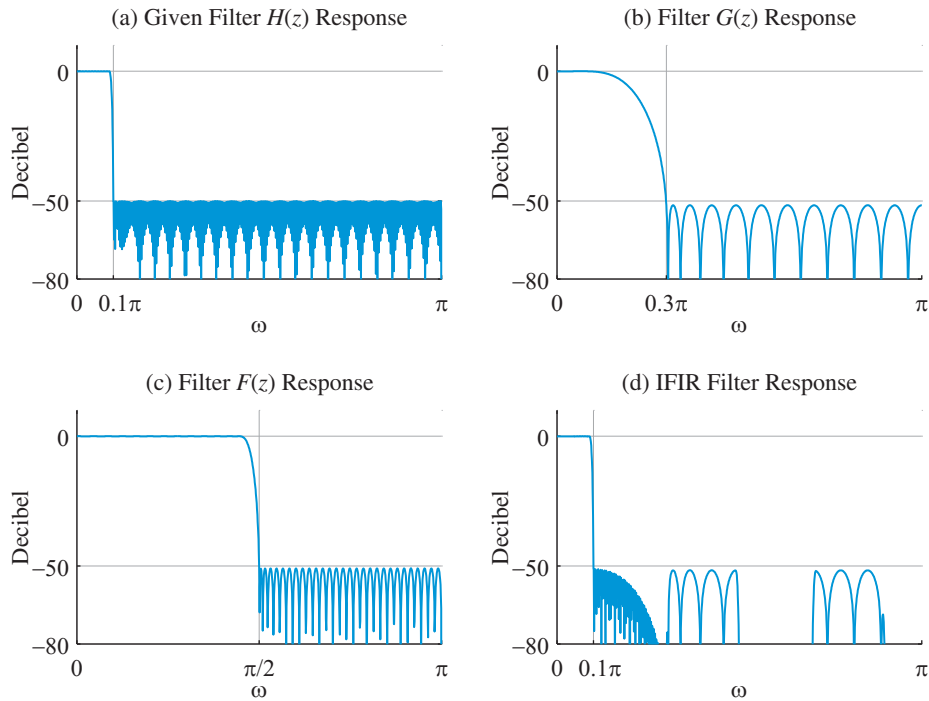
Figure 12.35 Frequency responses of filters designed in Example 12.7.

To reduce the design complexity we implement the narrowband filter $H(z)$ using two filters $G(z)$ and $F(z)$ given in (12.96) using $L = 5$. This implementation involves a cascade of $G(z)$ and the upsampled filter $F(z^L)$. In the worst case, the passband ripple of $H(z)$ is equal to the sum of the ripples of the cascaded filters. We divide A_p equally between the two filters. In the stopband, we want the overall ripple value to be at least as much as each of the $G(z)$ and $F(z^5)$. Hence for both filters we set the same stopband attenuation A_s.

First, we design the prototype filter $F(z)$. As shown in Figure 12.34(b) the specifications for $F(z)$ are given by: $\omega_{pF} = \omega_p L = 0.45\pi$, $\omega_{sF} = \omega_s L = 0.5\pi$, $A_{pF} = A_p/2 = 0.05$ dB, and $A_s = 50$ dB. An equiripple lowpass filter designed using the Parks–McClellan algorithm has length $L_F = 108$. The log-magnitude response of $F(z)$ is shown in Figure 12.35(c). This filter will be upsampled by inserting four zeros between each sample of $f[n]$ to obtain the filter $F(z^5)$.

We next design the interpolation filter $G(z)$. As shown in Figure 12.34(c), the cascaded upsampled filter $F(z^5)$ has the first stopband from $\omega_s = 0.1\pi$ to $2\pi/L - \omega_s = 0.3\pi$ due to its multiple images Hence we can choose 0.3π as the stopband edge for $G(z)$ instead of $\omega_s = 0.1\pi$ to reduce filter order. Thus the specifications for $G(z)$ are given by: $\omega_{pG} = \omega_p = 0.09\pi$, $\omega_{sG} = 0.3\pi$, $A_{pG} = A_p/2 = 0.05$ dB, and $A_s = 50$ dB. An equiripple lowpass filter designed using the Parks–McClellan algorithm has length $L_G = 27$ which is considerably less than what we would obtain with a narrower transition band. The log-magnitude response of $G(z)$ is shown in Figure 12.35(b).

After convolving $g[n]$ with the upsampled $f[n]$ we obtain the impulse response $h_{\text{IFIR}}[n]$ whose log-magnitude response is shown in Figure 12.35(d). A comparison with the response of $H(z)$ shows that the IFIR filter satisfies the given specifications. While $H(z)$ requires 489 coefficients, the IFIR filter which implements the cascade (12.96) requires only $L_G + L_F = 135$ coefficients. ∎

12.4   Two-channel filter banks

A *filter bank* is a collection of filters with a common input or a common output. The two basic types of filter bank are shown in Figure 12.36. The *analysis filter bank* splits a signal $x[n]$ into K signals $v_k[n]$, known as *sub-band signals*, using the analysis filters $H_k(z)$. The *synthesis filter bank* consists of K synthesis filters $G_k(z)$, which combine K signals $s_k[n]$ into a signal $y[n]$. If $s_k[n] = v_k[n]$, for $k = 0, 1, \ldots, K-1$, we would like $y[n]$ to provide an accurate reconstruction of $x[n]$. In general, what we do with the sub-band signals and how we design the filters $H_k(z)$ and $G_k(z)$ depends on the application. In most applications, the sub-band signals are obtained by dividing the spectrum into K separate bands using filters with overlapping bands to prevent gaps in the spectrum. If all sub-bands have the same width, see Figure 12.37, the filter bank is called *uniform*; however, in certain applications it may be preferable to use nonuniform filter banks. Thus, filter banks use lowpass, bandpass, and highpass filters to cover the entire frequency range.

A problem with the analysis filter bank in Figure 12.36 is that the number of output samples is K times the number of input samples. However, if we recall that each band of a uniform bank has width [1] π/K we can decimate each $v_k[n]$ by a factor of less or equal to K without aliasing. We choose the decimation factor equal to K to achieve the maximum computational and storage efficiency. Clearly the processed sub-band signals should be
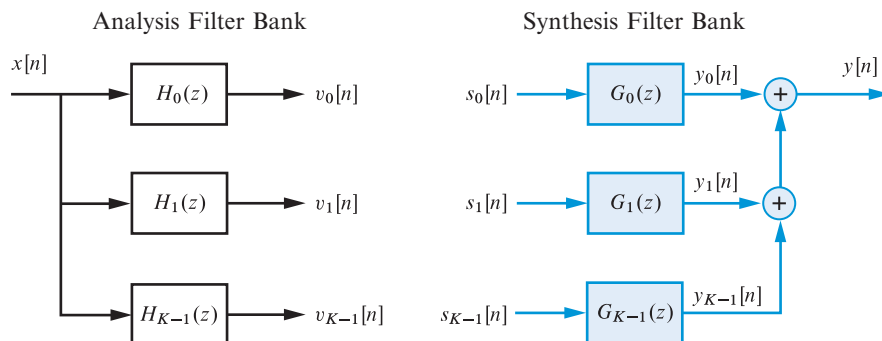
Figure 12.36 Analysis and synthesis filter banks for $K = 3$ channels.

---

[1] A real ideal lowpass filter has bandwidth π/K if $H(e^{j\omega}) = 0$ for $|\omega| > \pi/K$; we use the same definition of bandwidth for complex bandpass filters with one-sided frequency responses.
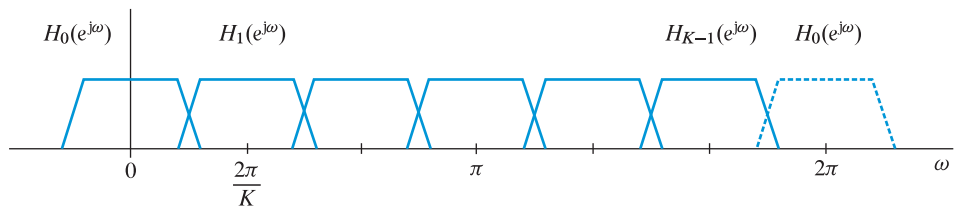
Figure 12.37 Frequency response of a lowpass prototype and the resulting uniform DFT filter bank for $K = 6$. Due to periodicity the filter $H_0(e^{j\omega})$ covers the two ends of the fundamental frequency range.
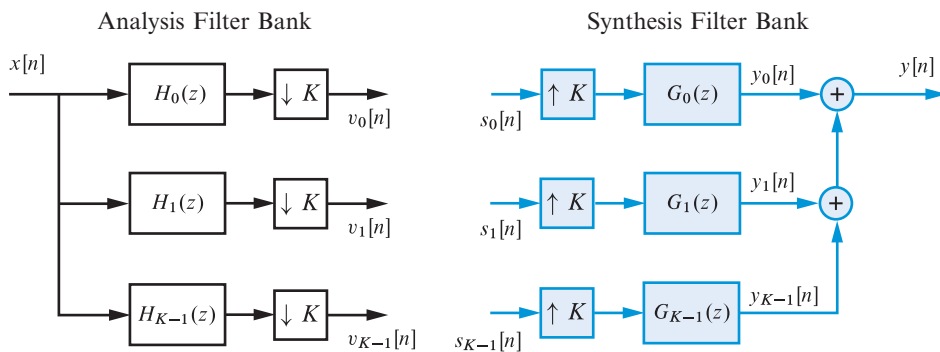
Figure 12.38 A maximally decimated multirate filter bank with $K = 3$ channels.

interpolated by a factor of K before they are combined to form the output signal. These ideas lead to the *maximally decimated multirate filter bank* shown in Figure 12.38. In this section we consider only two-channel uniform maximally decimated multirate filter banks. Multirate filter banks find many applications in audio processing, image processing, and communication systems.

12.4.1   Input-output description

Consider the two-channel filter bank shown in Figure 12.39(a). Using (12.55) and (12.58), the output of the upper channel is described by the following equations:

$$V_0(z) = \frac{1}{2}H_0(z^{1/2})X(z^{1/2}) + \frac{1}{2}H_0(-z^{1/2})X(-z^{1/2}), \qquad (12.97a)$$

$$Y_0(z) = V_0(z^2)G_0(z), \qquad (12.97b)$$

$$Y_0(z) = \frac{1}{2}\big[H_0(z)X(z) + H_0(-z)X(-z)\big]G_0(z). \qquad (12.97c)$$

In a similar way, the output of the lower channel is

$$Y_1(z) = \frac{1}{2}\big[H_1(z)X(z) + H_1(-z)X(-z)\big]G_1(z). \qquad (12.98)$$
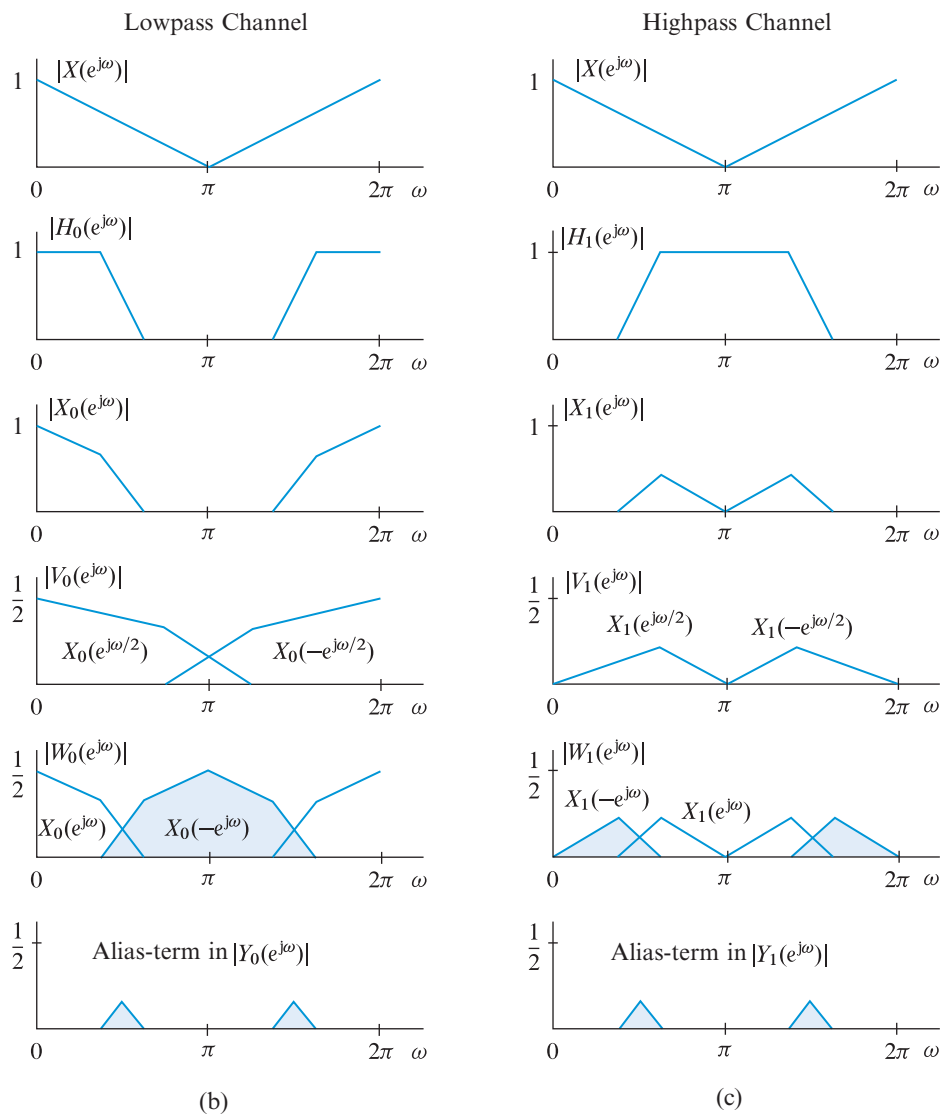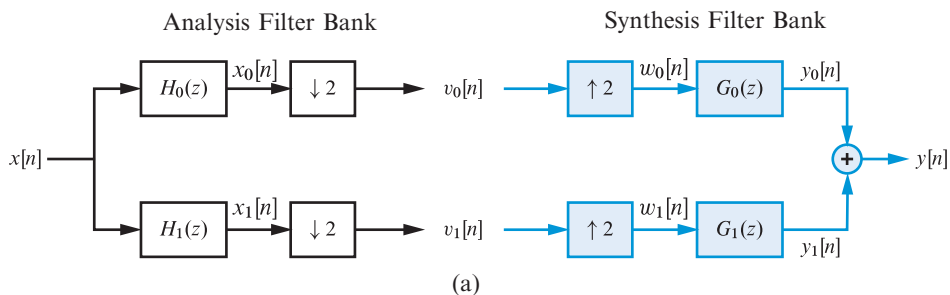
Figure 12.39 (a) Two-channel filter bank. Explanation of the alias generation and cancellation mechanism in the frequency-domain for (b) the lowpass channel and (c) the highpass channel (see Vaidyanathan (1990) or Vaidyanathan (1993)).

The output $y[n] = y_0[n] + y_1[n]$ of the synthesis filter bank is given by

$$Y(z) = \frac{1}{2}\big[T(z)X(z) + A(z)X(-z)\big], \tag{12.99}$$

where

$$T(z) \triangleq H_0(z)G_0(z) + H_1(z)G_1(z), \tag{12.100a}$$

$$A(z) \triangleq H_0(-z)G_0(z) + H_1(-z)G_1(z). \tag{12.100b}$$

The effective system function, $A(z)$, between $X(-z)$ and $Y(z)$ causes aliasing while the effective system function, $T(z)$, between $X(z)$ and $Y(z)$ may cause magnitude and phase distortion. We note that the alias-free filter bank, obtained by enforcing $A(z) = 0$, becomes a linear time-invariant system with system function $T(z)/2$.

To understand the operation of the two-channel filter bank we look at the spectra of the internal signals at the points shown in Figure 12.39(a) using a fictitious input spectrum $X(e^{j\omega})$. In the lowpass channel, shown in Figure 12.39(b), the alias component $X(-e^{j\omega/2})/2$ overlaps with the desired component $X(e^{j\omega/2})/2$. The contribution of $X(-z)$ to $Y_0(z)$ (shaded area) is the alias component, which, in general, overlaps the unshaded area. A similar mechanism for the highpass channel is shown in Figure 12.39(c). We note that the aliasing term is the result of aliasing and imaging components introduced by the downsampling and upsampling systems. The filters $G_0(z)$ (lowpass) and $G_1(z)$ (highpass) are designed to eliminate the aliasing term.

12.4.2 Conditions for perfect reconstruction

The condition for perfect (or distortionless) reconstruction from Section 5.3 is

$$y[n] = Gx[n - n_d] \overset{\mathcal{Z}}{\longleftrightarrow} Y(z) = Gz^{-n_d}X(z). \tag{12.101}$$

From (12.100) we conclude that perfect reconstruction is possible if and only if

$$A(z) = H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0, \tag{12.102a}$$

$$T(z) = H_0(z)G_0(z) + H_1(z)G_1(z) = Gz^{-n_D}, \tag{12.102b}$$

which can be written in matrix form as follows

$$\begin{bmatrix} H_0(z) & H_1(z) \\ H_0(-z) & H_1(-z) \end{bmatrix} \begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \begin{bmatrix} Gz^{-n_D} \\ 0 \end{bmatrix}. \tag{12.103}$$

The condition (12.102a) ensures the elimination of aliasing distortion from the output signal. The condition (12.102b), which requires $T(z)$ to be allpass filter with linear phase,

guarantees the absence of amplitude and phase distortion from the reconstructed signal. We set $G = 2$ to preserve the amplitude of the output signal.

If we have determined the analysis filters, we can obtain the synthesis filters by solving the linear system of equations (12.103). The result is

$$G_0(z) = \frac{2z^{-n_D}}{\Delta_m(z)} H_1(-z), \quad G_1(z) = -\frac{2z^{-n_D}}{\Delta_m(z)} H_0(-z), \tag{12.104}$$

where $\Delta_m(z)$, the determinant of the matrix in (12.103), is given by

$$\Delta_m(z) = H_0(z)H_1(-z) - H_0(-z)H_1(z). \tag{12.105}$$

The solutions in (12.104) exist if $\Delta_m(z) \neq 0$. We note that choosing the synthesis filters according to (12.104) ensures cancellation of aliasing error for any choice of analysis filters. Then, the condition (12.102b) for perfect reconstruction becomes

$$z^{n_D} H_0(z)G_0(z) + z^{n_D} H_1(z)G_1(z) = 2. \tag{12.106}$$

To facilitate the analysis and design processes, we define a *product filter $R(z)$* by

$$R(z) \triangleq z^{n_D} H_0(z)G_0(z). \tag{12.107}$$

Inserting $G_0(z)$ from (12.104) into (12.107), the product filter can be written in terms of the analysis filters as

$$R(z) = \frac{2}{\Delta_m(z)} H_0(z)H_1(-z). \tag{12.108}$$

Since $\Delta_m(-z) = -\Delta_m(z)$ from (12.105), we can express the product $z^{n_d} H_1(z)G_1(z)$ as

$$z^{n_D} H_1(z)G_1(z) = \frac{-2}{\Delta_m(z)} H_0(-z)H_1(z) = R(-z). \tag{12.109}$$

The last two expressions make it possible to express the perfect reconstruction condition (12.106) in terms of a single product filter $R(z)$ using the relation

$$R(z) + R(-z) = 2. \tag{12.110}$$

The product filter $R(z)$ plays a crucial role in analyzing and designing filter banks.

Perfect reconstruction condition (12.110) imposes some critical constraints on the structure of $R(z)$. Indeed, using the following polyphase decomposition of $R(z)$:

$$R(z) = R_0(z^2) + z^{-1}R_1(z^2), \tag{12.111}$$

the perfect reconstruction condition (12.110) yields

$$R_0(z^2) + z^{-1}R_1(z^2) + R_0(z^2) - z^{-1}R_1(z^2) = 2, \qquad (12.112)$$

which implies that $R_0(z^2) = 1$. Therefore, the product filter in (12.111) is given by

$$R(z) = 1 + z^{-1}R_1(z^2). \qquad (12.113)$$

Comparing (12.113) with (12.83) we conclude that $R(z)$ *must be a half-band filter*. Constructing a function $R(z)$ that satisfies (12.113) is not difficult. However, to obtain a useful filter bank we have to impose additional requirements.

In conclusion, to design a two-channel perfect reconstruction filter bank, it is necessary and sufficient to obtain an $R(z)$ satisfying (12.113), perform the factorization $R(z) = z^{n_D}H_0(z)G_0(z)$, and assign the remaining filters from (12.104). The choice of $R(z)$ and the particular factorization taken determine the properties of the filter bank. There are two cases of primary interest:

1. The product filter $R(z) = H(z)H(z^{-1})$ is the z-transform of an autocorrelation sequence. This requires the design of a single filter and leads to what is known as *orthogonal* or *para-unitary* filter banks.
2. The product filter $R(z) = H_0(z)G_0(z)$ is the z-transform of a correlation sequence. This requires the design of two filters and leads to the more general *bi-orthogonal* filter banks. These banks are outside the scope of this book.

In many practical applications we prefer FIR filters or FIR filters with linear phase. The enforcement of these constraints may result in unrealizable or inadequate perfect reconstruction filter banks. In such cases, the solution is to use a satisfactory "near-perfect reconstruction" filter bank (see Section 12.4.4).

12.4.3 Perfect reconstruction orthogonal FIR filter banks

From (12.104) and (12.105) we conclude that, in general, the synthesis filters $G_0(z)$ and $G_1(z)$ will be IIR even if the analysis filters $H_0(z)$ and $H_1(z)$ are FIR. However, if $H_0(z)$ and $H_1(z)$ are FIR filters and the determinant satisfies the condition

$$\Delta_m(z) = c_0 z^{-n_0}, \qquad (12.114)$$

the synthesis filters obtained by (12.104) will be FIR as well. In this chapter, *we focus on filter banks designed using FIR filters with real coefficients* to avoid potential causality and stability problems inherent in IIR filters.

The first FIR filter bank with perfect reconstruction was introduced independently by Smith and Barnwell (1984) and Mintzer (1985). They defined the analysis filters in terms of a single Mth order FIR prototype filter $H(z)$ as

$$H_0(z) = H(z), \qquad (12.115a)$$
$$H_1(z) = -z^{-M}H(-z^{-1}). \qquad (12.115b)$$

Filters satisfying (12.115) are called *conjugate quadrature filters* (CQFs). If we assume that the order M of the prototype filter $H(z)$ is *odd*, the determinant (12.105) becomes

$$\Delta_m(z) = H_0(z)H_1(-z) - H_0(-z)H_1(z)$$
$$= z^{-M}\left[H(z)H(z^{-1}) + H(-z)H(-z^{-1})\right]. \tag{12.116}$$

To ensure that the synthesis filters are also FIR we require that

$$H(z)H(z^{-1}) + H(-z)H(-z^{-1}) = 1. \tag{12.117}$$

Since $H(z)$ has real coefficients, we have $H(e^{-j\omega}) = H^*(e^{j\omega})$. Thus, evaluating (12.117) on the unit circle yields the equivalent frequency domain condition

$$|H(e^{j\omega})|^2 + |H(e^{j(\omega-\pi)})|^2 = 1. \tag{12.118}$$

Using (12.115) we can also express (12.118) as follows:

$$|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1. \tag{12.119}$$

Two filters $H_0(z)$ and $H_1(z)$ that satisfy condition (12.119) are called *power complementary*. This term is also used for any filters that satisfy (12.117) or (12.118).

Comparing (12.116) with (12.114) we conclude that, when (12.117) holds, $c_0 = 1$ and $n_0 = M$. If we also choose $n_D = M$, the synthesis filters specified by (12.104) and (12.105) are given by

$$G_0(z) = 2H_1(-z) = -2z^{-M}H(z^{-1}), \tag{12.120a}$$
$$G_1(z) = -2H_0(-z) = -2H(-z). \tag{12.120b}$$

To check whether the filter bank specified by (12.115) and (12.120) has the perfect reconstruction property, we note that the product filter is given by

$$R(z) = z^{n_D}H_0(z)G_0(z) = 2H(z)H(z^{-1}). \tag{12.121}$$

Therefore, the perfect reconstruction condition (12.110) becomes

$$R(z) + R(-z) = 2H(z)H(z^{-1}) + 2H(-z)H(-z^{-1}) = 2, \tag{12.122}$$

which is identical to (12.117). Thus, if the filter bank defined by (12.115) and (12.120) satisfies condition (12.117), it has the perfect reconstruction property. Equivalently, we conclude that the product filter $R(z)$ is a half-band filter (see Section 12.3.1). In addition,

we need to know under what conditions a filter $H(z)$ that satisfies (12.117) exists, that is, when the solution of the spectral factorization problem (12.121) exists. To answer this question we recall that the impulse response of the filter defined by (see Section 5.8)

$$R_h(z) = H(z)H(z^{-1}) \tag{12.123}$$

is equal to the autocorrelation sequence of $h[n]$. That is, we have

$$r_h[n] = h[n] * h[-n] \xrightarrow{\mathcal{Z}} R_h(z) = H(z)H(z^{-1}). \tag{12.124}$$

Since $h[n]$ is real, the Fourier transform of (12.124) is given by

$$R_h(e^{j\omega}) = \left|H(e^{j\omega})\right|^2 \geq 0. \tag{12.125}$$

This condition guarantees the solution of the spectral factorization problem (12.121). Thus, a filter $H(z)$ satisfying (12.117) exists if the product filter (12.121) satisfies the condition $R(e^{j\omega}) \geq 0$. Since the autocorrelation sequence (12.124) is equal to zero for even n, except $n = 0$, using (12.82) and (12.124) we obtain

$$\sum_{k=0}^{M} h[k]h[k + 2n] = 0. \quad n \neq 0. \tag{12.126}$$

If we normalize $h[n]$ to have unit energy, that is,

$$\sum_{n=0}^{M} |h[n]|^2 = 1, \tag{12.127}$$

we can express the perfect reconstruction requirement in the time-domain as

$$\sum_{k=0}^{M} h[k]h[k + 2n] = \delta[n], \tag{12.128}$$

which states that the sequence $h[n]$ is orthogonal to its own even translates, with the exception of $n = 0$. This led to the name orthogonal filter banks.

In conclusion, the design of a perfect reconstruction bank reduces to the design of a filter $H(z)$ that satisfies (12.117). However, before we discuss the design of $H(z)$, we determine the impulse response of $H_1(z)$. Note that for $M = 3$ we have

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3},$$
$$H(-z) = h[0] - h[1]z^{-1} + h[2]z^{-2} - h[3]z^{-3},$$
$$H(-z^{-1}) = h[0] - h[1]z + h[2]z^2 - h[3]z^3,$$
$$-z^{-3}H(-z^{-1}) = -h[0]z^{-3} + h[1]z^{-2} - h[2]z^{-1} + h[3]$$
$$= h[3] - h[2]z^{-1} + h[1]z^{-2} - h[0]z^{-3}.$$

Thus, the impulse response of $H_1(z)$ is given by $h_1[n] = (-1)^n h[M - n]$ for $0 \leq n \leq M$. The impulse response and the frequency response of the filters used in a perfect reconstruction bank are given by

$$h_0[n] = h[n] \xleftarrow{\text{DTFT}} H_0(e^{j\omega}) = H(e^{j\omega}), \qquad (12.129\text{a})$$

$$h_1[n] = (-1)^n h[M - n] \xleftarrow{\text{DTFT}} H_0(e^{j\omega}) = -H(e^{-j\omega})e^{-j\omega M}, \qquad (12.129\text{b})$$

$$g_0[n] = h[M - n] \xleftarrow{\text{DTFT}} G_0(e^{j\omega}) = 2H(e^{-j\omega})e^{-j\omega M}, \qquad (12.129\text{c})$$

$$g_1[n] = -(-1)^n h[n] \xleftarrow{\text{DTFT}} G_1(e^{j\omega}) = -2H(e^{-j\omega}). \qquad (12.129\text{d})$$

The design procedure of a perfect reconstruction CQF bank has the following steps:

1. Design a lowpass zero-phase half-band FIR filter $R_0(z)$ of order $2M$, where the number M *must* be an *odd* integer (see Section 12.3.1).
2. If the minimum value $\delta_{\min}$ of the real and even function $R_0(e^{j\omega})$ is negative, form a nonnegative function as

$$R_+(e^{j\omega}) = R_0(e^{j\omega}) + |\delta_{\min}| \geq 0. \qquad (12.130)$$

This is equivalent to adding the value $|\delta_{\min}|$ to the sample $r_0[0]$, that is,

$$r_+[n] = r_0[n] + |\delta_{\min}| \delta[n]. \qquad (12.131)$$

3. Scale $R_+(z)$ so that the frequency response is equal to $1/2$ at $\omega = \pi/2$,

$$R(z) = \frac{1/2}{1/2 + |\delta_{\min}|} R_+(z). \qquad (12.132)$$

4. Determine the minimum-phase filter $H(z)$ by solving the spectral factorization problem $R(z) = H(z)H(z^{-1})$ (see Section 5.8).
5. Specify the remaining filters of the bank using (12.115b) and (12.120).

The steps of this design procedure are illustrated in the following example.

Example 12.8   Perfect reconstruction CQF bank

We shall design a CQF bank using the Parks–McClellan algorithm. For illustrative purposes we choose an FIR prototype $H(z)$ with $M = 7$, which requires the design of half-band filter $R_0(z)$ of order $2M = 14$. The edge frequencies of $R_0(z)$ should be symmetric about $\omega = \pi/2$; thus, we choose $\omega_p = 0.425\pi$ and $\omega_s = 0.575\pi$. The following MATLAB function provides the impulse response and the minimum value of the frequency response function:

```
[r0,deltamin]=firpm(14,[0 0.425 0.575 1],[1 1 0 0],[1 1]).
```
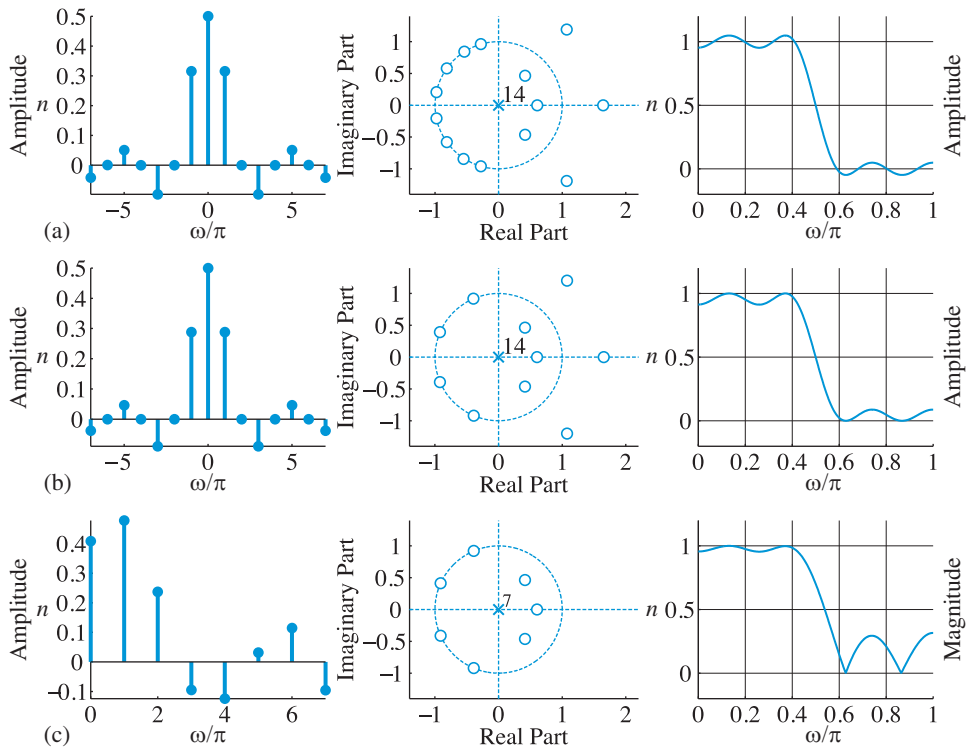
Figure 12.40 The design process for conjugate quadrature filter banks using the Parks–McClellan algorithm.

Figure 12.40(a) shows the impulse response, pole-zero pattern, and amplitude response of $R_0(z)$, which has all the characteristics of a half-band filter. We note that the four pairs of zeros correspond to the four zero crossings of $R_0(e^{j\omega})$.

Since $R_0(e^{j\omega})$ takes negative values, we obtain a valid half-band filter $R(z)$ using (12.130) and (12.132); the characteristics of this filter are shown in Figure 12.40(b). We note that shifting the curve $R(e^{j\omega})$ according to (12.132) has changed the location of the zeros on the unit circle. In theory, this yields double zeros on the unit circle; however, as shown in the plot, this is a numerically sensitive process and the double zeros are not perfectly identical. This deviation leads to reconstruction error, which can be avoided by using a slightly increased value of $\delta_{\min}$ at the expense of reduced stopband attenuation (see Tutorial Problem 12). This process forces all zeros to appear in mirror-image pairs, which simplifies minimum-phase spectral factorization.

The minimum-phase spectral factorization of $R(z)$ is obtained by selecting all zeros inside the unit circle and only one zero from each pair of zeros on the unit circle. The result is a minimum-phase prototype CQF filter $H(z)$ of order $M = 7$. We can also obtain a maximum-phase filter and various mixed-phase filters; however, no linear-phase filter is possible. Figure 12.40(c) shows the impulse response, pole-zero pattern, and magnitude response of $H(z)$. Figure 12.41 shows the magnitude responses of the original half-band filter $R_0(z)$ and the conjugate quadrature filters $H_0(z)$ and $H_1(z)$. We note that $R_0(z)$ and
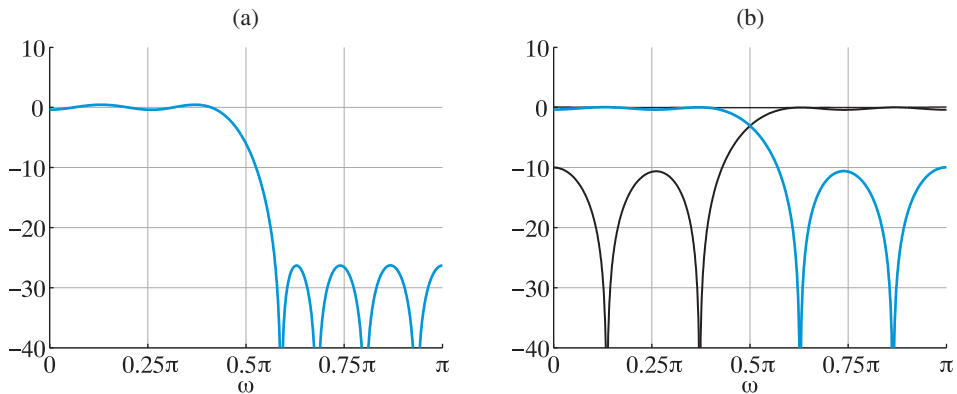
Figure 12.41 Magnitude responses of the half-band filter (a), and the resulting lowpass and highpass conjugate quadrature filters (b).

$H_0(z)$ have the same stopband edge frequency but different stopband attenuations. As a rule of thumb, to obtain a filter $H_0(z)$ with stopband attenuation A_s dB, we should start with a half-band filter having stopband attenuation larger than $(2A_s + 6)$ dB.   ■

Another type of filter for orthogonal filter banks, proposed by Daubechies (1988), is known as Daubechies' family of binomial or maximally flat filters. These filters, which can be used to generate wavelet bases, are required to have a large number of zeros at $\omega = \pi$. The function $R(z)$ has the form

$$R(z) = (1 + z)^m (1 + z^{-1})^m Q(z), \tag{12.133}$$

where $Q(z) = q[0] + \sum_{n=1}^{m-1} q[n](z^n + z^{-n})$ is chosen so that $R(z)$ satisfies (12.110). The minimum-phase spectral factorization of $R(z)$ gives the desired Daubechies' type filter $H_0(z)$, which automatically has m zeros at $z = -1$. This approach leads to lowpass and highpass filters which are maximally flat at $\omega = \pi$ and $\omega = 0$, respectively. This property is important in signal compression applications. Although the subject of wavelets is beyond the scope of this book, we discuss some simple cases in the problems.

12.4.4   FIR quadrature mirror filter (QMF) banks

Quadrature mirror filter (QMF) banks provide complete cancellation of the output aliasing error, but the perfect reconstruction property (12.102b) is only approximately satisfied. The basic idea, introduced by Croisier *et al.* (1976), is to design a lowpass filter $H(z)$ and then determine the filters in the bank as follows:

$$H_0(z) = H(z), \quad H_1(z) = H(-z), \tag{12.134a}$$

$$G_0(z) = H(z), \quad G_1(z) = -H(-z). \tag{12.134b}$$

The time and frequency domain characteristics of these filters are given by

$$h_0[n] = h[n] \xleftarrow{\text{DTFT}} H_0(e^{j\omega}) = H(e^{j\omega}), \tag{12.135a}$$

$$h_1[n] = (-1)^n h[n] \xleftarrow{\text{DTFT}} H_1(e^{j\omega}) = H(e^{j(\omega-\pi)}), \tag{12.135b}$$

$$g_0[n] = h[n] \xleftarrow{\text{DTFT}} G_0(e^{j\omega}) = H(e^{j\omega}), \tag{12.135c}$$

$$g_1[n] = (-1)^{n+1} h[n] \xleftarrow{\text{DTFT}} H_1(e^{j\omega}) = -H(e^{j(\omega-\pi)}), \tag{12.135d}$$

where clearly $H(e^{j(\omega-\pi)})$ is a highpass filter. A simple change of variables yields

$$\left| H_1(e^{j(\pi/2-\omega)}) \right| = \left| H_0(e^{j(\pi/2+\omega)}) \right|, \tag{12.136}$$

that is, the magnitude responses are symmetric about the "quadrature" frequency $\omega = 2\pi/4$ (see Figure 12.43). This symmetry led to the term *quadrature mirror filter* banks.

Using the following polyphase decomposition of the lowpass prototype:

$$H(z) = P_0(z^2) + z^{-1} P_1(z^2), \tag{12.137}$$

we can obtain the efficient implementation of the QMF bank shown in Figure 12.42.

To determine the properties of QMF banks, we compute the quantities $A(z)$, $T(z)$, and $\Delta_m(z)$. The QMF bank is aliasing-free because

$$A(z) = H_0(-z)G_0(z) + H_1(-z)G_1(z)$$
$$= H(-z)H(z) - H(z)H(-z) = 0. \tag{12.138}$$

The system function of the alias-free QMF bank is given by

$$T(z) = H_0(z)G_0(z) + H_1(z)G_1(z)$$
$$= H^2(z) - H^2(-z). \tag{12.139}$$

Finally, the determinant (12.105) is given by

$$\Delta_m(z) = H_0(z)H_1(-z) - H_0(-z)H_1(z)$$
$$= H^2(z) - H^2(-z). \tag{12.140}$$
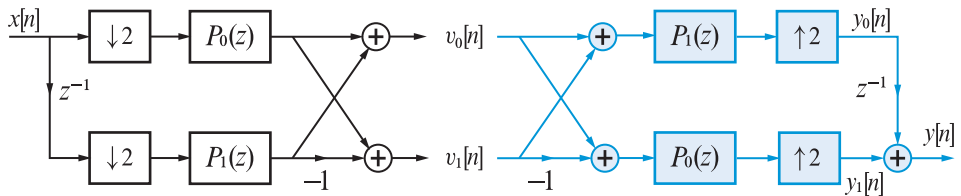
Figure 12.42 Polyphase implementation of the two-channel QMF filter bank.

From (12.137), (12.139), and (12.140) we obtain

$$T(z) = \Delta_m(z) = 2z^{-1}P_0(z^2)P_1(z^2). \tag{12.141}$$

If the product $P_0(z^2)P_1(z^2)$ is a pure delay, conditions (12.138) and (12.141) ensure that perfect reconstruction with an FIR filter bank is possible. To satisfy this requirement, we choose $P_0(z) = b_0 z^{-n_0}$ and $P_1(z) = b_1 z^{-n_1}$. Then the filter $H(z)$ has the form

$$H(z) = b_0 z^{-2n_0} + b_1 z^{-(2n_1+1)}, \tag{12.142}$$

which is of limited practical usefulness. In conclusion, there are *no* practical alias-free FIR QMF banks with perfect reconstruction.

Elimination of phase distortion If we restrict $H(z)$ to be an FIR filter with linear phase, then $T(z)$ given by (12.141) also has linear phase. Thus, we can create an alias-free QMF bank without phase distortion. A lowpass FIR filter with linear phase should have a symmetric impulse response (Type I or II) and its frequency response can be expressed as (see Section 10.2)

$$H(e^{j\omega}) = A(e^{j\omega})e^{-j\omega M/2}. \tag{12.143}$$

Substituting (12.143) into (12.139) and recalling that $|H(e^{j\omega})|$ has even symmetry, we obtain

$$T(e^{j\omega}) = e^{-j\omega M}\left[|H(e^{j\omega})|^2 - (-1)^M|H(e^{j(\omega-\pi)})|^2\right]. \tag{12.144}$$

If M is even, then $T(e^{j\pi/2}) = 0$, which causes severe amplitude distortion in the signal transmission path. Therefore, we must choose M to be an *odd* number (Type II FIR filter), which leads to the following relation

$$T(e^{j\omega}) = e^{-j\omega M}\left[|H(e^{j\omega})|^2 + |H(e^{j(\omega-\pi)})|^2\right], \tag{12.145a}$$

$$= e^{-j\omega M}\left[|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2\right]. \tag{12.145b}$$

Perfect reconstruction requires $H_0(z)$ and $H_1(z)$ to be power complementary, that is,

$$|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1. \tag{12.146}$$

However, the only FIR filters satisfying (12.146) are the trivial filters in (12.142). Thus, it is not possible to design practical FIR QMF banks with perfect reconstruction. Therefore, the best we can do is to design the prototype filter $H(z)$ so that condition (12.146) is approximated as closely as possible. Johnston (1980) proposed a design technique based on minimization of the following criterion:

$$J = \alpha \int_{\omega_s}^{\pi} |H(e^{j\omega})|^2 d\omega + (1-\alpha) \int_0^{\pi} \left(1 - |T(e^{j\omega})|^2\right) d\omega, \tag{12.147}$$
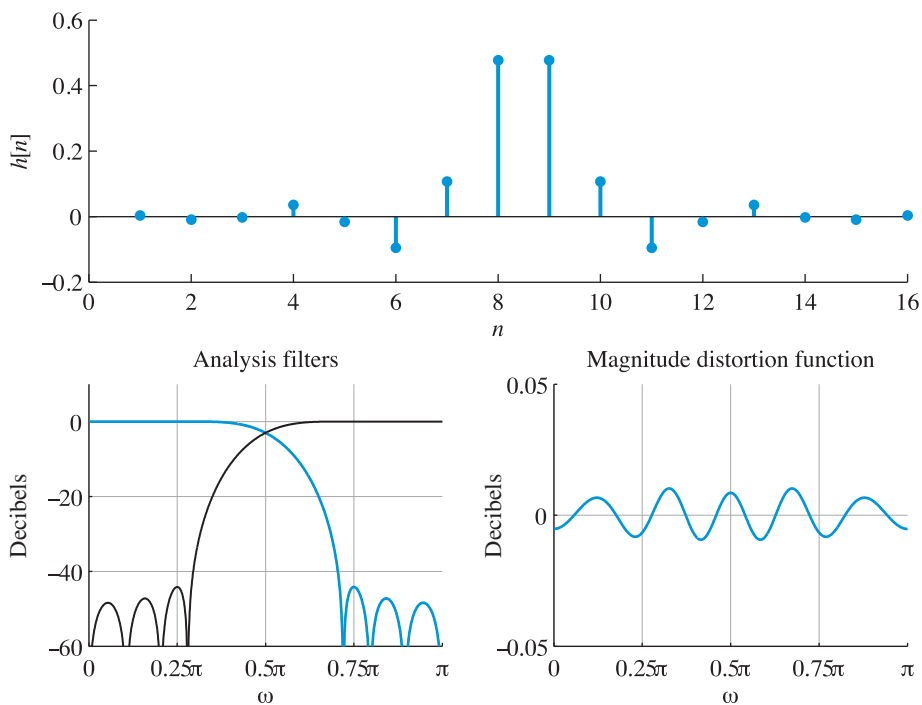
Figure 12.43 Impulse response, magnitude responses, and magnitude distortion function for a QMF bank designed using the linear-phase Johnston16B filter.

where α is a weighting factor in the range $0 < \alpha < 1$. This design technique tries to approximate the perfect reconstruction conditions with the second term of the cost function, while minimizing the stopband energy or aliasing error with the first term. Tables of optimum filter coefficients designed with this objective in mind are provided by Johnston (1980) and Crochiere and Rabiner (1983). Figure 12.43 shows the impulse response, magnitude responses of the analysis filters, and the magnitude distortion function for a QMF bank using a 16B linear-phase lowpass FIR filter designed by Johnston (1980). More details can be found in Tutorial Problem 15.

12.5   Multichannel filter banks

The matrix approach introduced in Section 12.4 can be extended and used to obtain closed form solutions for designing K-channel filter banks without aliasing and with perfect reconstruction. However, the design process is complicated because we have to find K different filters that satisfy the conditions for perfect reconstruction. In this section we discuss two simple approaches to bypass this problem. In the first approach, all required filters are derived from a single lowpass prototype using modulation. In the second approach we

use a two-channel filter bank in a tree structure. More detailed treatments of multichannel filter banks are provided in Fliege (1994) and Strang and Nguyen (1996).

12.5.1   Modulated filter banks

The basic idea is to start with a lowpass prototype filter and create all other filters by frequency translation. Consider a prototype causal lowpass filter $H(z)$ with real impulse response $h[n]$ and approximate bandwidth π/K. The analysis filters are defined by

$$h_k[n] = h[n]e^{j(2\pi/K)kn} = h[n]W_K^{-kn}, \quad k = 0, 1, \ldots, K-1 \tag{12.148}$$

where $W_K \triangleq e^{-j\frac{2\pi}{K}}$ is the twiddle factor. The z-transform of (12.148) is given by

$$H_k(z) = \sum_{n=0}^{\infty} h_k[n]z^{-n} = \sum_{n=0}^{\infty} h[n]\left(zW_K^k\right)^{-n} = H\left(zW_K^k\right). \tag{12.149}$$

Evaluating the z-transform for $z = e^{j\omega}$ yields the frequency response functions

$$H_k\left(e^{j\omega}\right) = H\left(e^{j(\omega - 2\pi k/K)}\right). \quad k = 0, 1, \ldots, K-1 \tag{12.150}$$

Thus, the frequency response of the kth filter $h_k[n]$ is obtained by shifting the frequency response of the lowpass prototype to the right by $2\pi k/K$ rads. The resulting arrangement of spectral channels is shown in Figure 12.37. The same filters can be used in the synthesis filter bank.

The filter bank obtained is computationally inefficient. However, we can obtain a more efficient implementation using the polyphase decomposition of the prototype filter, given by (see Section 12.2.3)

$$H(z) = \sum_{m=0}^{K-1} z^{-m} P_m\left(z^K\right), \tag{12.151}$$

where

$$P_m(z) = \sum_{n=0}^{\infty} p_k[nK + m]z^{-n}. \tag{12.152}$$

Replacing z by zW_K^k in (12.151) and using the identity $W_K^{kK} = 1$ we obtain the K-band polyphase decomposition of $H_k(z)$. The result is

$$H_k(z) = \sum_{m=0}^{K-1} z^{-m} W_K^{-km} P_m(z^K W_K^{kK}) = \sum_{m=0}^{K-1} z^{-m} W_K^{-km} P_m(z^K). \tag{12.153}$$

This is a set of K linear equations which can be written in matrix form as follows:

$$
\begin{bmatrix}
H_0(z) \\
H_1(z) \\
\vdots \\
H_{K-1}(z)
\end{bmatrix}
=
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & W_K^1 & W_K^2 & \cdots & W_K^{(K-1)} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & W_K^{(K-1)} & W_K^{2(K-1)} & \cdots & W_K^{(K-1)^2}
\end{bmatrix}^*
\begin{bmatrix}
P_0(z^K) \\
z^{-1}P_1(z^K) \\
\vdots \\
z^{-(K-1)}P_{K-1}(z^K)
\end{bmatrix},
$$

(12.154)

where * denotes complex conjugate. The matrix in (12.154) is equal to the conjugate of the K-point DFT matrix, that is, $W_K^* = NW_K^{-1}$ (see Section 7.2). Thus, the matrix by vector multiplication in (12.154) computes the inverse DFT of the polyphase vector multiplied by N. The filter bank obtained is known as a *uniform DFT filter bank*. Using this interpretation, we can see that the analysis filter bank in (12.154) is equivalent to the polyphase-based filter bank shown in Figure 12.44. Since the inverse DFT is a fixed linear operation, we can move the downsamplers before the inverse DFT. Furthermore, the downsamplers can be moved before the filters by using the multirate identity for downsampling (see Figure 12.18). The result is an efficient polyphase implementation of the uniform DFT analysis filter bank shown in Figure 12.45(a). Using a similar approach we can obtain an efficient polyphase implementation for the synthesis bank (see Problem 16), which is shown in Figure 12.45(b). The computational complexity of a polyphase bank using FIR filters with N coefficients is $(N/K) + 2\log_2 K + 1$ real multipliers per input sample, compared to $2KN$ real multipliers for the original bank. See Tutorial Problem 17 for an example of a four-channel DFT bank.

Uniform DFT filter banks (with or without critical sampling) imitate the short-time Fourier transform (see Section 7.6.5). It is not possible to design FIR uniform DFT banks with perfect reconstruction. Perfect reconstruction with IIR filters is possible, however, either the analysis or synthesis filters are noncausal, and performance may be poor. A better approach is to keep the aliasing and filtering errors arbitrarily small resulting in filter banks with "*near perfect reconstruction.*" A widely used approach is to design consecutive pairs of filters that are approximately power complementary between their center frequencies. Thus, we need to compensate only for aliasing caused by adjacent channels. All other alias spectra are suppressed by designing a prototype filter with sufficiently
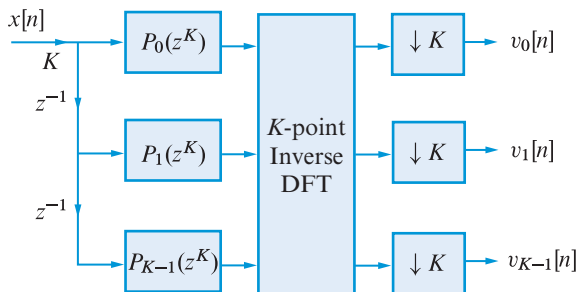
Figure 12.44 First step towards a polyphase uniform DFT analysis filter bank.

(a) Analysis   (b) Synthesis
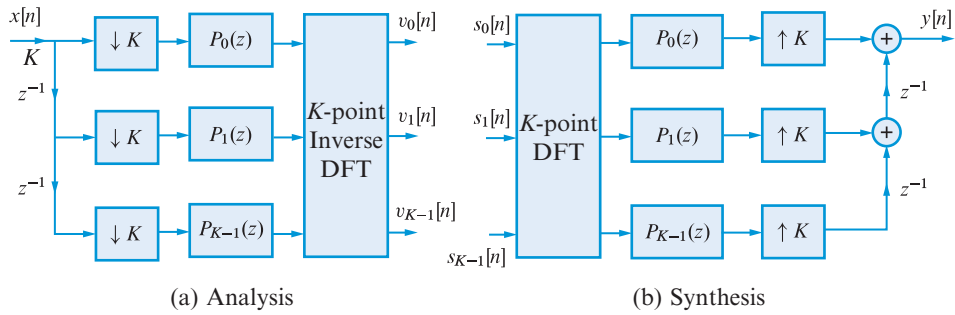
Figure 12.45 (a, b) Polyphase implementation of a uniform DFT filter bank.

high stopband attenuation. Such filter banks are known as *pseudo-QMF* banks. If the modulation by complex exponentials is replaced by cosine modulation, we obtain *cosine-modulated filter banks* (see Tutorial Problem 18 for an example). Cosine-modulated QMF filter banks are used for audio coding in the MPEG compression standards; see Bosi and Goldberg (2003). More details about pseudo-QMF banks can be found in Fliege (1994) and Vaidyanathan (1993).

12.5.2   Tree-structured filter banks

An alternative way to split a signal into multiple sub-bands is by using tree-structured filter banks. A simple way to design the required filters is by cascading two-channel filter banks. The most widely used structures are (a) the uniform-band tree structure illustrated in Figure 12.46, and (b) the octave-band tree structure illustrated in Figure 12.47. If the two-band filter banks used as building blocks have perfect reconstruction, the resulting tree-structured band provides perfect reconstruction as well.

The uniform-band tree structure is essentially a different implementation of a uniform filter bank with $K = 2^\kappa$ channels. However, the octave-band tree structure provides a nonuniform (logarithmic) decomposition of the spectrum, which is useful in many coding
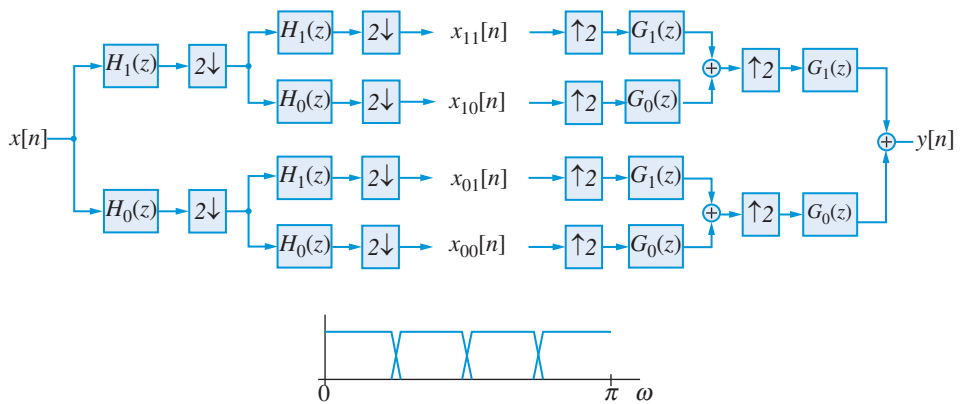
Figure 12.46 A two-level uniform tree-structured filter bank.
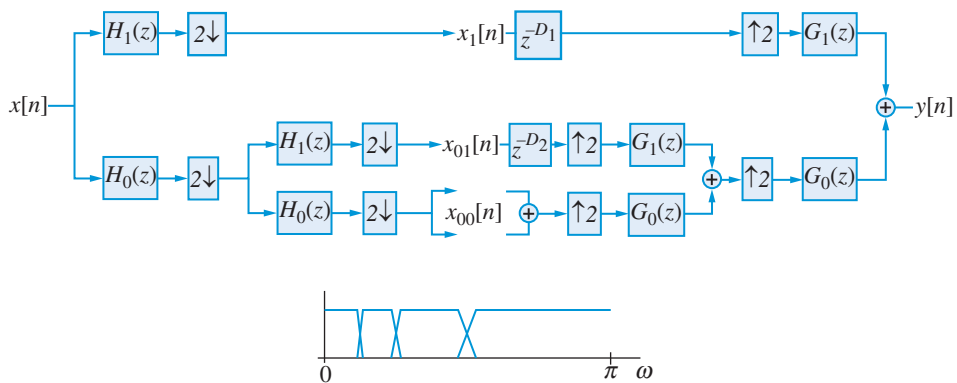
Figure 12.47 Two-level octave-band tree structured filter bank.

applications. In this case, the original signal is split into two equal frequency bands by a lowpass half-band filter $H_0(z)$ and a highpass half-band filter $H_1(z)$. The upper channel produces a low-frequency ("smooth") signal and the lower channel produces a high-frequency ("detail") signal. In coding applications, the detail signal is downsampled by a factor of two and then quantized using a small number of bits. This "level-1" smooth signal is downsampled by a factor of two and the resulting signal is decomposed to "level-2" smooth and detail components using the same filter bank. The level-2 detail signal can be quantized using an appropriate number of bits. This process can be continued to as many levels as are required by a particular application. The synthesis filter bank follows the inverse process. The delays D_1 and D_2 ensure the availability of all outputs of the analysis bank to the input of the synthesis bank at the same time. The lowest frequency band provides a smooth signal with "few details." Since higher bands are shifted-up in the frequency axis and have larger bandwidths, they provide signals with "more details" or "more resolution." In this sense, we say that the octave-tree bank provides a multiresolution decomposition of the original signal. Such multiresolution decompositions are closely related to the *discrete wavelet transform*. Basically, the discrete-wavelet transform is an octave-tree filter bank with analysis and synthesis filters that satisfy some special properties; see Strang and Nguyen (1996) and Fliege (1994) for more details.

Learning summary

- Let $x[n] = x_c(nT_x)$ and $y[n] = x_c(nT_y)$ be two discrete-time signals obtained from the same continuous-time signal without aliasing. The computation of $y[n]$ from $x[n]$ or vice versa is known as sampling rate conversion.

- Decreasing the sampling rate by an integer factor D (downsampling) may cause aliasing. To avoid aliasing, we precede the downsampler by a lowpass filter with cutoff frequency $\omega_c = \pi/D$ (decimation system).

- Increasing the sampling rate by an integer factor I (upsampling) always creates spectral images. To remove these images we put after the upsampler a lowpass filter with cutoff frequency $\omega_c = \pi/I$ (interpolation system).

- Combining interpolation and decimation, it is possible to change the sampling rate by a rational factor I/D. Conversion by arbitrary factors requires more complicated systems.

- Efficient implementation of sampling rate converters is made possible with the use of polyphase structures, which essentially move the filtering operation on the side of the system with the lower sampling rate.

- Filter banks allow the decomposition and processing of signals in sub-bands to achieve (a) more efficient representations for storage or transmission applications, or (b) more efficient implementation of filters and transforms.

- Two channel filter banks with perfect reconstruction provide a powerful tool for implementation of multichannel filter banks or wavelet transforms.

TERMS AND CONCEPTS

Analysis filter bank A filter bank with a common input that splits a signal into a number of subband signals.

Bi-orthogonal filter bank A two-channel filter bank in which the product filter is the z-transform of a cross-correlation sequence.

Decimation A sampling rate reduction (by a factor D) system in which a lowpass filter with cutoff frequency π/D is followed by a downsampler.

Downsampler A system that chooses one out of every D samples of a discrete-time signal. Also called a sample rate compressor.

Discrete-time sampling rate Denoted by f_s which means keep one out of every D samples and is given by $f_s = 1/D$.

Filter bank A collection of filters with a common input or a common output.

Fractional delay A noninteger delay between samples of two signals. It is obtained using a combination of interpolation, integer delay, and decimation operations.

Half-band filter An ideal lowpass filter with cutoff frequency $\pi/2$ needed in the decimation-by-2 system. More generally, any zero-phase filter that has every even-ordered sample zero except the one at $n = 0$ which is one-half.

Ideal discrete-time interpolator A non-realizable system that reconstructs samples between two samples of a bandlimited signal using sinc interpolating functions.

Interpolation A sampling rate increment (by a factor I) system in which an upsampler is followed by a lowpass filter with cutoff frequency π/I.

Interpolated FIR (IFIR) filter An efficient implementation of a narrowband lowpass FIR filter that uses a wider transition band lowpass followed by an interpolated prototype filter of wider transition band,

thereby reducing the overall number of filter coefficients.

K-band filter An ideal lowpass filter with cutoff frequency π/K needed in the decimation-by-K system. More generally, any zero-phase filter that has every Kth-ordered sample zero except the one at $n = 0$ which is $1/K$. Also known as a Nyquist filter.

Linear interpolation An interpolator that reconstructs samples between two samples of a bandlimited signal using a straight-line approximation.

Maximally flat multirate filter bank A combination of analysis and synthesis uniform filter banks that uses decimation after analysis and interpolation before synthesis.

Modulated filter bank A uniform filter bank which is created using a basic lowpass filter and all its frequency translates.

Multirate identities A set of identities that enable the understanding and simplification of multirate system operations. Also known as Noble identities.

Multirate system A discrete-time system with different sampling rates at various parts of the system.

Nyquist filter An ideal lowpass filter with cutoff frequency π/K needed in the decimation-by-K system. More generally, any zero-phase filter that has every Kth-ordered sample zero except the one at $n = 0$ which is $1/K$. Also known as a K-band filter.

Octave-band filter bank A filter bank which splits (combines) signal over (from) power-of-two frequency bands.

Orthogonal (para-unitary) filter bank A two-channel filter bank in which the product filter is the z-transform of an autocorrelation sequence.

Perfect reconstruction An implementation of maximally flat multirate filter banks that obtains an exact reconstruction of the input signal.

Polyphase representation Decomposition of a finite $N = ML$ length filter impulse response $h[n]$ into M subfilters each of length L where their impulse response is obtained by downsampling the shifted $h[nM + k]$.

Polyphase structures Discrete-time system structures created for decimators and interpolators using polyphase decomposition of their impulse response. These are efficient structures.

Product filter A filter that is a product of the basic analysis and synthesis filters.

Quadrature mirror bank (QMF) filter bank A two-channel filter bank that provides a complete cancellation of the output aliasing error but satisfies the perfect reconstruction property only approximately.

Resampling A process that changes sampling rate of the equivalent analog signal from its samples using decimation, interpolation, or a combination of the two. Also known as sampling rate conversion.

Sampling rate change (conversion) A process that changes sampling rate of the equivalent analog signal from its samples using decimation, interpolation, or a combination of the two. Also known as resampling.

Sampling rate compressor A system that chooses one out of every D samples of a discrete-time signal. Also called a downsampler.

Sampling rate expander A system that inserts $I - 1$ zeros between every sample of a discrete-time signal. Also called an upsampler.

Sub-band signals Decomposition of a signal into a number of signals, each with mutually exclusive but exhaustive bands in the frequency-domain.

Synthesis filter bank A filter bank with a common output that combines sub-band signals into one fullband signal.

Tree structured filter bank A nonuniform filter bank created using a two channel uniform filter bank in a tree structure.

Uniform DFT filter bank An efficient implementation of the uniform modulated filter bank using DFT.

Uniform filter bank A filter bank which splits (combines) signal over (from) uniform frequency bands.

Upsampler A system that inserts $I - 1$ zeros between every sample of a discrete-time signal. Also called a sampling rate expander.

MATLAB functions and scripts

Name	Description	Page
decimate	Resamples data at a lower rate after lowpass filtering	713
downsample	Retains every Nth sample starting with the first	707
firdec*	FIR decimation by an integer factor	715
interp	Resamples data at a higher rate using lowpass interpolation	719
ppdecim*	Polyphase implementation of the decimation filter	734
ppinterp*	Polyphase implementation of the interpolation filter	735
resample	Changes the sampling rate of a signal	735
reshape	Changes the shape of an array	731
src*	Sample rate compressor	707
sre*	Sample rate expander	717
upfirdn	Resamples a signal using FIR filter	713
upsample	Inserts $(N-1)$ zeros between input samples	717

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A more detailed treatment of multirate signal processing, at the same level as in this book, is given in Porat (1997), Mitra (2006), and Proakis and Manolakis (2007).
2. A lucid introduction to decimation and interpolation from a signal processing viewpoint is given in a classic paper by Schafer and Rabiner (1973). The paper by Crochiere and Rabiner (1981) and the book by Crochiere and Rabiner (1983) provide an extensive coverage of multirate signal processing up to the date of publication.
3. The interconnected areas of multirate signal processing, filter banks, and wavelets from a signal processing perspective are covered by Fliege (1994), Vaidyanathan (1993), Akansu and Haddad (2001), Vetterli and Herley (1992), Strang and Nguyen (1996), and Burrus *et al.* (1998).

Review questions

1. What is a multirate system and why do we need such a system?
2. What are the basic components of a multirate system?
3. Describe conceptual operations needed in sampling rate conversion.
4. What is a downsampler and how many different output sequences are possible for downsampling by D?
5. Is downsampling a linear operation? Is it time-invariant? Explain.
6. Are there any differences between the sampling of continuous-time signals and the sampling of discrete-time signals in terms of spectra? Explain.
7. Describe the graphical procedure to obtain the spectrum of a decimated signal from that of the given discrete-time signal.
8. What is an ideal decimator and why is it different from a downsampler?

9. What is an ideal antialiasing filter and what should be its bandwidth for decimation by a factor of D?

10. The continuous-time ideal bandlimited interpolation and the discrete-time ideal interpolation given a decimated signal are related by a simple relation. What is this relation?

11. What is an upsampler and how many different output sequences are possible for upsampling by I?

12. Is upsampling a linear operation? Is it time-invariant? Explain.

13. Describe the effect of upsampling on the spectrum of a discrete-time signal.

14. What is the difference between upsampling and interpolation?

15. What is an ideal anti-imaging filter and what should be its bandwidth for interpolation by a factor of I.

16. Can linear interpolation be applied in real-time? If not, how should it be implemented?

17. How is sampling rate conversion by a rational factor implemented in practice? Explain using a block diagram.

18. Explain multirate identities and their usefulness.

19. What is a polyphase signal decomposition. Explain using $N = 9$ and $M = 3$.

20. Describe the polyphase decimation structure using $D = 4$ and explain why it is efficient.

21. Describe the polyphase interpolation structure using $D = 3$ and explain why it is efficient.

22. What is a half-band filter? Describe its properties.

23. What is a K-band filter? Describe its properties.

24. Multistage decimation or interpolation is more efficient than single-stage decimation or interpolation. Do you agree or disagree? Clearly explain why.

25. How is IFIR different from an ordinary FIR filter and in what way is it efficient?

26. What is a filter bank and why is decimation/interpolation used in its implementation.

27. What are analysis filters and what are their functions?

28. What are synthesis filters and what are their functions?

29. Explain the different types of filter bank.

30. What are the conditions for perfect reconstruction in a two-channel filter bank?

31. Explain orthogonal (para-unitary) and bi-orthogonal two-channel filter banks.

32. What are conjugate quadrature and power complementary filters and in which application do we need them?

33. What are QMF filter banks and what purpose do they serve?

34. Explain the difference between a modulated filter bank and a tree-structured filter bank.

35. Explain the difference between a uniform-band and an octave-band filter bank.

Problems

Tutorial problems

1. Generate 100 samples of the signal $x[n] = \sin(0.125\pi n)$. We want to decimate this signal using $D = 2$.
   (a) Design a 25-order lowpass filter in Figure 12.5 using the Parks–McClellan algorithm. Choose the appropriate band-edge frequencies and equal ripple values to obtain $h[n]$.
   (b) Implement the decimator of Figure 12.5 to obtain $x_D[n]$. Use the `src` function. Plot steady-state values of $x_D[n]$.
   (c) Implement the decimator using the `firdec` function to obtain $x_D[n]$ and plot steady-state values of $x_D[n]$.
   (d) Implement the decimator using the `upfirdn` function to obtain $x_D[n]$ and plot steady-state values of $x_D[n]$.
   (e) Compare your results in parts (b), (c), and (d) and comment.

2. Consider implementations of the decimator given in the block diagram of Figure 12.5 and the MATLAB function `firdec`.
   (a) Develop and draw a direct form structure of the decimator in Figure 12.5 assuming an FIR filter.
   (b) Develop and draw a direct form structure of the decimator implemented in `firdec`. Compare the two structures.

3. Generate 100 samples of the signal $x[n] = \sin(\pi n)$. We want to interpolate this signal using $I = 4$.
   (a) Design a 45-order lowpass filter in Figure 12.9 using the Parks–McClellan algorithm. Choose the appropriate band-edge frequencies and equal ripple values to obtain $h[n]$.
   (b) Implement the interpolator of Figure 12.9 to obtain $x_I[n]$. Use the `sre` function. Plot steady-state values of $x_I[n]$.
   (c) Implement the interpolator using the `upfirdn` function to obtain $x_I[n]$ and plot steady-state values of $x_I[n]$.
   (d) Implement the interpolator using the `interp` function to obtain $x_I[n]$ and plot steady-state values of $x_I[n]$.
   (e) Compare your results in parts (b), (c), and (d) and comment.

4. Let $x[n] = \cos(0.9\pi n)$, $n = 0, \ldots, 50$.
   (a) Using the `interp` function, interpolate using $I = 2$, $I = 4$, and $I = 8$. Stem plot the original and interpolated signals.
   (b) Using the second output argument of the `interp` function, plot the frequency response of the lowpass filter used in each of the above interpolations.

5. Generate a 101-length sequence $x[n]$ using the `fir2` function with normalized-frequency array `[0,0.1,0.2,0.5,0.55,0.6,1]` and the corresponding magnitude array `[2,2,1.5,1,0.5,0,0]`. The frequency normalization is with respect to π radians.
   (a) Compute and plot the magnitude spectra of $x[n]$.
   (b) Upsample $x[n]$ using $I = 2$ and plot the spectrum of the decimated signal.
   (c) Upsample $x[n]$ using $I = 3$ and plot the spectrum of the decimated signal.

(d) Upsample $x[n]$ using $I = 4$ and plot the spectrum of the decimated signal.

(e) Comment on your spectra.

6. Using Figure 12.13 as a guide, express the linear interpolation formula (12.47) as a convolution summation given in (12.44).

7. Figure 12.14 shows frequency responses of the ideal and linear interpolators. In this problem we explore zero-order-hold (ZOH) and linear (FOH))interpolators. The impulse response of the ZOH interpolator for a factor I is given by $g_{ZOH}[n] = u[n] - n[n - I]$ while that of the FOH interpolator is given in (12.48).

(a) On the same graph plot magnitude responses of the ideal, ZOH, and FOH interpolators for $I = 3$.

(b) Repeat (a) for $I = 5$.

8. Let $x[n] = 2\cos(0.1\pi n) + \sin(0.2\pi n) + 0.5\cos(0.4\pi n)$. Use the `resample` function with default parameters.

(a) Resample the sequence $x[n]$ at $4/5$ times the original rate to obtain $x_{I_1}[m]$ and provide the `stem` plots of both sequences.

(b) Resample the sequence $x[n]$ at $5/4$ times the original rate to obtain $x_{I_2}[m]$ and provide the `stem` plots of both sequences.

(c) Resample the sequence $x[n]$ at $2/3$ times the original rate to obtain $x_{I_3}[m]$ and provide the `stem` plots of both sequences.

(d) Explain which of the three output sequences retain the "shape" of the original sequence $x[n]$.

9. Consider the sequence given in Figure 12.15, $x[n] = \cos(0.04\pi n) + 3\sin(0.0072\pi n)$, $0 \le n \le 80$.

(a) Obtain the plots shown in Figure 12.15 for $D = I = 5$.

(b) Repeat for $D = I = 3$ and $D = I = 10$. Explain the ramp like behavior in the beginning. Explain why the smaller the values of $D = I$ the better response we obtain.

10. Following the steps used in obtaining (12.84) and (12.85) for the half-band filter, show that the K-band filter satisfies (12.90) and (12.91).

11. Following the steps used in obtaining the two-stage decimation system of Figure 12.30, obtain the necessary set of equations, using the multirate identity for interpolation, that justifies the implementation given in Figure 12.32.

12. Consider the design of the CQF bank in Example 12.8. It was noted that the design steps, and specifically (12.130), lead to double zeros on the unit circle with increased sensitivity that can lead to reconstruction error. Modify (12.130) so that

$$R_+(e^{j\omega}) = R_0(e^{j\omega}) + 2|\delta_{min}| \ge 0.$$

Obtain the new design with plots similar to those in Figure 12.40 and verify that there are no double zeros on the unit circle.

13. A signal $x[n]$ is bandlimited to $2\pi/3$ radians. We want to change its bandwidth to π radians using a fractional sampling rate converter. This can be achieved by upsampling, followed by lowpass filtering, and finally downsampling the signal.

(a) Determine the smallest values of I and D.

(b) For the obtained value of I above, sketch the spectrum of the upsampled signal.

(c) Sketch a plot of the magnitude response of the required lowpass filter.

(d) Finally, using the obtained minimum value of D, sketch the spectrum of the downsampled signal and verify its bandwidth.

14. Consider the polyphase interpolation structure given in Figure 12.25. Show that the transpose direct form II structure, Figure 12.25(a), has the same complexity as in Figure 12.25(b) but at a higher rate.

15. Using Johnston's 16B QMF bank set of coefficients, obtain and plot the impulse response, magnitude response, and the magnitude distortion function.

16. Following steps similar to those used in the development of the analysis filter bank in Figure 12.45(a), develop the synthesis filter bank equation and verify the structure in Figure 12.45(b).

17. Using equations (12.151) through (12.154) develop a four-channel filter bank and draw its efficient structure for both analysis and synthesis filters.

18. Consider the cosine modulated filter bank, which is termed a Pseudo QMF bank.
    (a) Develop the structure of the filter bank.
    (b) Explain the compensation of the alias components.
    (c) Obtain the design procedure using a raised-cosine filter.

Basic problems

19. Consider the signal $x[n] = 0.9^n u[n]$. It is to be downsampled by a factor of $D = 3$ to obtain $x_D[n]$.
    (a) Compute the spectrum of $x[n]$ and plot its magnitude.
    (b) Compute the spectrum of $x_D[n]$ and plot its magnitude.
    (c) Compare the two spectra.

20. Determine the output $y[n]$ in terms of the input $x[n]$ for the following system

$$x[n] \rightarrow \boxed{\uparrow 5} \rightarrow \boxed{\downarrow 15} \rightarrow \boxed{\uparrow 5} \rightarrow y[n].$$
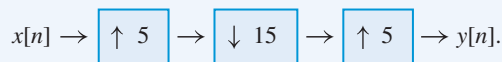
21. Using the `downsample` function, resample the following sequences using the given parameters D and the offset k. Using the `stem` function, plot the original and downsampled signals.
    (a) $x[n] = \sin(0.2\pi n)$, $0 \leq n \leq 50$, $D = 4$, $k = 0$, and $k = 2$.
    (b) $x[n] = \cos(0.3\pi n)$, $0 \leq n \leq 60$, $D = 3$, $k = 0$, and $k = 1$.
    (c) $x[n] = 0.2n$, $0 \leq n \leq 100$, $D = 5$, $k = 0$, and $k = 3$.
    (d) $x[n] = \sin(0.25\pi n)$, $0 \leq n \leq 32$, $D = 4$, $k = 0$, and $k = 2$.
    (e) $x[n] = 1 - \cos(0.6\pi n)$, $0 \leq n \leq 100$, $D = 2$, $k = 1$.

22. Using the `decimate` function, resample the following sequences using the given parameters D. Using the `stem` function, plot the original and decimated signals. Obtain results using both the default IIR and FIR decimation filters and comment on the results.
    (a) $x[n] = \cos(0.4\pi n)$, $0 \leq n \leq 100$, $D = 2$.
    (b) $x[n] = \sin(0.15\pi n)$, $0 \leq n \leq 100$, $D = 3$.
    (c) $x[n] = \cos(0.05\pi n) + 2\sin 0.001\pi n$, $0 \leq n \leq 150$, $D = 5$.
    (d) $x[n] = \cos(0.25\pi n)$, $0 \leq n \leq 100$, $D = 4$.
    (e) $x[n] = 1 - \sin(0.01\pi n)$, $0 \leq n \leq 100$, $D = 5$.

23. Generate 100 samples of the signal $x[n] = \sin(0.1\pi n)$. We want to decimate this signal using $D = 5$.

   (a) Design a lowpass filter in Figure 12.5 using the Parks–McClellan algorithm. Choose appropriate band-edge frequencies and ripple parameters of $A_p = 1$ dB and $A_s = 50$ dB to obtain $h[n]$.

   (b) Implement the decimator of Figure 12.5 to obtain $x_D[n]$. Plot steady-state values of $x_D[n]$.

   (c) Implement the decimator using the `firdec` function to obtain $x_D[n]$ and plot steady-state values of $x_D[n]$.

   (d) Implement the decimator using the `upfirdn` function to obtain $x_D[n]$ and plot steady-state values of $x_D[n]$.

   (e) Implement the decimator using the `decimate` function to obtain $x_D[n]$ and plot steady-state values of $x_D[n]$.

   (f) Compare your results in parts (b) through (e) and comment.

24. Generate a 151-length sequence $x[n]$ using the `fir2` function with normalized-frequency array `[0,0.05,0.1,0.11,1]` and the corresponding magnitude array `[1,1,1,0,0]`. The frequency normalization is with respect to π radians. Use the default FIR filter in the following decimations:

   (a) Compute and plot the magnitude spectra of $x[n]$.

   (b) Decimate $x[n]$ using $D = 2$ and plot the spectrum of the decimated signal.

   (c) Decimate $x[n]$ using $D = 3$ and plot the spectrum of the decimated signal.

   (d) Decimate $x[n]$ using $D = 4$ and plot the spectrum of the decimated signal.

   (e) Comment on your spectra.

25. Using the `upsample` function perform the $I = 4$ upsampling on the following sequences. Stem plot the original and upsampled signals and use various values of the offset parameters.

   (a) $x[n] = 2\sin(0.8\pi n)$, $n = 0, \ldots, 20$.

   (b) $x[n] = 4\cos(0.0.85\pi n)$, $n = 0, \ldots, 30$.

   (c) $x[n] = 5\cos(0.5\pi n)$, $n = 0, \ldots, 25$.

   (d) $x[n] = 3\sin(0.65\pi n)$, $n = 0, \ldots, 40$.

   (e) $x[n] = \sin(0.45\pi n)$, $n = 0, \ldots, 10$.

26. Let $x[n] = 2\cos(0.1\pi n) + \sin(0.05\pi n)$, $0 \le n \le 60$.

   (a) Using the `interp` function, interpolate using $I = 3$, $I = 6$, and $I = 9$. Stem plot the original and interpolated signals.

   (b) Using the second output argument of the `interp` function, plot the frequency response of the lowpass filter used in each of the above interpolations.

27. Generate a 101-length sequence $x[n]$ using the `fir2` function with normalized-frequency array `[0,0.1,0.2,0.6,0.65,0.7,1]` and the corresponding magnitude array `[1,1,1,1,0.5,0,0]`. The frequency normalization is with respect to π radians.

   (a) Compute and plot the magnitude spectra of $x[n]$.

   (b) Upsample $x[n]$ using $I = 2$ and plot the spectrum of the decimated signal.

   (c) Upsample $x[n]$ using $I = 3$ and plot the spectrum of the decimated signal.

   (d) Upsample $x[n]$ using $I = 4$ and plot the spectrum of the decimated signal.

   (e) Comment on your spectra.

28. Using the Parks–McClellan algorithm, design an interpolator that increases the input sampling rate by a factor of $I = 3$.
    (a) Determine the coefficients of the FIR filter that has 0.1 dB ripple in the passband and 50 dB attenuation in the stopband. Choose reasonable values for the band-edge frequencies.
    (b) Provide plots of the impulse and the log-magnitude responses.
    (c) Determine the corresponding polyphase structure for implementing the filter.
    (d) Let $x[n] = \cos(0.6\pi n)$. Generate 100 samples of $x[n]$ and process it using the above-designed filter to interpolate by $I = 2$ to obtain $x_I[n]$. Provide the stem plots of both sequences.

29. The ideal discrete-time interpolation formula to obtain $x[n]$ from the decimated sequence $x_D[n]$ is given in (12.28). In practice we use the windowed sinc interpolation approximation. Let $x_D[m] = \sin(\pi m/2)$, $0 \le m \le N$ with $N = 32$ and $D = 2$.
    (a) Let $g_L[n] = \frac{\sin(\pi n/D)}{\pi n/D}$, $-LD < n < LD$ be the windowed sinc interpolation function. Using (12.28) and the above windowed function for $L = 2$, obtain $x_K[n]$. Compare it with the exact interpolated sequence $x[n]$.
    (b) Another efficient approximation is due to Wang *et al.* (1992) and uses the DFT. The steps are: 1. Modify the sequence $x_D[m]$ to obtain $\tilde{x}_D[m] = x_D[m] - bm$, $0 \le m \le N$ where $b = (x_D[m] - x_D[0])/N$; 2. Take an N-point DFT of $\tilde{x}_D[m]$, $0 \le m < N$ to obtain $\tilde{X}_D[k]$; 3. Zero-pad the DFT $\tilde{X}_D[k]$ in the middle by $(D-1)N$ zeros to make an ND-point sequence $\tilde{X}[k]$ after equally distributing the DFT value $\tilde{X}_D[N/2]$ value between the two halves; 4. Take the ND-point IDFT of $\tilde{X}[k]$ to obtain $\tilde{x}[n]$ and append with $\tilde{x}[0]$; 5. Finally, recover the interpolated sequence as $x[n] = \tilde{x}[n] + (b/D)n$, $0 \le n \le DN$. For the given $x_D[m]$, D, and N, obtain the sequence $x[n]$. Compare it with the exact interpolation and comment.

30. We want to design a sampling rate converter that reduces the sampling rate by a factor of 2/5.
    (a) Using the Parks–McClellan algorithm, determine the coefficients of the FIR filter that has 0.1 dB ripple in the passband and 40 dB attenuation in the stopband. Choose reasonable values for the band-edge frequencies.
    (b) Provide plots of the impulse and the log-magnitude responses.
    (c) Specify the sets of the time-varying coefficients $g(m, n)$ and the corresponding coefficients in the polyphase filter realization.
    (d) Let $x[n] = \sin(0.3\pi n) + 2\cos(0.4\pi n)$. Generate 500 samples of $x[n]$ and process it using the above-designed filter to resample by 2/5 to obtain $x_R[m]$. Provide the stem plots of both sequences.

31. Using polyphase decomposition, develop the impulse response representation for the linear interpolation.

32. Consider the following filter:

$$H(z) = 1 - 9z^{-2} - 16z^{-3} - 9z^{-4} + z^{-6}.$$

    (a) Compute and plot its magnitude response.
    (b) Show that the filter is a lowpass half-band filter.

33. Design a half-band filter to satisfy the following requirements: $\omega_s = 0.58\pi$, $A_s = 55$ dB, $A_p = 0.5$ dB.

34. We want to design a decimator to reduce sampling rate from 1000 Hz to 50 Hz. Let the ripple parameters of the needed lowpass filter be $A_p = 0.5$ db and $A_s = 50$ dB.
    (a) Obtain a lowpass FIR filter for a single-stage decimator. Choose reasonable values for the band-edge frequencies. Plot the magnitude response of the filter. What is the computational complexity?
    (b) Obtain FIR filters for a two-stage decimator with $D_1 = 5$ and $D_2 = 4$ using IFIR filter design. Plot magnitude responses of each filter. What is the computational complexity?
    (c) Obtain the equivalent single-stage filter from (b) above and compute its computational complexity.

Assessment problems

35. Using the `src` function, resample the following sequences using the given parameters D and the offset k. Using the `stem` function, plot the original and downsampled signals.
    (a) $x[n] = \cos(0.4\pi n)$, $0 \le n \le 40$, $D = 2$, $k = 0$, and $k = 1$.
    (b) $x[n] = \cos(0.15\pi n)$, $0 \le n \le 30$, $D = 3$, $k = 0$, and $k = 1$.
    (c) $x[n] = nu[n] - 2nu[n - 20] + u[n - 40]$, $D = 3$, $k = 0$, and $k = 2$.
    (d) $x[n] = \cos(0.45\pi n)$, $0 \le n \le 90$, $D = 2$, $k = 0$, and $k = 1$.
    (e) $x[n] = 1 - \cos(0.05\pi n)$, $0 \le n \le 100$, $D = 5$, $k = 1$ and $k = 3$.

36. Using the `decimate` function, resample the following sequences using the given parameters D. Using the `stem` function, plot the original and decimated signals. Obtain results using both the default IIR and FIR decimation filters and comment on the results.
    (a) $x[n] = \sin(0.1\pi n)$, $0 \le n \le 100$, $D = 5$.
    (b) $x[n] = \cos(0.015\pi n)$, $0 \le n \le 100$, $D = 10$.
    (c) $x[n] = \sin(n/11) + 2\cos n/31$, $0 \le n \le 150$, $D = 6$.
    (d) $x[n] = \sin(0.025\pi n)$, $0 \le n \le 100$, $D = 8$.
    (e) $x[n] = 1, 2, 3, 4, 3, 2, 1, 0$, periodic with period 8, $0 \le n \le 64$, $D = 2$.

37. Generate a 201-length sequence $x[n]$ using the `fir2` function with normalized-frequency array `[0,0.1,0.18,0.2,1]` and the corresponding magnitude array `[1,1,1,0,0]`. The frequency normalization is with respect to π radians. Use the default FIR filter in the following decimations.
    (a) Compute and plot the magnitude spectrum of $x[n]$.
    (b) Decimate $x[n]$ using $D = 3$ and plot the spectrum of the decimated signal.
    (c) Decimate $x[n]$ using $D = 4$ and plot the spectrum of the decimated signal.
    (d) Decimate $x[n]$ using $D = 5$ and plot the spectrum of the decimated signal.
    (e) Comment on your spectra.

38. Determine the output $y[n]$ in terms of the input $x[n]$ for the following system

$$x[n] \rightarrow \boxed{\uparrow 2} \rightarrow \boxed{H(z)} \rightarrow \boxed{\downarrow 5} \rightarrow \boxed{\uparrow 2} \rightarrow y[n].$$
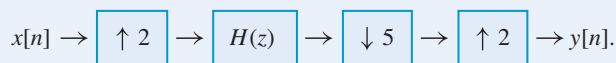
39. Generate 100 samples of the signal $x[n] = \sin(\pi/2n)$. We want to interpolate this signal using $I = 5$.

(a) Design a 45-order lowpass filter in Figure 12.9 using the Parks–McClellan algorithm. Choose the appropriate band-edge frequencies and equal ripple values to obtain $h[n]$.

(b) Implement the interpolator of Figure 12.9 to obtain $x_I[n]$. Use the `sre` function. Plot steady-state values of $x_I[n]$.

(c) Implement the interpolator using the `upfirdn` function to obtain $x_I[n]$ and plot steady-state values of $x_I[n]$.

(d) Implement the interpolator using the `interp` function to obtain $x_I[n]$ and plot steady-state values of $x_I[n]$.

(e) Compare your results in parts (b), (c), and (d) and comment.

40. Let $x[n] = \cos(0.65\pi n)$, $n = 0, \ldots, 80$.

(a) Using the `interp` function, interpolate using $I = 5$, $I = 10$, and $I = 15$. Stem plot the original and interpolated signals.

(b) Using the second output argument of the `interp` function, plot the frequency response of the lowpass filter used in each of the above interpolations.

41. Generate a 101-length sequence $x[n]$ using the `fir2` function with normalized-frequency array `[0,0.1,0.3,0.5,0.55,0.6,1]` and the corresponding magnitude array `[1,2,1.5,1,0.5,0,0]`. The frequency normalization is with respect to π radians.

(a) Compute and plot the magnitude spectrum of $x[n]$.

(b) Upsample $x[n]$ using $I = 2$ and plot the spectrum of the decimated signal.

(c) Upsample $x[n]$ using $I = 3$ and plot the spectrum of the decimated signal.

(d) Upsample $x[n]$ using $I = 4$ and plot the spectrum of the decimated signal.

(e) Comment on your spectra.

42. The impulse response of a family of cubic or third-order (TOH) interpolators for a factor I is given by

$$
g_{TOH}[n] = \begin{cases} (a+2)\left|\frac{n}{I}\right|^3 - (a+3)\left|\frac{n}{I}\right|^2 + 1, & 0 \le |n| \le I, \\ a\left|\frac{n}{I}\right|^3 - 5\left|\frac{n}{I}\right|^2 + 8a\left|\frac{n}{I}\right| - 4a, & I \le |n| \le 2I \\ 0. & \text{otherwise} \end{cases}
$$

(a) On the same graph plot magnitude responses of the ideal and TOH interpolators for $I = 3$.

(b) Implement the TOH interpolator for $x_D[m] = \sin(0.5\pi n)$, $0 \le n \le 50$ and $I = 3$ to obtain $x[n]$. Compare it with exact interpolation and comment on the cubic interpolation.

(c) Repeat (a) for $I = 5$.

43. Consider two sequences $x_1[n]$ and $x_2[n]$,

$$
x_1[n] = \max(20 - |n|, 0),
$$

$$
x_2[n] = \min(|n|, 20), \; 0 \le |n| \le 20.
$$

Using the `resample` function with default parameters:

(a) Resample the sequence $x_1[n]$ at $3/2$ times the original rate to obtain $x_{I_1}[m]$ and provide the `stem` plots of both sequences.

(b) Resample the sequence $x_2[n]$ at $3/2$ times the original rate to obtain $x_{I_2}[m]$ and provide the `stem` plots of both sequences.

(c) Explain why the resampled plot of $x_{I_2}[n]$ has inaccuracies near the boundaries that $x_{I_1}[m]$ does not have.

(d) Plot the frequency response of the filter used in the resampling operation.

44. Let $x[n] = \cos(0.4\pi n) + \cos(0.6\pi n)$. Using the `resample` function with default parameters:

(a) Resample the sequence $x[n]$ at $4/5$ times the original rate to obtain $x_{I_1}[m]$ and provide the `stem` plots of both sequences.

(b) Resample the sequence $x[n]$ at $5/4$ times the original rate to obtain $x_{I_2}[m]$ and provide the `stem` plots of both sequences.

(c) Resample the sequence $x[n]$ at $2/3$ times the original rate to obtain $x_{I_3}[m]$ and provide the `stem` plots of both sequences.

45. Generate 101 samples of a sequence $x[n]$ using the `fir2` function with parameters `[0,0.1,0.2,0.5,0.55.0.6,1]` and `[0,0.5,1,1,0.5,0,0]`.

(a) Compute and plot the magnitude spectrum of $x[n]$.

(b) Resample $x[n]$ by a factor of $4/3$ and plot the spectrum of the resulting signal.

(c) Resample $x[n]$ by a factor of $3/4$ and plot the spectrum of the resulting signal.

(d) Resample $x[n]$ by a factor of $3/5$ and plot the spectrum of the resulting signal.

(e) Comment on your spectra.

46. Consider the following filter

$$H(z) = 3 - 19z^{-2} - 32z^{-3} - 19z^{-4} + 3z^{-6}.$$

(a) Compute and plot its magnitude response.

(b) Show that the filter is a lowpass half-band filter.

47. Design a half-band filter to satisfy the following requirements: $\omega_s = 0.6\pi$, $A_s = 60$ dB, $A_p = 0.2$ dB.

48. We want to design a decimator to reduce sampling rate from 1500 Hz to 100 Hz. Let the ripple parameters of the needed lowpass filter be $A_p = 0.5$ db and $A_s = 50$ dB.

(a) Obtain a lowpass FIR filter for a single-stage decimator. Choose reasonable values for the band-edge frequencies. Plot the magnitude response of the filter. What is the computational complexity?

(b) Obtain FIR filters for a two-stage decimator with $D_1 = 5$ and $D_2 = 3$ using IFIR filter design. Plot magnitude responses of each filter. What is the computational complexity?

(c) Obtain the equivalent single-stage filter from (b) above and compute its computational complexity.

Review problems

49. A signal $x[n]$ is to be resampled by a factor of $3/2$. It has a total bandwidth of 0.8π, but we want to preserve frequencies only up to 0.75π and require that the band up to 0.6π be free of aliasing in the resampled signal.

(a) Using the Parks–McClellan algorithm, determine the coefficients of the FIR filter that has 0.11 dB ripple in the passband and 40 dB attenuation in the stopband.

(b) Provide plots of the impulse and the log-magnitude responses.

(c) Determine the corresponding polyphase structure of the resampling filter.

(d) Using the `fir2` function, generate a 101-point sequence $x[n]$ using normalized frequency and magnitude parameters `f=[0,.3,.7,.75,.8,1]` and `[0.7,1,1,0.5,0,0]`, respectively. Process $x[n]$ using the above-designed filter to resample it by a factor of 3/2 to obtain $x_R[m]$. Provide the spectral plots of both sequences.

50. A signal $x[n]$ is to be resampled by a factor of 3/8. It has a total bandwidth of 0.5π, but we want to preserve frequencies only up to 0.35π and require that the band up to 0.3π be free of aliasing in the resampled signal.

(a) Using the Parks–McClellan algorithm, determine the coefficients of the FIR filter that has 0.5 dB ripple in the passband and 50 dB attenuation in the stopband.

(b) Provide plots of the impulse and the log-magnitude responses.

(c) Determine the corresponding polyphase structure of the resampling filter.

(d) Using the `fir2` function, generate a 101-length sequence $x[n]$ using normalized frequency and magnitude parameters `f=[0,.25,0.5,0.55,0.6,1]` and `[1,1,1,0.5,0,0]`, respectively. Process $x(n)$ using the above-designed filter to resample it by a factor of 3/8 to obtain $x_R[m]$. Provide the spectral plots of both sequences.

13 Random signals

In this chapter we are concerned with probability models for the mathematical description of random signals. We start with the fundamental concepts of random experiment, random variable, and statistical regularity and we show how they lead into the concepts of probability, probability distributions, and averages, and the development of probabilistic models for random signals. Then, we introduce the concept of stationary random process as a model for random signals, and we explain how to characterize the average behavior of such processes using the autocorrelation sequence (time-domain) and the power spectral density (frequency-domain). Finally, we discuss the effect of LTI systems on the autocorrelation and power spectral density of stationary random processes.

Study objectives

After studying this chapter you should be able to:

- Understand the concepts of randomness, random experiment, statistical variability, statistical regularity, random variable, probability distributions, and statistical averages like mean and variance.

- Understand the concept of correlation between two random variables, its measurement by quantities like covariance and correlation coefficient, and the meaning of covariance in the context of estimating the value of one random variable using a linear function of the value of another random variable.

- Understand the concept of a random process and the characterization of its average behavior by the autocorrelation sequence (time-domain) and power spectral density (frequency-domain), develop an insight into the processing of stationary processes by LTI systems, and be able to compute mean, autocorrelation, and power spectral density of the output sequence from that of the input sequence and the impulse response.

- Develop an understanding of white noise process, linear processes, autoregressive-moving average processes, and harmonic processes and in particular, be able to compute autocorrelation sequences of these processes given their parameters and vice versa.

Probability models and random variables

In the preceding chapters we have introduced mathematical models for deterministic signals and used them to analyze, design, and predict the performance of signal processing systems. The key feature of *deterministic* signal models is that for each value of time we have a rule which enables us to determine the precise value of the signal. In this sense, we can say that the future values of a deterministic signal can be predicted exactly from its past values.

However, in practical applications we encounter many signals which are not of this type. Let us consider, for example, a speech signal; since the purpose of speech is to convey information, a speech waveform must have the property that its future values cannot be predicted exactly by the listener. This behavior is typical of a wide range of practical signals which evolve in an unpredictable manner. Formally, we say that a signal is *random* if we cannot predict exactly its future values from its past values.

Since the values of a random signal cannot be precisely specified by a formula or rule, we *cannot* use the mathematical techniques developed for deterministic signals to study random signals. The mathematical tools pertinent to the study of random signals belong to the fields of probability and statistics.

13.1.1 ## Randomness and statistical regularity

In the study of probability and statistics, we consider experiments whose outcome *cannot* be predicted with certainty. Such experiments are called *random experiments*. Each experiment ends in an outcome, denoted by the Greek letter ζ, that cannot be determined with certainty before the experiment is performed. However, the experiment is such that the collection of all possible outcomes, called the *sample space S*, can be described and perhaps listed. The following examples illustrate what we mean by random experiments, outcomes, and their sample spaces:

- Two dice are cast and the total number of spots on the sides that are "up" are counted. The sample space is $S = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ and contains a finite number of outcomes.
- A fair coin is flipped successively at random until the first head is observed. If we let ζ denote the number of flips of the coin required, then the sample space is $S = \{1, 2, 3, \dots\}$, which consists of an infinite, but countable, number of outcomes.
- A wheel of fortune, which is calibrated from zero to unity, is spined once. The reading ζ of the wheel is a real number in the continuous interval $0 \leq \zeta < 1$. Therefore, the sample space S consists of an uncountably infinite number of outcomes.

Clearly, it is impossible to predict exactly the outcome of any of these experiments from past outcomes. However, experience shows that, if we repeat a random experiment a large number of times, under identical conditions, we can predict in advance from previous experience what will happen on the average. These empirical observations, which are illustrated below, lie at the root of the ideas of probability and statistics.
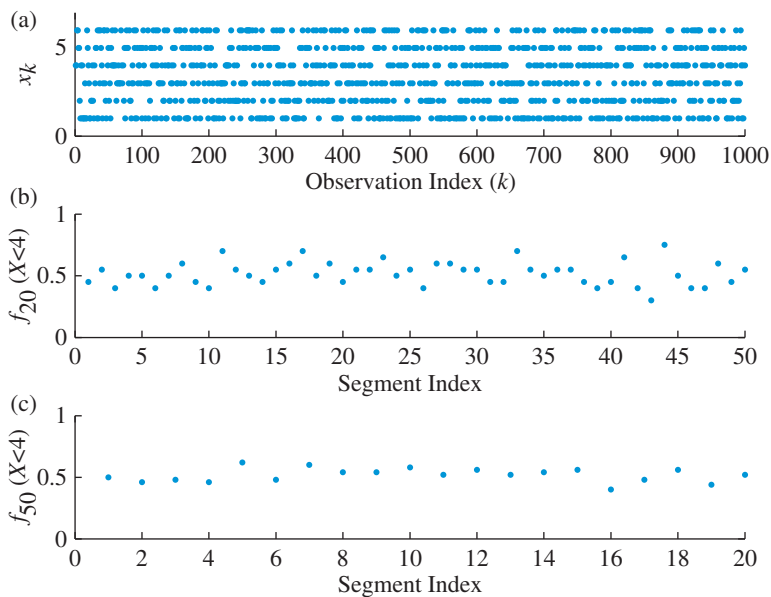
Figure 13.1 Randomness (statistical variability) and statistical regularity of the long-run relative frequency in a die throwing random experiment.

Figure 13.1(a) shows the results of 1000 throws of a die under identical conditions. In this random experiment, which has sample space $S = \{1, 2, 3, 4, 5, 6\}$, we are not interested in the actual value of the number X of points showing, but only in whether X is less than four ($X < 4$). In probability language, particular subsets of S are called *events*. The event ($X < 4$) is the subset $\{1, 2, 3\}$ of S. The values of X on the first ten repetitions or *trials* of the experiment are as follows:

Trial number:	1	2	3	4	5	6	7	8	9	10
Value of X:	6	3	2	1	5	6	1	3	5	2

The event ($X < 4$) occurred on trials 2,3,4,7,8, and 10, that is, six times. We denote this for brevity by the formula $N(X < 4) = 6$. Dividing this by the number 10 of trials, we obtain the fraction 0.6, which we call the *relative frequency* of the event ($X < 4$). In general, if an event A occurs $N(A)$ times in N trials of an experiment, the relative frequency of the event is given by

$$f_N(A) \triangleq \frac{N(A)}{N} = \frac{\text{Number of occurrences of event } A}{\text{Total number of trials}} \qquad (13.1)$$

We next divide the 1000 observations into 50 segments of $N = 20$ trials each and we determine the relative frequency of the event ($X < 4$) for each sequence of 20 trials. The values obtained are displayed in Figure 13.1(b), where each point corresponds to one sequence of 20 trials. Figure 13.1(c) shows the relative frequencies of ($X < 4$) in 20 sequences of $N = 50$ trials each. Careful examination of these figures shows that, with longer sequences, the relative frequency of the event ($X < 4$) is less variable and hence more predictable than when the sequences are shorter. Thus, while we cannot predict exactly the outcome of any

particular trial, we can predict quite precisely the relative frequency of an event in a large number of trials. This stability of long-run relative frequencies, which is known as *statistical regularity*, provides the basis for the foundation of a mathematical theory for random signals.

13.1.2 Random variables

As a motivation for a definition of random variable, let us consider an example. We throw a coin three times, and we observe the sequence of heads and tails. The sample space is

$$S = \{HHH, HHT, HTT, HTH, TTT, TTH, THH, THT\}. \tag{13.2}$$

Examples of random variables defined on S are (1) the total number of heads $X = \{0, 1, 2, 3\}$, (2) the total number of tails $Y = \{0, 1, 2, 3\}$, and (3) the number of heads minus the number of tails $Z = X - Y = \{-3, -1, 1, 3\}$. Each of these is a real-valued function defined on the sample space S; that is, each is a rule that assigns a real number to every point (outcome) $\zeta \in S$. Since the outcome ζ is random, the corresponding number $X(\zeta)$ is random as well.

In general, a *random variable* is a function from S to the real numbers. Typically, we use capital letters to denote random variables and lower-case letters for values they may take. For example, the equation $X = x$ states that the random variable X takes on the value x. For simplicity, we often drop the dependence on the outcome ζ. For technical reasons, we have to make a distinction between *discrete* and *continuous* random variables. A discrete random variable can take values from a finite or countably infinite set of numbers. In contrast, a continuous random variable can take as a value any real number in a specified range.

Since the value of a random variable is determined by the outcome of a random experiment, we can determine the relative frequency of the event $(X < a)$ for any value of a. There is ample empirical evidence that the relative frequency, which is usually very unstable for small values of N, tends to stabilize about some number, say $P(A)$, as N increases. The number $P(A)$, such that $0 \leq P(A) \leq 1$, is called the *probability* of the event A. This is illustrated in Figure 13.2, which shows the relative frequency of the event $(X < 4)$ in a series of N throws of a die as a function of N.

A complete description of a random variable is given by specifying all the values it can take together with their associated probabilities. For discrete random variables with a finite number M of possible values this is easily done by specifying the pairs $\{x_i, p(x_i), i = 1, 2, \ldots, M\}$. For continuous random variables, which can take any value from an uncountably infinite number of real numbers in an interval, the probability of picking any specific number is zero. To get around this problem, instead of assigning probabilities to single points, we assign probabilities to intervals, and we do it by representing probabilities as areas over intervals. In the rest of this chapter, we concentrate on continuous random variables.

13.1.3 Probability distributions

In practice, to understand the nature of a random variable, we construct the *histogram* of a set of observations. To this end, we first divide the horizontal axis into intervals of
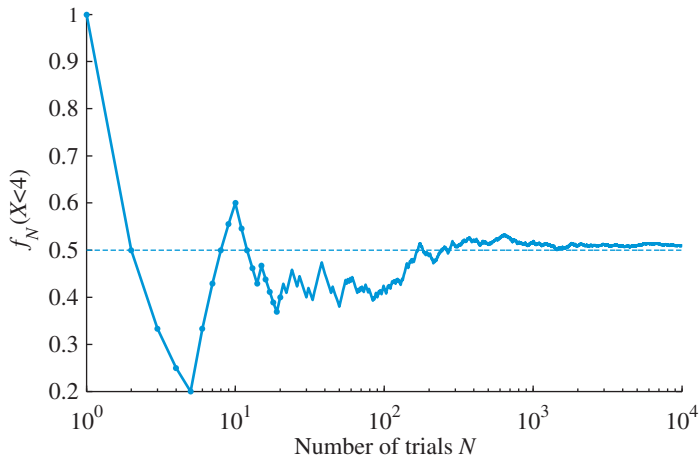
Figure 13.2 Relative frequency of the event $(X < 4)$ in a sequence of N throws of a die as a function of the number of trials N.

```
function [xo,px]=epdf(x,Nbins);
[nx,xo]=hist(x,Nbins);
dx=xo(3)-xo(2);
px=nx/(dx*length(x));
```

Figure 13.3 MATLAB function for computation of the empirical probability density function.

appropriate size Δx and we determine the number n_i of observations in the ith interval. Then we draw over each interval a rectangle with area proportional to the relative frequency $f_i = n_i/N$ of the observations. This is illustrated in Figure 13.4 for a data set obtained by recording the audio noise in the cockpit of an F-16 airplane (see Tutorial Problem 2). If we make the area of each rectangle equal to $n_i/(N\Delta x)$, the area under the histogram is equal to unity. Then, the area between a_1 and a_2 provides the percentage of observations within this interval. This can be done using the MATLAB function epdf given in Figure 13.3.

If we knew all observations that might conceptually occur, which are typically infinite, we could obtain a theoretical histogram by making Δx arbitrarily small. The result is a smooth continuous curve $f_X(x)$ called the *probability density function* (pdf) of random variable X. The name arises because the integral

$$\Pr(a_1 < x < a_2) = \int_{a_1}^{a_2} f_X(x)\mathrm{d}x \tag{13.3}$$

gives the probability that a value x lies between a_1 and a_2. Thus, if $f_X(a_1) > f_X(a_2)$, then values close to a_1 are more likely to occur than values close to a_2. Note that $a_1 = a_2 = a$, gives $\Pr(X = a) = 0$, as expected. Since X will take some value in the interval $-\infty < x < \infty$ with probability 1, the area under the whole curve must be equal to 1. It is important to realize that the function $f_X(x)$ does not represent the probability of any event, and that only when the $f(x)$ is integrated between two points do we obtain a probability.
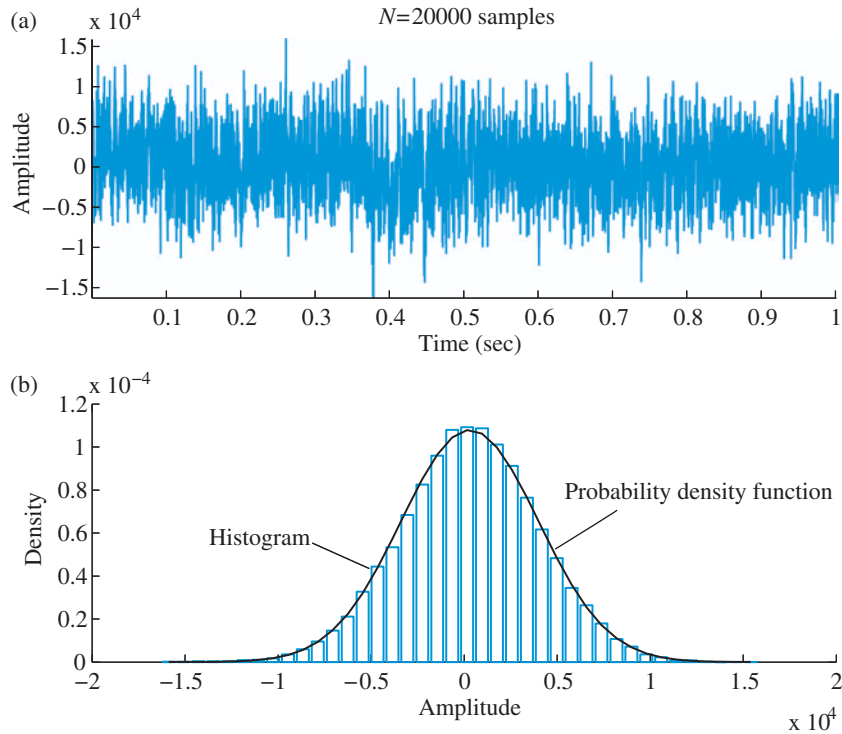
Figure 13.4 (a) Waveform of F-16 noise recorded at the co-pilot's seat with a sampling rate of 19.98 kHz using a 16 bit ADC. (b) Histogram and theoretical probability density function. (Courtesy of TNO, Soesterberg, The Netherlands.)

The *cumulative distribution function* (CDF) of a random variable X is defined by

$$F_X(a) \triangleq \Pr(x \leq a) = \int_{-\infty}^{a} f_X(x)dx, \tag{13.4}$$

which is the area under the curve $f_X(x)$ from $-\infty$ to a. Since $f_X(x) \geq 0$, the function $F_X(x)$ is monotonically increasing from $F_X(-\infty) = 0$ to $F_X(\infty) = 1$.

13.1.4   Statistical averages

Frequently, it is not possible to determine the distribution of a random variable exactly. Then it becomes necessary to summarize the key characteristics of the distribution by a few numbers. Simplest and most useful are *mean* and *variance*.

Mean value   The ordinary arithmetic average of a set of observations is given by

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i. \tag{13.5}$$

If we denote by $\tilde{x}_k$ the center of each histogram interval, we can rewrite (13.5) as

$$\bar{x} \simeq \frac{1}{N} \sum_k n_k \tilde{x}_k = \sum_k \tilde{x}_k \left(\frac{n_k}{N\Delta x} \right) \Delta x \triangleq \sum_k \tilde{x}_k \hat{f}(\tilde{x}_k) \Delta x, \qquad (13.6)$$

where $\hat{f}(\tilde{x})$ denotes the histogram. If we can make Δx arbitrarily small, the histogram $\hat{f}(\tilde{x})$ tends to pdf $f_X(x)$ and the summation becomes integration. Motivated by this intuitive explanation, we define the *mean value* or *expectation* of a random variable X as

$$m_x \triangleq \mathrm{E}(X) \triangleq \int_{-\infty}^{\infty} x f_X(x) \mathrm{d}x. \qquad (13.7)$$

The mean value provides a measure of the center or location of the distribution. Thus m_x is an "average" of the values that a random variable takes on, where each value is weighted by the probability that the random variable is equal to that value. The mean value, in the sense of (13.7), is a measure of where the values of the random variable X are "centered."

Variance The *variance* of a random variable X is defined by

$$\sigma_x^2 \triangleq \mathrm{var}(X) \triangleq \mathrm{E}[(x - m_x)^2] = \int_{-\infty}^{\infty} (x - m_x)^2 f_X(x) \mathrm{d}x, \qquad (13.8)$$

and provides a measure of the spread or dispersion of the distribution about its mean value. Intuitively, the variance determines an interval around the mean where the values of the random variable are most likely to occur. A small variance indicates that X is more likely to assume values close to its mean. In contrast, a large variance indicates that the values of X are spread over a wider interval about the mean. Thus, we often use the variance as a measure of uncertainty or a measure of variability of a random variable.

A measure of spread, which has the same units as the original observations, is the *standard deviation* σ_x. It is defined as the positive square root of the variance

$$\sigma_x \triangleq \sqrt{\mathrm{var}(X)}. \qquad (13.9)$$

When there is no doubt about the considered random variable, we drop the subscript from the mean, variance, and standard deviation symbols.

Expectation Suppose now that we wish to determine the expected value of the random variable $Y = g(X)$, where g is some given function. Since $g(X)$ takes on the value $g(x)$ when X takes on the value x, it seems intuitive that $\mathrm{E}[g(X)]$ should be a weighted average of the possible values $g(x)$ with, for a given x, the weight given to $g(x)$ being equal to the probability (or probability density in the continuous case) that X will be equal to x. This suggests the following result

$$\mathrm{E}[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) \mathrm{d}x, \qquad (13.10)$$

which can be shown to be true for any continuous distribution. We note that if $g(x) = x$, then $E[g(X)] = E(X)$ is the mean of X. If $g(x) = (x - m_x)^2$, we have

$$\text{var}(X) = E[(X - m_x)^2] = \int_{-\infty}^{\infty} (x - m_x)^2 f_X(x) dx$$

$$= E(X^2) - 2m_x E(X) + m_x^2 = E(X^2) - m_x^2, \qquad (13.11)$$

that is, the variance is equal to the mean of the square minus the square of the mean.

13.1.5   Two useful random variables

We next discuss the properties of two random variables which are widely used in many random signal processing applications.

Uniform distribution A random variable X is said to be uniformly distributed over the interval (a, b), if $f_X(x)$ is given by

$$f_X(x) = \begin{cases} 1/(b-a), & \text{if } a < x < b \\ 0. & \text{otherwise} \end{cases} \qquad (13.12)$$

The mean and variance of a uniform (a, b) random variable are obtained as follows

$$E(X) = \frac{1}{b-a} \int_a^b x dx = \frac{b^2 - a^2}{2(b-a)} = \frac{b+a}{2}, \qquad (13.13)$$

$$E(X^2) = \frac{1}{b-a} \int_a^b x^2 dx = \frac{a^2 + b^2 + ab}{3}, \qquad (13.14)$$

and so

$$\text{var}(X) = E(X^2) - [E(X)]^2 = \frac{1}{12}(b-a)^2. \qquad (13.15)$$

A random variable X uniformly distributed over the interval (a, b) is denoted by $X \sim U(a, b)$. If X is $U(0, 1)$, approximate values of X can be simulated on most computers using a random number generator. In fact, it should be called a pseudo-random number generator because the programs that produce the random numbers are usually such that if the starting number (seed) is known, all subsequent numbers in the sequence may be determined by simple arithmetic operations. Yet, despite their deterministic origin, pseudo-random numbers do behave as if they were truly random. The MATLAB function `rand` generates random numbers according to $X \sim U(0, 1)$.

Normal distribution A random variable X is said to be normally distributed with mean m and variance σ^2, denoted $X \sim N(m, \sigma^2)$ or $N(x; m, \sigma^2)$, if its probability density function is given by

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-m)^2/2\sigma^2}. \qquad -\infty < x < \infty \qquad (13.16)$$

The normal or Gaussian density is a bell-shaped curve that is symmetric about its mean $E(X) = m$. The width of the distribution is determined by its variance $\text{var}(X) = \sigma^2$ (see
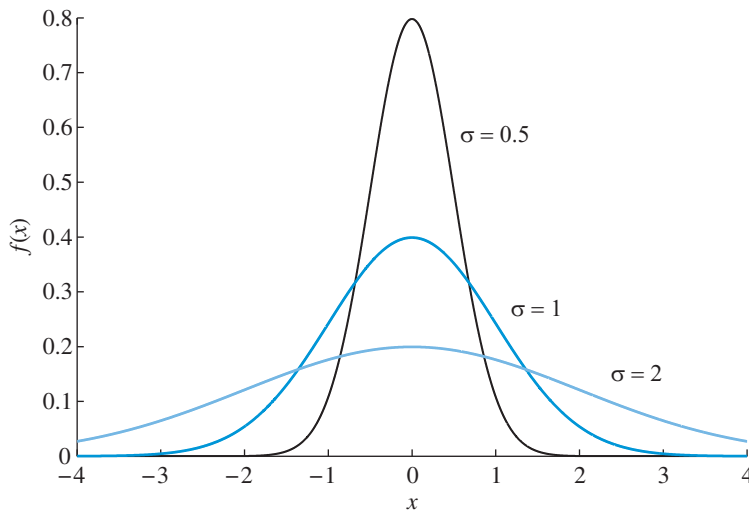
Figure 13.5 The probability density function of a normal distribution for various values of the standard deviation. The spread of the distribution increases with increasing σ; the height is reduced because the area is always equal to unity.

Figure 13.5). We note that if σ^2 is small, the random variable X assumes values close to its mean. A large σ^2 indicates that the values of X are more widely spread around its mean. In this sense, we can think of σ^2 as a measure of uncertainty for the unobserved values of X. The normal distribution is a good example of a case where the mean and standard deviation are good measures of the center of the pdf and its spread about the center. However, there are situations where the information provided by the mean and standard deviation about the shape of the pdf can be misleading (see Tutorial Problem 3).

An important property of normal random variables is that if X is normal with mean m and variance σ^2, then $aX + b$ is normally distributed with mean $am + b$ and variance $a^2\sigma^2$ (see Tutorial Problem 4), that is

$$X \sim N(m, \sigma^2) \Rightarrow Y = aX + b \sim N(am + b, a^2\sigma^2). \tag{13.17}$$

One implication of this result is that the random variable

$$Z = \frac{X - m}{\sigma} \tag{13.18}$$

is normally distributed with zero mean and unit variance. Such a random variable is said to have a *unit* or *standard* normal distribution. The MATLAB function `randn` generates standard normal random numbers. To generate N random numbers for a Gaussian random variable X with mean m and variance σ^2 we use `x = sigma*randn(N,1)+mu`.

The wide applicability of normal random variables in probability and statistics arises from two important results. The first is that any linear combination of normal random variables is itself a normal random variable. The second is the central limit theorem, which asserts that the sum of a large number of independent random variables has approximately a normal distribution.

13.2   Jointly distributed random variables

In many random experiments, we are interested in the relationships between two or more random variables. In theory, multiple random variables are completely characterized by their joint probability distribution. In practice, we typically use their means, variances, and pairwise covariances.

13.2.1   Probability functions

The best way to bring out the relationship between two variables X and Y is by a scatter plot, that is, a graph where each observation (x_i, y_i) of (X, Y) is represented by a dot. The "swarm" of points in a scatter plot reveals the relationship between the two variables. This is illustrated in Figure 13.6(a), where the variable X represents the height of a father and Y the height of his son, both measured in inches. There are several ways to characterize the information provided by a scatter diagram.

One way is to construct a two-dimensional histogram of the data. First, the (x, y) plane is divided into a rectangular grid, where each cell has area $\Delta x \times \Delta y$. Then, on top of each cell we raise a rectangular column with volume equal to $n_{ij}/(N\Delta x\Delta y)$, where n_{ij} is the number of points in the ijth cell. This process is illustrated in Figure 13.6(b); see Tutorial Problem 5 for details. If we knew all observations that might conceptually occur, we could obtain a smooth function $f(x, y)$ by making Δx and Δy arbitrarily small. The pair of random variables X and Y is then completely described by the *joint probability density function $f_{X,Y}(x, y)$*. The volume between the $f_{X,Y}(x, y)$ surface and the (x, y) plane is equal to 1. The volume under the probability density function $f_{X,Y}(x, y)$ above a two-dimensional
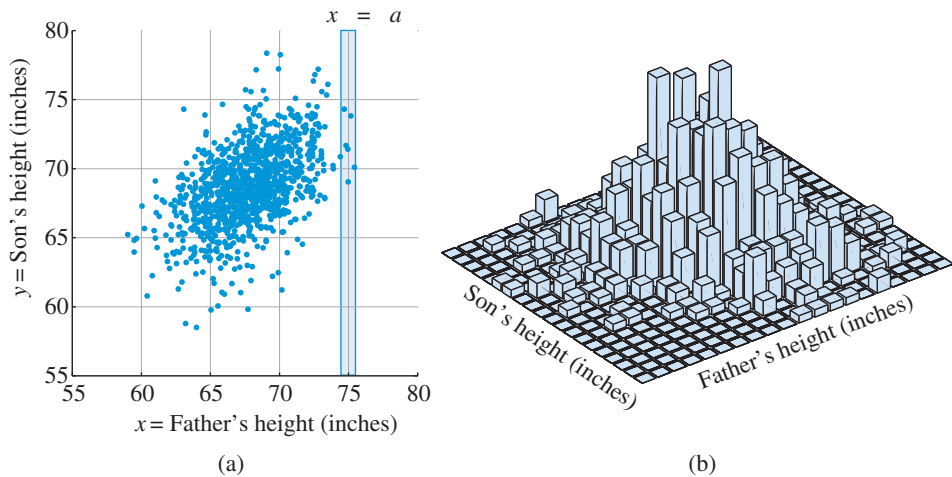
Figure 13.6 (a) A scatter plot of the heights of 1078 sons versus the heights of their fathers, and (b) the corresponding two-dimensional (2-D) histogram.

area A gives the probability for an observation (x, y) to be in A. The distributions of random variables X and Y are obtained by integration as follows:

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)dy \quad \text{and} \quad f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x, y)dx, \tag{13.19}$$

and are sometimes called the *marginal* distributions of X and Y, respectively.

Expectation The expectation of the random variable $Z = g(X, Y)$, where g is some given function, can be evaluated using the formula

$$E[g(X, Y)] = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y)f_{X,Y}(x, y)dxdy, \tag{13.20}$$

which can be intuitively justified following the reasoning leading to (13.10). As an example, let $Z = aX + bY$ where a and b are constants. Using (13.20) gives

$$E(aX + bY) = a \iint_{-\infty}^{\infty} xf_{X,Y}(x, y)dxdy + b \iint_{-\infty}^{\infty} yf_{X,Y}(x, y)dxdy. \tag{13.21}$$

The first term on the right hand side is a times the expectation of the function $g_1(x, y) = x$, and the second is b times the expectation of the function $g_2(x, y) = y$. Thus, we obtain the following linearity property of expectation:

$$E(aX + bY) = aE(X) + bE(Y). \tag{13.22}$$

Statistical independence If we consider a narrow vertical strip raised over the range $a < x < a + \Delta a$, we can determine the histogram of y for all values within the strip (see Figure 13.6(a)). As Δa becomes arbitrarily small, we obtain the *conditional probability density function* of Y given that $X = a$. We write $f_{Y|X}(Y|X = a)$, where the symbol "|" stands for the word "given." In general, the conditional distribution changes with $X = a$. If $f_{Y|X}(y|x) = f_Y(y)$, we say that X and Y are *statistically independent*. From these interpretations, we conclude that

$$f_{X,Y}(x, y) = f_{Y|X}(y|x)f_X(x) = f_{X|Y}(x|y)f_Y(y). \tag{13.23}$$

If X and Y are statistically independent, we obtain the simplified relation

$$f_{X,Y}(x, y) = f_X(x)f_Y(y), \tag{13.24}$$

that is, the joint probability density may be obtained by multiplying the individual densities. In this case, the random variables X and Y are statistically independent, in the sense that the likelihood of the values of one does not depend upon the likelihood of the other. In other words, there is little (if any) that can be said about the value taken by one random variable when the value taken by the other is known.

13.2.2   Covariance and correlation

Suppose that we wish to summarize numerically the information provided by the scatter plot in Figure 13.6(a). We can use the mean and standard deviation of X and Y values to describe the center of the cloud and its horizontal and vertical spread. However, these parameters do *not* describe the relationship between the two variables. We next introduce some quantities which can be used to provide a measure of the association between two random variables X and Y.

Covariance   The *covariance* of two random variables X and Y is defined by

$$c_{xy} \triangleq \text{cov}(X, Y) \triangleq E[(X - m_x)(Y - m_y)]. \tag{13.25}$$

Using the linearity property of expectation (13.21) we can easily show that

$$\text{cov}(X, Y) = E(XY) - E(X)E(Y). \tag{13.26}$$

Two random variables with zero covariance are said to be *uncorrelated*. Thus,

$$X, Y \text{ uncorrelated} \iff \text{cov}(X, Y) = 0 \text{ or } E(XY) = E(X)E(Y). \tag{13.27}$$

If X and Y are independent, that is $f_{X,Y}(x, y) = f_X(x)f_Y(y)$, we can show that $E(XY) = E(X)E(Y)$, that is X and Y are uncorrelated. The reverse is not always true; that is, $\text{cov}(X, Y) = 0$ does not always imply that the random variables X and Y are independent (see Tutorial Problem 6).

To gain insight into the meaning of the covariance between X and Y, we consider the scatter plot shown in Figure 13.7 and we draw a vertical line at $\bar{x}$ and a horizontal line at $\bar{y}$, where $\bar{x}$ and $\bar{y}$ are the averages of the two sets of data. The two lines divide the plot
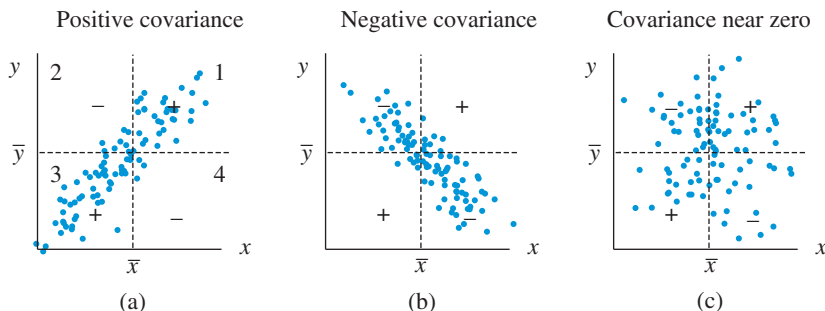
Figure 13.7 Examples of scatter diagrams. The covariance measures the extent to which the scatter diagram is packed in around a line. If the sign is positive, the line slopes up. If the sign is negative the line slopes down. If the covariance is near zero, there is no "obvious" or "preferred" direction.

into four quadrants. It is clear that the products $(x_i - \bar{x})(y_i - \bar{y})$ are positive in the first and third quadrants, and negative in the second and fourth quadrants. If there are more points in the first and third quadrant, the relationship between X and Y is positive (as X increases Y increases), and the sum $\sum_i (x_i - \bar{x})(y_i - \bar{y})$ is likely to be positive (see Figure 13.7(a)). Conversely, if the relationship between X and Y is negative (as X increases Y decreases), then there are more points in the second and fourth quadrants, and the sum $\sum_i (x_i - \bar{x})(y_i - \bar{y})$ is likely to be negative (see Figure 13.7(b)). A value of the sum close to zero, which results from a "circular" swarm of points, does not provide any indication about how the changes in X and Y are related (see Figure 13.7(c)). Thus, we can measure the association between the two random variables using the average of the observation products $(x_i - \bar{x})(y_i - \bar{y})$ or the expectation of the centered random variables product $(X - m_x)(Y - m_y)$.

Correlation coefficient   The value of $\mathrm{cov}(X, Y)$ does not tell us much about the "strength" of the relationship between X and Y because it depends upon the units of measurement. Indeed, we can easily show that for any constants a and b, we have $\mathrm{cov}(aX, bY) = ab\,\mathrm{cov}(X, Y)$. To avoid this drawback we define the normalized random variables

$$\tilde{X} \triangleq \frac{X - m_x}{\sigma_x} \quad \text{and} \quad \tilde{Y} \triangleq \frac{Y - m_y}{\sigma_y}, \tag{13.28}$$

which have zero mean and unit variance. The covariance between the normalized variables

$$\rho_{xy} \triangleq \mathrm{cov}(\tilde{X}, \tilde{Y}) = \frac{\mathrm{cov}(X, Y)}{\sqrt{\mathrm{var}(X)}\sqrt{\mathrm{var}(Y)}} = \frac{c_{xy}}{\sigma_x \sigma_y} \tag{13.29}$$

is known as the *correlation coefficient* between X and Y. Since the variance of any random variable cannot be negative, we have

$$\mathrm{Var}(\tilde{X} \pm \tilde{Y}) = \mathrm{Var}(\tilde{X}) + \mathrm{Var}(\tilde{Y}) \pm 2\mathrm{cov}(\tilde{X}, \tilde{Y}) = 2(1 \pm \rho_{XY}) \geq 0. \tag{13.30}$$

From the last inequality we see that the correlation coefficient can only take values in the following range

$$-1 \leq \rho_{xy} \leq 1. \tag{13.31}$$

These properties make ρ_{xy} a useful quantity for measuring both the direction and the strength of the *linear relationship* between two random variables. We stress that $\rho_{xy} = 0$ does not necessarily imply that X and Y are not related. Since $\mathrm{cov}(X, Y)$ and ρ_{xy} measure only linear relationships, it simply states that they are not linearly related. As the clustering of a scatter diagram around a line becomes tighter, $|\rho_{xy}|$ gets closer to one.

In summary, we can use the covariance to measure the "direction" of the relationship between two variables. When a scatter diagram is tightly clustered around a line, there is a strong linear association between the variables. Covariance is a measure of the *linear relationship* between two random variables. If the relationship between the random variables is nonlinear, the covariance may not reflect the strength of this relationship.

Orthogonal random variables   We now introduce another related concept, starting with the relationship

$$\mathrm{E}[(X + Y)^2] = \mathrm{E}(X^2) + \mathrm{E}(Y^2) + 2\mathrm{E}(XY). \tag{13.32}$$

Table 13.1 Relationship between orthogonal and uncorrelated random variables.		
Correlation:	$r_{xy} = E(XY)$	$r_{xy} = 0 \Rightarrow X, Y$ orthogonal
Covariance:	$c_{xy} = E[(X - m_x)(Y - m_y)]$	$c_{xy} = 0 \Rightarrow X, Y$ uncorrelated

If $E(XY) = 0$, we have $E[(X + Y)^2] = E(X^2) + E(Y^2)$, which resembles the Pythagorean theorem. With this motivation, we can think of X and Y as vectors with lengths $\sqrt{E(X^2)}$ and $\sqrt{E(Y^2)}$, and inner product $E(XY)$. Therefore, two random variables X and Y are called *orthogonal*, denoted by $X \perp Y$, if $E(XY) = 0$, symbolically,

$$X \perp Y \Leftrightarrow E(XY) = 0. \tag{13.33}$$

We note that if X and Y are uncorrelated, then $[X - E(X)] \perp [Y - E(Y)]$. Also, if X and Y are uncorrelated and $E(X) = 0$ or $E(Y) = 0$ then $X \perp Y$. The ratio

$$\frac{E(XY)}{\sqrt{E(X^2)}\sqrt{E(Y^2)}} \tag{13.34}$$

is the cosine of the angle θ between the "vectors" X and Y. We note that ρ_{xy} is equal to the cosine of the angle between $X - E(X)$ and $Y - E(Y)$; hence, $-1 \leq \rho_{xy} \leq 1$.

Correlation The quantity $E(XY) \triangleq R_x$ is called the *correlation* of X and Y. This terminology may create some confusion because the condition $E(XY) = 0$ (zero correlation) implies that X and Y are orthogonal, whereas the condition $cov(X, Y) = 0$ implies that X and Y are uncorrelated. The two properties are identical if $E(X) = E(Y) = 0$. These distinctions are summarized in Table 13.1.

13.2.3 Linear combinations of random variables

We often need to determine the mean and variance of the linear combination

$$Y = a_1 X_1 + a_2 X_2, \tag{13.35}$$

where a_1, a_2 are constants and X_1, X_2 are random variables. The mean is given by

$$m_y \triangleq E(Y) = a_1 E(X_1) + a_2 E(X_2) \triangleq a_1 m_1 + a_2 m_2. \tag{13.36}$$

To compute the variance, we first subtract (13.36) from (13.35) to determine $Y - m_y$, and then we square both sides. This yields

$$(Y - m_y)^2 = a_1^2 (X_1 - m_1)^2 + a_2^2 (X_2 - m_2)^2 + 2a_1 a_2 (X_1 - m_1)(X_2 - m_2).$$

Using the linearity property of the expectation (13.22), we have

$$var(Y) = a_1^2 var(X_1) + a_2^2 var(X_2) + 2a_1 a_2 cov(X_1, X_2). \tag{13.37}$$

We note that the mean of Y can be expressed in terms of the means of X_1 and X_2. However, we cannot determine the variance of Y if only the variances of X_1 and X_2 are known; we need the covariance between X_1 and X_2. If the random variables are uncorrelated, that is, $\text{cov}(X_1, X_2) = 0$, we have

$$\text{var}(Y) = a_1^2 \text{var}(X_1) + a_2^2 \text{var}(X_2), \tag{13.38}$$

which is significantly simpler than (13.37). We shall see below that assuming uncorrelated random variables simplifies the solution of many problems.

When we deal with many random variables $X_1, X_2, \ldots, X_p$ it is convenient to organize them as a p-dimensional random vector

$$x \triangleq \begin{bmatrix} X_1 & X_2 & \ldots & X_p \end{bmatrix}^{\text{T}}. \tag{13.39}$$

The mean values $m_i = \text{E}(X_i)$, $i = 1, 2, \ldots, p$ can be organized as a $p \times 1$ vector m_x, called the *mean vector*, as follows:

$$m_x \triangleq \begin{bmatrix} m_1 & m_2 & \cdots & m_p \end{bmatrix}^{\text{T}}, \tag{13.40}$$

while the covariances $c_{ij} \triangleq \text{cov}(X_i, X_j)$, for all $i, j = 1, 2, \ldots, p$ can be organized as a $p \times p$ matrix C_x, called the *covariance matrix*, as follows

$$C_x \triangleq \begin{bmatrix} c_{11} & c_{12} & \ldots & c_{1p} \\ c_{21} & c_{22} & \ldots & c_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ c_{p1} & c_{p2} & \ldots & c_{pp} \end{bmatrix}. \tag{13.41}$$

Since $\text{cov}(X_i, X_j) = \text{cov}(X_j, X_i)$, the covariance matrix is symmetric. For uniformity, we often use c_{kk} instead of σ_k^2. The linear combination of p random variables and its mean can also be expressed more concisely using vector notation as follows:

$$Y = \sum_{i=1}^{p} a_i X_i = a^{\text{T}} x, \tag{13.42}$$

$$\text{E}(Y) = \sum_{i=1}^{p} a_i \text{E}(X_i) = a^{\text{T}} m_x. \tag{13.43}$$

Simple matrix operations show that the variance in (13.37) can be expressed as

$$\text{var}(Y) = a_1^2 c_{11} + a_2^2 c_{22} + 2a_1 a_2 c_{12} = \begin{bmatrix} a_1 & a_2 \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}. \tag{13.44}$$

Thus, in general, the variance of Y (13.42) can be expressed in matrix form as

$$\text{var}(Y) = \text{var}(a^{\text{T}} x) = a^{\text{T}} C_x a. \tag{13.45}$$

Since var$(Y) \geq 0$, for any random variable, we conclude that

$$a^T C_x a \geq 0. \tag{13.46}$$

A matrix C_x that satisfies (13.46) for any $a \neq 0$ is called *nonnegative definite*. In a similar way we can show that

$$E(Y^2) = a^T R_x a \geq 0, \tag{13.47}$$

where R_x is a $p \times p$ nonnegative definite matrix with elements $r_{ij} \triangleq E(X_i X_j)$, known as the *correlation* matrix.

Normal random vectors If the components of the random vector x in (13.42) are normally distributed, the random variable Y, defined by the linear combination $Y = a^T x$, is normal with mean and variance given by (13.43) and (13.45), respectively. More specifically, we have

$$X_k \sim N(m_k, \sigma_k^2) \Rightarrow Y = \sum_{k=1}^{p} a_k X_k \sim N(a^T m, a^T C a). \tag{13.48}$$

The proof of this important result can be found in Papoulis and Pillai (2002). The random vector X is normally distributed if the joint pdf is given by

$$f(x) = \frac{1}{(2\pi)^{p/2} |C|^{1/2}} \exp\left\{ -\frac{1}{2}(x-m)^T C^{-1}(x-m) \right\}, \tag{13.49}$$

where $|C|$ is the determinant of the covariance matrix. The covariance matrix for the two-dimensional case is given by

$$C = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}, \tag{13.50}$$

where we have used the fact that $\text{cov}(X_1, X_2) = \rho\sigma_1\sigma_2$. The locus of the points with constant pdf is an ellipse centered at the point (m_1, m_2). If $\rho = 0$, the axes of the ellipse are parallel to the x_1 and x_2 axes, and if $\rho \neq 0$, they are tilted. Figure 13.7(a–c) shows the scatter plots of 100 pairs of bivariate normal random variables with correlation coefficients $\rho = 0.9$, $\rho = -0.9$, and $\rho = 0$ (see Tutorial Problem 7).

13.3 Covariance, correlation, and linear estimation

In many problems of practical interest we wish to estimate (or guess or predict) the value taken by a random variable Y when the value taken by a related random variable X is known or measured. In general, an estimate of the value assumed by Y, say $\hat{y}$, can be expressed as a function of the value x assumed by X, as follows

$$\hat{y} = h(x). \tag{13.51}$$

In contrast, the random variable $\hat{Y}$ defined by

$$\hat{Y} = h(X) \tag{13.52}$$

is known as the *estimator*. Evaluating the estimator $\hat{Y}$ at a particular value of $X = x$ yields an *estimate* $\hat{y}$. The problem is to find a function $h(\cdot)$ that gives the "best" estimates $\hat{y}$ of Y. The first step in solving this problem is to determine what we mean by "best." There are several criteria that can be used for estimation problems, but the most widely used criterion of goodness in signal processing applications is the expected value of the square of the error

$$\varepsilon \triangleq Y - \hat{Y}. \tag{13.53}$$

This results in the *mean squared error* or *mean square error* (mse)

$$J \triangleq \mathrm{E}(\varepsilon^2) = \mathrm{E}[(Y - \hat{Y})^2]. \tag{13.54}$$

The next step is to specify a class of functions which lead to useful estimators that can be determined by solving simple optimization problems. In this context, we restrict $h(\cdot)$ to be a linear function of the form

$$\hat{Y} = aX + b, \tag{13.55}$$

where a and b are unknown constants. Strictly speaking, this is an *affine* function rather than a linear function; it is linear if $b = 0$. This terminology is common, however, and we will use it. The linear estimator (13.55), which is determined by a straight line, is a reasonable choice if the scatter diagram indicates that the random variables X and Y have a strong linear dependence.

Finally, we should find the values of a and b which minimize the mse

$$J(a, b) = \mathrm{E}[(Y - aX - b)^2]. \tag{13.56}$$

We usually find the optimal a and b by setting the derivatives of $J(a, b)$ with respect to a and b equal to zero and verifying that the found solution is a global minimum (see Tutorial Problem 13). However, we will use an alternative approach which expresses $J(a, b)$ into the sum and differences of positive terms.

We start by rewriting the error (13.53) in terms of mean-removed random variables

$$Y - (aX + b) = (Y - m_y) + m_y - [(aX + b) - (am_x + b)] + (am_x + b)$$
$$= (Y - m_y) - a(X - m_x) - (b - m_y + am_x).$$

If we substitute the last relation into (13.56), after some straightforward algebraic manipulations, we obtain the following expression for the mse

$$J(a, b) = \sigma_y^2 + a^2 \sigma_x^2 - 2ac_{xy} + (b - m_y + am_x)^2. \tag{13.57}$$

The first three terms in (13.57) do not depend on b, so we can choose b to minimize the last term. The minimum value of this squared term, which is zero, is attained if

$$b_o = m_y - am_x. \tag{13.58}$$

Substituting (13.58) into (13.57) yields

$$J(a, b_o) = \sigma_y^2 + a^2\sigma_x^2 - 2ac_{xy}. \tag{13.59}$$

We shall next determine the optimum a by completing the square in (13.59) to obtain the equivalent form

$$J(a, b_o) = \sigma_y^2 + (a^2\sigma_x^4 - 2ac_{xy}\sigma_x^2 + c_{xy}^2 - c_{xy}^2)/\sigma_x^2, \tag{13.60}$$

$$= \sigma_y^2 - \frac{c_{xy}^2}{\sigma_x^2} + \frac{(a\sigma_x^2 - c_{xy})^2}{\sigma_x^2}. \tag{13.61}$$

This expression is minimized if the last term is set equal to zero, which yields

$$a_o = \frac{c_{xy}}{\sigma_x^2} = \rho_{xy}\frac{\sigma_y}{\sigma_x}. \tag{13.62}$$

Therefore, the minimum mse for the optimum linear estimator is

$$J(a_o, b_o) = \sigma_y^2 - \frac{c_{xy}^2}{\sigma_x^2} = \sigma_y^2(1 - \rho_{xy}^2). \tag{13.63}$$

If we substitute (13.58) and (13.62) into (13.55), the *linear minimum mse* estimator can be expressed as

$$\hat{Y}_o = \frac{c_{xy}}{\sigma_x^2}(X - m_x) + m_y = \rho_{xy}\frac{\sigma_y}{\sigma_x}(X - m_x) + m_y. \tag{13.64}$$

For each observed value $X = x$, equation (13.64) gives the best estimate $\hat{y}_o$ of Y according to the mse criterion. We note that the straight line defined by (13.64) passes through the point (m_x, m_y). We also remark that a strong correlation, that is, a value of ρ_{xy} close to ± 1, implies that $J(a_o, b_o) = 0$ and we can accurately predict one variable from the other using a linear relationship. Indeed, if the random variables are linearly related, that is, $Y = aX + b$, we have $\rho_{xy} = 1$ if $a > 0$ (positive slope) and $\rho_{xy} = -1$ if $a < 0$ (negative slope). Here we see again, from a different point of view, that the correlation coefficient is a measure of the strength of the linear relation between X and Y.

Careful inspection of (13.64) and (13.63) shows that *all we need to determine the linear minimum mse estimator is knowledge of the first- and second-order statistics* $E(X)$, $E(Y)$, $E(X^2)$, $E(Y^2)$, *and* $E(XY)$. The variance shows how accurately we can "guess" the value of a random variable without any other information; the correlation between two random variables shows how accurately we can "guess" the value of one random variable if we know the value of the other (see Tutorial Problem 13). This result, which only holds for linear estimators obtained by minimizing the mse criterion, has led to the widespread use of linear mse estimation in practical applications.

Linear versus affine estimators Because the affine function $\hat{Y} = aX + b$ does not satisfy the principle of superposition when $b \neq 0$, we often use the linear function $\hat{Y}_\ell = hX$. The optimum h for the linear estimator is obtained by minimizing the mse by completing the square as follows

$$J(h) = E[(Y - hX)^2] = E(Y^2) - \frac{E^2(XY)}{E(X^2)} + \frac{[E(XY) - hE(X^2)]^2}{E(X^2)}. \tag{13.65}$$

(a)   (b)
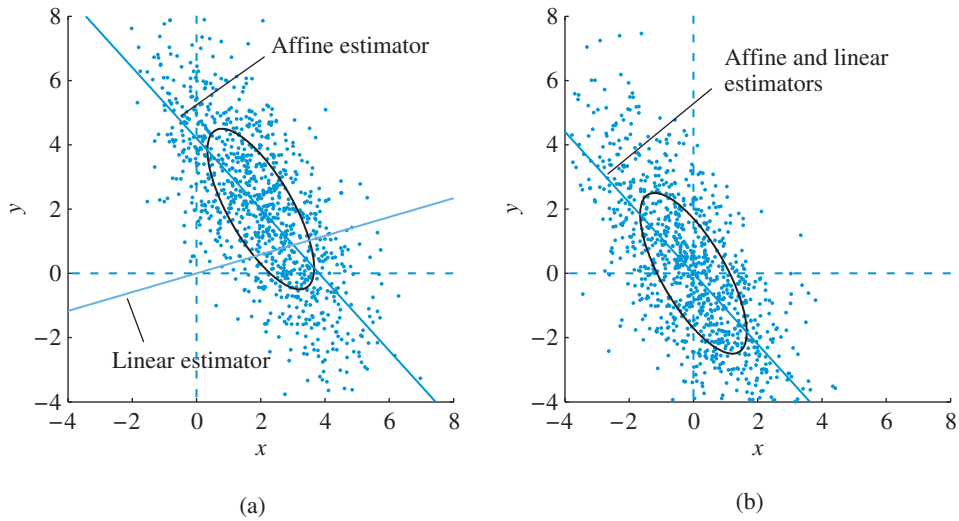
Figure 13.8 (a) The linear estimator is defined by a line that should always pass through the origin. As a result, the unconstrained affine estimator has better performance for random variables with nonzero mean values. (b) The affine and linear estimators are identical for zero-mean random variables.

The optimum value h_o and the minimum mse $J(h_o)$ are given by

$$h_o = \frac{E(XY)}{E(X^2)} \quad \text{and} \quad J(h_o) = E(Y^2) - \frac{E^2(YX)}{E(X^2)}. \tag{13.66}$$

If $m_X = m_Y = 0$, the optimum mse estimator $\hat{Y}_o = a_o X + b_o$ is equivalent to the estimator $\hat{Y}_{o,\ell} = h_o X$ because $a_o = c_{xy}/\sigma_x^2$ and $b_o = 0$. However, this is not the case for $m_X \neq 0$ or $m_Y \neq 0$. For example, if only $m_x = 0$, we have

$$J(h_o) = \sigma_y^2(1 - \rho^2) + m_y^2 = J(a_o, b_o) + m_y^2 > J(a_o, b_o). \tag{13.67}$$

In general, $J(h_o) > J(a_o, b_o)$ for random variables with nonzero mean values. The cause of this performance degradation is that the line defining the linear estimator is constrained to pass *always* through the origin (see Figure 13.8).

We often assume that the involved random variables have zero mean and we restrict interest to linear estimators. However, this is not always possible. For example, the pixels of a digital image always take nonnegative values and hence have nonzero means. In such cases it is preferable to use affine estimators because they give better performance. The common practice in signal processing is to assume zero-mean values and focus on linear estimators; we consider affine estimators only when it is necessary.

Nonzero-mean values and centering Therefore, to attain the lower mse $J(a_o, b_o)$ we could either (a) use the affine estimator $Y = aX + b$ or (b) remove the known means m_X and m_Y from X and Y by

$$\tilde{X} \triangleq X - m_X, \quad \tilde{Y} \triangleq Y - m_Y, \tag{13.68}$$

Table **13.2** Relationship between linear and affine minimum mse estimators. Since $c_{xy} = E(XY) - m_x m_y$, the two estimators are equivalent when $m_x = m_y = 0$.

$\hat{Y}_\ell = hX$ (Linear)	$\hat{Y} = aX + b$ (Affine)
$\hat{Y}_{o,\ell} = \dfrac{E(XY)}{E(X^2)} X$	$(\hat{Y}_o - m_Y) = \dfrac{c_{xy}}{\sigma_x^2}(X - m_X)$
$J(h_o) = E(Y^2) - \dfrac{E^2(YX)}{E(X^2)}$	$J(a_o, b_o) = \sigma_y^2 - \dfrac{c_{xy}^2}{\sigma_x^2}$
Linear $\Rightarrow$ Affine:	$E(XY) \to c_{xy}, E(X^2) \to \sigma_x^2, E(Y^2) \to \sigma_y^2$
	$X \to X - m_x, \hat{Y} \to \hat{Y} - m_y$

and then determine the linear estimator $\tilde{Y} = \tilde{h}\tilde{X}$. The two approaches are equivalent; however, "centering" the random variables around zero simplifies considerably the solution of linear minimum mse estimation problems. To show the equivalence of the two approaches, we first find the linear minimum mse estimator for the zero mean random variables $\tilde{Y}$ and $\tilde{X}$. From (13.66), the linear mse estimator is

$$\hat{\tilde{Y}} = \tilde{h}\tilde{X} = \frac{E(\tilde{Y}\tilde{X})}{E(\tilde{X}^2)}\tilde{X}. \tag{13.69}$$

However, because $E(\tilde{Y}\tilde{X}) = \sigma_{XY}$ and $E(\tilde{X}^2) = \sigma_X^2$, equation (13.69) can be written as

$$(\hat{Y} - m_Y) = \frac{c_{xy}}{\sigma_x^2}(X - m_X), \tag{13.70}$$

which is identical to the affine estimator (13.64). The relationship between linear and affine estimators is clarified in Table 13.2, which shows how to convert from one to the other by simple substitutions.

13.4   Random processes

The defining feature of most real-world signals is that it is impossible to predict with certainty their future values from past values. We may know the range from which a value comes, and that some values within this range are more likely than others, but the exact value will only be known after observation. In view of this inherent unpredictability, we say that the value of a signal at any instant of time is a random variable. Random signal models are based on the notion that the signal that is to be analyzed or processed has been generated by a *random (or stochastic) process*, with a structure that can be characterized and described. In other words, a random process provides a mathematical description of the random nature of the process that generated the observed values of the signal under study. In this section we introduce the concept of a stochastic process, we discuss the statistical description of random processes in the time and frequency domains, and we study how random processes are changed by LTI systems.

Statistical specification of random processes

In order to define a random process, we first recall that a random variable X is a rule for assigning to every outcome ζ of a random experiment a real number $X(\zeta)$. The random variable may take different values if we repeat the experiment, and we do not know in advance which value will occur. Similarly, a random vector assigns a vector to each outcome ζ of the sample space. We can define a random process using a similar approach: to each ζ we assign a time function $X(t, \zeta)$ (*continuous-time stochastic process*) or a sequence $X[n, \zeta]$ (*discrete-time stochastic process*). From the definition, it is clear that a random process is not one function (or sequence), just as a random variable is not one number. A random process is a collection or *ensemble* of functions (or sequences) with "some" probability assigned to each. Thus, each time we perform the random experiment, we observe only one *realization* or *sample function* (sequence) of the process. Thus a stochastic process has a dual interpretation: if we fix the time instant, we obtain a random variable, but if we perform the experiment once, we get a single function or sequence from the ensemble (see Figure 13.9). It is important to draw a distinction between a random process and a set of random variables. A random process takes the notion of time into account, that is, the order in which the observations are made is very important. Therefore, in a random process, samples close together in time behave more similarly than those far apart in time; in general, this is not true for observations from an arbitrary set of random variables.

We shall focus on discrete-time stochastic processes. Formally, a *discrete-time stochastic process* is defined as a sequence of random variables $X[n, \zeta]$. For simplicity we drop the dependence on ζ and we use lower case letters for both a random variable and its value. Thus, the notation $x[n]$ is used to denote the stochastic process $X[n, \zeta]$, a sample sequence
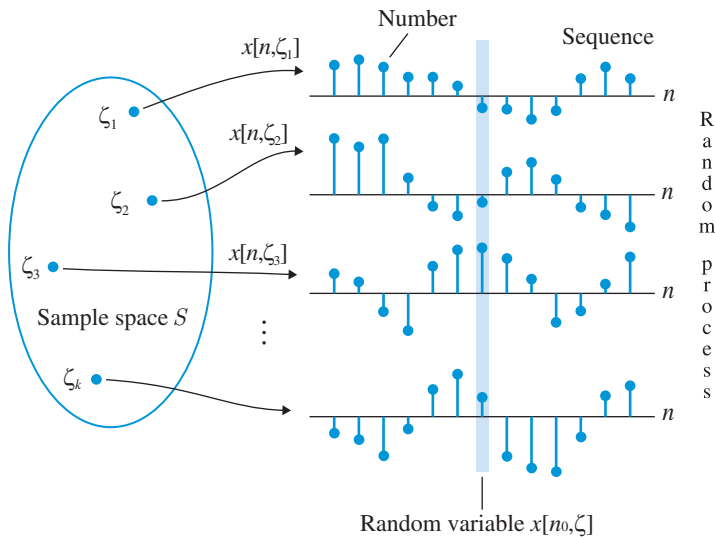
Figure 13.9 The concept of a random (stochastic) process as a mapping from the sample space of a random experiment to an ensemble of sequences.

$x[n]$, or the value of the sequence at time instant n. The exact meaning will be obvious from the context. To illustrate these concepts we consider the following example.

Example 13.1   A simple random process

As a simple and concrete example of a random process, we consider the collection of all sequences $x[n]$ generated by flipping a coin at times $n = 1, 2, \ldots$ and setting $x[n] = 1$ for heads and $x[n] = 0$ for tails. The sequence *HTTHTH* ..., for example, corresponds to $x[n] = \{1\ 0\ 0\ 1\ 0\ 1\ \ldots\}$, which is one realization of the process.

The sample space S for the random process in this example is the collection of all one-sided sequences consisting of 0s and 1s. To each such sequence corresponds a real number ζ between zero and one, which can be expressed in binary notation as

$$\zeta = \sum_{n=1}^{\infty} x[n] 2^{-n} = (0.b_1 b_2 b_3 \ldots)_2, \quad x[n] = 0 \text{ or } 1. \tag{13.71}$$

The converse is also true; to each number ζ, $0 \leq \zeta < 1$, corresponds a one-sided sequence of 1s and 0s.   ■

This example illustrates some fundamental points regarding the nature of stochastic processes:

- First, since the points of any interval of positive length are noncountable, the sample space $S = \{\zeta : 0 \leq \zeta < 1\}$ is an uncountably infinite set. This situation is typical in the study of stochastic processes; we seldom deal with discrete sample spaces except as a convenience. The element of randomness in the considered stochastic process comes from the selection of ζ; given the value of ζ the value of $x[n]$ for each n is uniquely determined by (13.71).
- Second, the ensemble of this process includes some particular sequences with peculiar properties. For example, the sequences $x[n]$ with all ones and all zeros belong to the ensemble, but neither of them can be regarded as random. Both sequences, however, are possible outcomes of the coin-flipping experiment and must be included in the sample space. To understand the implications of such realizations we should distinguish between those events which are impossible and those which merely have zero probability. The latter, according to the frequency interpretation of probability, occur sufficiently seldom so that they can be neglected in a very large number of trials. In theory, we can lump all "nonrepresentative" sample sequences in a set of zero probability.
- The third point, which concerns the relationship between the sample space and the time evolution of a random process is clearly captured in the following quote by Gray and Davisson (2004). "It is important to understand that nature has selected ζ at the beginning of time, but the observer has no way of determining ζ completely without waiting until the end of time. Nature only reveals one bit of ζ per unit time, so the observer can only get an improved estimate of ζ as time goes on. This is an excellent example of how a random process can be modeled by nature selecting only a single elementary outcome, yet the observer sees a process that evolves forever."

A discrete-time stochastic process is completely specified if the joint probability distribution of the set of random variables $x[n_1], x[n_2], \ldots, x[n_p]$ is available for any set of times $n_1, n_2, \ldots, n_p$ (not necessarily consecutive) and any number of points p. Unfortunately, the complete specification of the probability distribution of a stochastic process is usually impossible. Hence, some simplifying assumptions are necessary to obtain more manageable but still useful classes of stochastic process.

13.4.2 Stationary random processes

The most important simplifying assumption is that of stationarity, which requires the process to be in a particular state of "statistical equilibrium." A stochastic process is said to be *strictly stationary* if the sets of random variables $x[n_1], \ldots, x[n_p]$ and $x[n_1+k], \ldots, x[n_p+k]$ have the same joint probability distribution for any set of points, any number p, and any shift k. That is, the joint distribution of any set of random variables from the process, depends only upon their relative positions in the sequence:

- For $p = 1$, this implies that $f(x[n]) = f(x[n + k])$, that is, the marginal probability distribution does not depend on time. Therefore, the mean and variance of $x[n]$ must be constant, that is,

$$\mathrm{E}(x[n]) = m_x \quad \text{and} \quad \mathrm{var}(x[n]) = \sigma_x^2, \text{ for all } n. \tag{13.72}$$

- For $p = 2$, stationarity implies that all bivariate distributions $f(x[n], x[m])$, depend only upon the *lag* (time difference) $\ell \triangleq n - m, n \geq m$. Thus, the covariance of $x[n]$ and $x[m]$ depends on the lag ℓ, that is

$$c_{xx}[n, m] \triangleq \mathrm{cov}(x[n], x[m]) = c_{xx}[\ell]. \text{ for all } m, n \tag{13.73}$$

Thus we note that the stationarity assumption implies that the mean and the variance of the process are constant and that the autocovariance depends only on the lag ℓ. The autocovariance as a function of ℓ is referred to as *autocovariance sequence* (ACVS) of the process $x[n]$. The term "auto" comes about because the two random variables are taken from the same process.

A stochastic process that satisfies (13.72) and (13.73) is called *wide-sense stationary* (WSS) or *second-order stationary*. We note that strict stationarity implies wide-sense stationarity, but not vice versa. We focus on wide-sense stationary processes, which sometimes we simply call stationary.

The *autocorrelation sequence* (ACRS) of a wide-sense stationary process is

$$r_{xx}[m + \ell, m] \triangleq \mathrm{E}(x[m + \ell]x[m]) = r_{xx}[\ell] = c_{xx}[\ell] + m_x^2. \tag{13.74}$$

The ACVS provides a measure of the linear dependence between the values of a random process at two different times. In this sense, it determines how quickly the signal amplitude changes from sample to sample (time variation).

To study the linear dependence between two different random processes $x[n]$ and $y[n]$ we use the cross-covariance and cross-correlation sequences, defined by

$$c_{xy}[n, m] \triangleq \text{cov}(x[n], y[m]), \tag{13.75a}$$

$$r_{xy}[n, m] \triangleq \text{E}(x[n]y[m]) = c_{xy}[n, m] + \text{E}(x[n])\text{E}(y[m]). \tag{13.75b}$$

If $x[n]$ and $y[n]$ are wide-sense stationary and $c_{xy}[n, m]$ depends on the time difference $\ell = n - m$, we say that the two random processes are *jointly wide-sense stationary*. In this case, we have

$$r_{xy}[\ell] = c_{xy}[\ell] + m_x m_y. \tag{13.76}$$

Example 13.2   Sinusoidal random process

We can easily create a sinusoidal random process by making the amplitude, frequency, or phase of a sinusoidal sequence as random variables

$$x[n, \zeta_k] = A(\zeta_k) \cos[\omega(\zeta_k)n + \phi(\zeta_k)]. \tag{13.77}$$

This is a fixed-form random process because each realization has the same shape, which is controlled by a finite number of parameters. Clearly, the properties of the entire process cannot be obtained from a single realization $x[n, \zeta_k]$. Finally, since the values of any realization $x[n, \zeta_k]$ for $n > n_0$ can be determined from the values for $n \leq n_0$ the process is called predictable. The essential randomness in (13.77) is not in the character of individual realizations, but in the particular choice of the set of values (A, ω, ϕ). This process, in general, is not stationary (see Tutorial Problem 9).   ■

Example 13.3   WSS sinusoidal random process

If A and ω are fixed quantities and ϕ is a uniformly distributed random variable, the stochastic process

$$x[n] = A \cos(\omega n + \phi), \quad \phi \sim (0, 2\pi) \tag{13.78}$$

is wide-sense stationary. We first note that the mean value is

$$\text{E}(x[n]) = A\text{E}[\cos(\omega n + \phi)] = \frac{A}{2\pi} \int_0^{2\pi} \cos(\omega n + \phi)\text{d}\phi = 0. \tag{13.79}$$

The autocorrelation of the process is

$$\begin{aligned} r_x[n, m] &= \text{E}[A \cos(\omega n + \phi)A \cos(\omega m + \phi)] \\ &= \frac{A^2}{2\pi} \int_0^{2\pi} \frac{1}{2} \{\cos[\omega(n - m)] + \cos[\omega(n + m) + 2\phi]\} \, \text{d}\phi \\ &= \frac{A^2}{2} \cos \omega(n - m). \end{aligned}$$

Since the mean is constant and the autocorrelation depends on the lag $\ell = n - m$, the process is stationary. Therefore, the ACRS is given by

$$r_x(\ell) = \frac{1}{2}A^2 \cos \omega \ell. \tag{13.80}$$

■

Properties of correlation and covariance sequences From (13.74) and (13.76) we deduce that the correlation and covariance sequences of two jointly wide-sense stationary processes $x[n]$ and $y[n]$ satisfy the same general properties:

1. The ACRS and ACVS have even symmetry, that is,

$$r_{xx}[\ell] = r_{xx}[-\ell], \tag{13.81}$$
$$c_{xx}[\ell] = c_{xx}[-\ell], \tag{13.82}$$

   which follow easily from the definitions (13.73) and (13.74), respectively.

2. The cross-correlation and cross-covariance are *not* even sequences

$$r_{xy}[\ell] = r_{yx}[-\ell], \tag{13.83}$$
$$c_{xy}[\ell] = c_{yx}[-\ell]. \tag{13.84}$$

   Since the subscripts are reversed we stress that $c_{xy}[\ell] \neq c_{xy}[-\ell]$.

3. The cross-correlation and cross-covariance sequences are bounded by

$$r_{xy}^2[\ell] \leq r_{xx}[0]r_{yy}[0], \tag{13.85}$$
$$c_{xy}^2[\ell] \leq c_{xx}[0]c_{yy}[0] = \sigma_x \sigma_y, \tag{13.86}$$

   which follow from (13.29) and (13.34), respectively. In particular,

$$|r_{xx}[\ell]| \leq r_{xx}[0] = \mathrm{E}(x^2[n]), \tag{13.87}$$
$$|c_{xx}[\ell]| \leq c_{xx}[0] = \sigma_x^2. \tag{13.88}$$

   These equations allow the definition of normalized correlation and covariance sequences that are always bounded between -1 and 1.

4. The correlation matrix $\boldsymbol{R}_x$ of the random variables $x[n_1], x[n_2], \ldots, x[n_p]$ is nonnegative definite, that is,

$$\boldsymbol{a}^{\mathrm{T}} \boldsymbol{R}_x \boldsymbol{a} \geq 0 \tag{13.89}$$

   for any $n_1, n_2, \ldots, n_p$, any p, and any $\boldsymbol{a} \neq \boldsymbol{0}$. Thus, only a nonnegative definite sequence can serve as ACRS of a wide-sense stationary process. A similar discussion applies to the ACVS.

5. The correlation matrix of a random vector consisting of p consecutive samples of a stationary process is symmetric and Toeplitz. Indeed, if

$$\boldsymbol{x}[n] = \begin{bmatrix} x[n] & x[n-1] & \cdots & x[n-p+1] \end{bmatrix}^{\mathrm{T}}, \tag{13.90}$$

we have

$$\boldsymbol{R}_x = \begin{bmatrix} r_x[0] & r_x[1] & \cdots & r_x[p-1] \\ r_x[1] & r_x[0] & \cdots & r_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[p-1] & r_x[p-2] & \cdots & r_x[0] \end{bmatrix}, \tag{13.91}$$

because $r_{ij} = \mathrm{E}(x[n-i]x[n-j]) = r_x[j-i]$. A matrix whose elements depend only upon the difference between the column and the row indices is known as a Toeplitz matrix. The covariance matrix of (13.90) is also Toeplitz.

For many random processes, the samples $x[n]$ and $x[m]$ become uncorrelated as the time separation $|n - m|$ between them increases; one notable exception is the harmonic process discussed in Section 13.5.4. If this is true, the covariance sequences decay asymptotically to zero and the correlation sequences to $m_x m_y$ or m_x^2. For zero-mean processes correlation and covariance sequences are identical and may be used interchangeably.

13.4.3   Response of linear time-invariant systems to random processes

We begin by clarifying the meaning of applying a random process as an input to a system. The problem stems from the fact that a random process is not just one sequence but an entire family of sequences indexed by the parameter ζ, a point in the sample space. For any specific ζ, the realization $x[n, \zeta]$ is an ordinary sequence which is a legitimate input for a system. The response of a stable LTI system to the input $x[n, \zeta]$ is another sequence

$$y[n, \zeta] = \sum_{k=-\infty}^{\infty} h[k]x[n-k, \zeta], \tag{13.92}$$

corresponding to the same point ζ in the sample space (see Figure 13.10). In this sense, we can say that the response of an LTI system to a random process $x[n]$ is a jointly distributed random process $y[n]$ defined by

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k], \tag{13.93}$$

where for simplicity we drop the dependence on ζ. This is the simplest way to treat LTI systems with random process inputs. A rigorous treatment using the concept of stochastic convergence is lucidly described by Gray and Davisson (2004).

$x[n,\zeta] \longrightarrow \boxed{h[n]} \longrightarrow y[n,\zeta]$
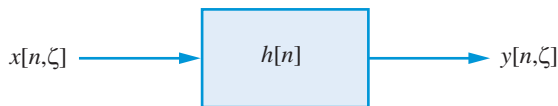
Figure 13.10 An LTI system operates on a single sample sequence of a random process at a time.

Example 13.4   Moving average

Consider the moving average filter

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k]. \tag{13.94}$$

The input $x[n] = s[n] + v[n]$, where $s[n] = 1$ and $v[n]$ is a sequence of zero-mean uncorrelated normal random variables with $\sigma_x^2 = 2$. The output at time n is a random variable $y[n]$ which is a linear combination of M random variables. From the results in Section 13.2.3 we conclude that $y[n]$ is a normal random variable with mean $m_y = m_x$ and variance $\sigma_y^2 = \sigma_x^2/M$. The smaller output deviation means that output values are more likely than input values to be near the mean $m_y = m_x$. The effect of averaging is thus to reduce the size of the fluctuations about the mean value. This is illustrated in Figure 13.11, which shows typical input and output realizations for an $M = 7$ point moving average filter. We note that in order to compute $y[n]$ exactly, we must know $x[n]$ exactly; however, if we are interested only in the mean and variance of $y[n]$, we only need to know the mean and variance of the input. This is a very important point that we should keep in mind throughout the remainder of this chapter. ∎

In general, the inherent unpredictability of a random process implies that although we cannot predict the effects of an LTI system on any specific realization of the input process, we can accurately predict its effect on the average properties. Below, we focus on first-order and second-order properties of wide-sense stationary processes.
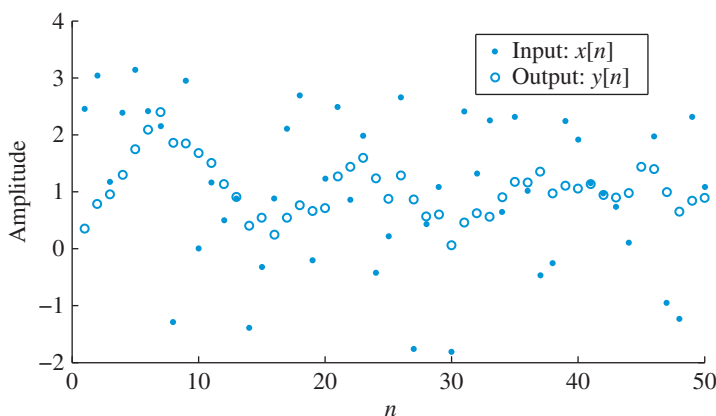
Figure 13.11 Output and purely random input for a moving average filter. The output samples are closer to the mean $m_y = 1$ than the input samples, on the average.

Time-domain analysis The mean of $y[n]$ is found from (13.93) using the linearity property of expectation:

$$E(y[n]) = \sum_{k=-\infty}^{\infty} h[k]E(x[n-k]). \tag{13.95}$$

The sum exists if the system is stable and the input mean is bounded. If $x[n]$ is stationary, we have $E(x[n-k]) = m_x$, and (13.95) yields

$$E(y[n]) = m_x \sum_{k=-\infty}^{\infty} h[k] \triangleq m_y. \tag{13.96}$$

Thus, if the input $x[n]$ is stationary the mean of the output process $y[n]$ is constant.

Next, we consider the ACRS of the output process $y[n]$. We do this in two steps. First, we pre-multiply both sides of (13.93) by $x[m]$ and take the expected value. The result is

$$E(x[m]y[n]) = \sum_{k=-\infty}^{\infty} h[k]E(x[m]x[n-k])$$

$$= \sum_{k=-\infty}^{\infty} h[k]r_{xx}[m, n-k]. \tag{13.97}$$

If the input process is stationary, in the wide sense, then we have

$$r_{xx}[m, n-k] = r_{xx}[(m-n)+k] = r_{xx}[\ell + k], \tag{13.98}$$

where $\ell = m - n$. Thus, the sum in the right hand side of (13.97) and the cross-correlation $E(x[m]y[n])$ become a function of the lag index ℓ. Hence, we obtain

$$r_{xy}[\ell] = \sum_{k=-\infty}^{\infty} h[k]r_{xx}[\ell + k] = \sum_{i=-\infty}^{\infty} h[-i]r_{xx}[\ell - i] = h[-\ell] * r_{xx}[\ell]. \tag{13.99}$$

In a similar way, it is straightforward to show that

$$r_{yx}[\ell] = \sum_{k=-\infty}^{\infty} h[k]r_{xx}[\ell - k] = h[\ell] * r_{xx}[\ell]. \tag{13.100}$$

We next post-multiply both sides of (13.42) by $y[m]$ and take the expectation. The result, which is the autocorrelation sequence of the output process, is

$$E(y[n]y[m]) = \sum_{k=-\infty}^{\infty} h[k]E(x[n-k]y[m])$$

$$= \sum_{k=-\infty}^{\infty} h[k]r_{xy}[(n-m)-k]. \tag{13.101}$$

We note that n and m enter into (13.101) only through their difference $\ell = n - m$; thus, we have

$$r_{yy}[\ell] = \sum_{k=-\infty}^{\infty} h[k]r_{xy}[\ell - k] = h[\ell] * r_{xy}[\ell].$$ (13.102)

Since $E(y[n]) = m_y$ is constant and $E(y[n]y[m])$ depends only on the difference $\ell = n - m$, the process $y[n]$ is wide-sense stationary. Hence *if the input stochastic process to a stable LTI system is wide-sense stationary, the output process is also wide-sense stationary*. If we recall that the deterministic ACRS of $h[n]$ is defined by (see Section 4.5.4)

$$r_{hh}[m] = \sum_{i=-\infty}^{\infty} h[i]h[m + i] = \sum_{k=-\infty}^{\infty} h[-k]h[m - k] = h[-m] * h[m],$$ (13.103)

we can combine (13.102) and (13.99) to obtain the equation

$$r_{yy}[\ell] = \sum_{m=-\infty}^{\infty} r_{hh}[m]r_{xx}[\ell - m] = r_{hh}[\ell] * r_{xx}[\ell],$$ (13.104)

which shows that the ACRS of the output of an LTI system is the convolution of the ACRS of the the input process with the ACRS of the system impulse response. This is illustrated in Figure 13.12.

We can easily show that relations identical to (13.99), (13.100), (13.102), and (13.104) hold for the ACVS of a wide-sense stationary process; just replace r by c everywhere, except for r_{hh} (see Tutorial Problem 12). We stress once again that it is the covariance that measures the "linear association" between random variables, not the correlation. Finding the distribution of the output process of an LTI system is extremely difficult, except in special cases. Thus, if $x[n]$ is a normal process, then $y[n]$ is a normal process with mean and ACRS given by (13.96) and (13.104) (see relationship (13.48)).
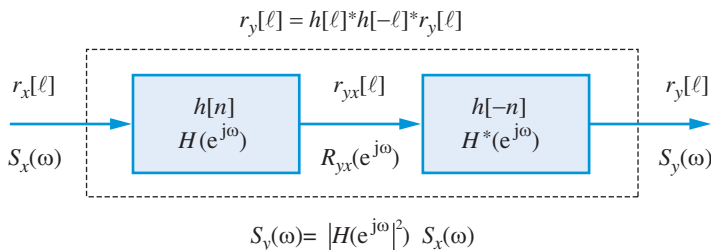
Figure 13.12 The ACRS and the PSD of the output process can be thought of as "filtered" by an LTI system with impulse response $r_{hh}[n] = h[n] * h[-n]$. Therefore, although LTI systems process individual sequences, they have the same effect on all sequences with the same mean and ACRS.

Frequency-domain analysis The convolution equations (13.104) and (13.103) can be greatly simplified by using transform techniques. Taking the DTFT of (13.104) and using the convolution theorem (4.153), we obtain

$$R_{yy}(e^{j\omega}) = R_{hh}(e^{j\omega})R_{xx}(e^{j\omega}), \tag{13.105}$$

where $R_{xx}(e^{j\omega})$, $R_{hh}(e^{j\omega})$, and $R_{yy}(e^{j\omega})$ are the Fourier transforms of $r_{xx}[\ell]$, $r_{hh}[\ell]$, and $r_{yy}[\ell]$, respectively. Also, from (13.103) we have

$$R_{hh}(e^{j\omega}) = H(e^{-j\omega})H(e^{j\omega}) = H^*(e^{j\omega})H(e^{j\omega}) = \left|H(e^{j\omega})\right|^2. \tag{13.106}$$

Hence,

$$R_{yy}(e^{j\omega}) = \left|H(e^{j\omega})\right|^2 R_{xx}(e^{j\omega}). \tag{13.107}$$

The frequency-domain version of (13.96) can be easily shown to be

$$m_y = H(e^{j0})m_x. \tag{13.108}$$

Equation (13.107), like the equivalent relation (13.104), *does not* contain information about the phase response $\angle H(e^{j\omega})$ of the system. A relation that includes *both* amplitude and phase response information is obtained by transforming (13.100) in the frequency domain. The result is

$$R_{yx}(e^{j\omega}) = H(e^{j\omega})R_{xx}(e^{j\omega}), \tag{13.109}$$

where $R_{yx}(e^{j\omega})$ is the Fourier transform of the cross-correlation sequence $r_{xy}[\ell]$.

In the z-domain, equations (13.109) and (13.107) take the form

$$R_{yx}(z) = H(z)R_{xx}(z), \tag{13.110}$$

$$R_{yy}(z) = H(z)H(1/z)R_{xx}(z), \tag{13.111}$$

where we assume that all ROCs include the unit circle $z = e^{j\omega}$. Similar equations can be obtained for covariance sequences by simply replacing R by C. Formal derivations are obtained in Tutorial Problem 14.

13.4.4   Power spectral densities

The Fourier transform $R_{xx}(e^{j\omega})$ of the ACRS $r_{xx}[\ell]$ of a wide-sense stationary process $x[n]$ is called the *power spectral density* (PSD) of the process. For notational convenience, we define the PSD by

$$S_{xx}(\omega) \triangleq \sum_{\ell=-\infty}^{\infty} r_{xx}[\ell]e^{-j\ell\omega}. \tag{13.112}$$

Since the ACRS and the PSD form a DTFT pair, we have

$$r_{xx}[\ell] = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{xx}(\omega)e^{j\omega\ell}d\omega. \tag{13.113}$$

The PSD (13.112) exists if the ACRS is absolutely summable, that is, $\sum_\ell |r_{xx}[\ell]| < \infty$.
We can now express (13.107) in terms of PSD functions as follows:

$$S_{yy}(\omega) = |H(e^{j\omega})|^2 S_{xx}(\omega). \tag{13.114}$$

Thus we see that the output PSD equals the input PSD multiplied by the squared magnitude of the system frequency response function (see Figure 13.12).

Suppose now that we wish to determine the *average power* $E(y^2[n])$ at the output of a stable LTI system. Setting $\ell = 0$ in (13.113) yields

$$E(y^2[n]) = r_{xx}(0) = \frac{1}{2\pi} \int_{-\pi}^{\pi} S_{yy}(\omega) d\omega, \tag{13.115}$$

that is, the area under the PSD function (divided by 2π) is equal to the average power of the random process. Using (13.114), we can write (13.115) as

$$E(y^2[n]) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 S_{xx}(\omega) d\omega. \tag{13.116}$$

To understand the physical meaning of the PSD we consider the narrow band filter ($\Delta\omega \ll \omega_c$) shown in Figure 13.13. The average power at the output of the filter is

$$E(y^2[n]) = 2\frac{1}{2\pi} \int_{\omega_c - \Delta\omega/2}^{\omega_c + \Delta\omega/2} S_{xx}(\omega_c) d\omega \approx (1/\pi) S_{xx}(\omega_c) \Delta\omega. \tag{13.117}$$

Thus to find the average power contained in any frequency band we integrate the PSD over the frequency band. Because the average power must be nonnegative for any choice of ω_c and $\Delta\omega$, it follows that any PSD must satisfy the condition

$$S_{xx}(\omega) \geq 0. \quad \text{for all } \omega \tag{13.118}$$
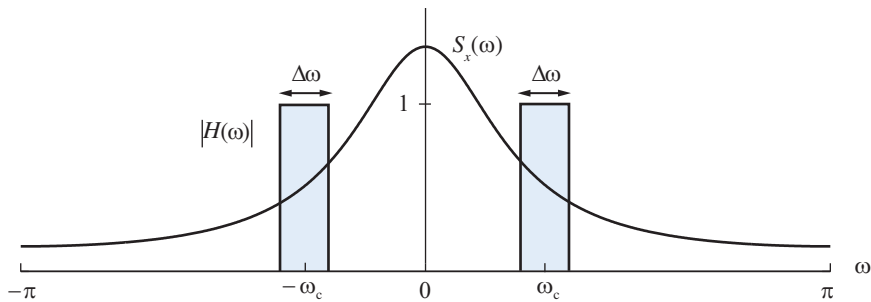
Figure 13.13 Physical interpretation of power spectrum density as power at the output of a narrowband LTI system.

This is the frequency domain equivalent of (13.89); they both stem from the fact that $E(y^2[n]) \geq 0$. However, it is easier to check whether the PSD is nonnegative than to check whether the ACRS is nonnegative definite.

Since the ACRS has even symmetry, $r_x[\ell] = r_x[-\ell]$, we can easily show that

$$S_{xx}(\omega) = r_{xx}[0] + 2 \sum_{\ell=1}^{\infty} r_{xx}[\ell] \cos \omega \ell, \qquad (13.119)$$

which implies that the PSD is a real-valued function with even symmetry, that is

$$S_{xx}(-\omega) = S_{xx}(\omega). \quad \text{(real and even)} \qquad (13.120)$$

The PSD function, like the ACRS, does not contain any phase information. Since the ACRS and the PSD form a DTFT pair they contain the same information but in a different form. Some information is better revealed in one domain than the other.

Thus, the value of the nonnegative PSD function at a given frequency shows the average power at a narrow spectral band centered at this frequency. In this sense, the shape of the PSD curve shows how the average power of a random process is distributed at the various frequencies. The total average power is obtained by integrating the PSD over any 2π interval.

Example 13.5   Calculation of PSD

Determine the PSD of a stationary process $x[n]$ with zero mean value and ACRS $r_{xx}[\ell] = a^{|\ell|}$, $-1 < a < 1$.

The condition $|a| < 1$ implies that the ACRS is absolutely summable:

$$\sum_{\ell=-\infty}^{\infty} |r_x[\ell]| = \sum_{\ell=-\infty}^{\infty} |a|^{|\ell|} = 1 + 2 \sum_{\ell=1}^{\infty} |a|^{\ell} = 1 + \frac{2}{1 - |a|} < \infty. \qquad (13.121)$$

Therefore, the PSD exists and it is given by

$$S_{xx}(\omega) = \sum_{\ell=-\infty}^{\infty} a^{|\ell|} e^{-j\omega\ell} = \sum_{\ell=0}^{\infty} a^{\ell} e^{-j\omega\ell} + \sum_{\ell=0}^{\infty} a^{\ell} e^{j\omega\ell} - 1$$

$$= \frac{1}{1 - ae^{-j\omega}} + \frac{1}{1 - ae^{j\omega}} - 1 = \frac{1 - a^2}{1 + a^2 - 2a \cos \omega}. \qquad (13.122)$$

We note that (13.122) satisfies the properties $S_{xx}(\omega) \geq 0$ and $S_{xx}(-\omega) = S_{xx}(\omega)$. ■

Cross power spectral densities The cross-correlation sequence is a natural tool for examining the relationship between two random processes in the time-domain. A complementary tool for the frequency domain is the *cross power spectral density* (CPSD) defined by

$$S_{yx}(\omega) \triangleq \sum_{\ell=-\infty}^{\infty} r_{yx}[\ell]e^{-j\ell\omega}. \tag{13.123}$$

We note that $S_{yx}(\omega)$ is a complex function because $r_{yx}[\ell]$ is not an even function. Using (13.123) we can write (13.109) as

$$S_{yx}(\omega) = H(e^{j\omega})S_{xx}(\omega). \tag{13.124}$$

If $S_{xx}(\omega) = \sigma_x^2$ for all ω, we have $S_{yx}(\omega) = \sigma_x^2 H(e^{j\omega})$. This idea can be used to develop techniques for estimation of the frequency response function of LTI systems in practical applications (see Tutorial Problem 15).

Important remark From the relation $r_{xx}[\ell] = c_{xx}[\ell] + m_x^2$, it follows that

$$S_{xx}(\omega) = C_{xx}(e^{j\omega}) + 2\pi m_x^2 \delta(\omega). \tag{13.125}$$

Therefore, if $m_x \neq 0$, then $S_{xx}(\omega)$ contains an impulse at $\omega = 0$. To avoid the problems caused by this impulse in spectral estimation (see Section 14.2) we consider processes with zero mean value. In the time-series literature, this problem is avoided by defining the PSD function as $S_{xx}(\omega) = C_{xx}(e^{j\omega})$. Then the PSD function shows how the variance of a wide-sense stationary process is distributed in the frequency domain.

Second-order moments A large number of problems can be solved using the mean value and ACRS or ACVS of a stationary process (second-order moments). The ACVS does not determine the waveform of the realizations. This is not surprising because the covariance of two random variables does not determine their probability distribution. Intuitively, the ACVS shows how quickly the realizations change from sample to sample.

13.5   Some useful random process models

We next describe several types of random process which provide useful models for signals encountered in practical signal processing applications.

13.5.1   White noise process

The simplest random process, which is used as a building block to create more complicated processes, is a sequence of uncorrelated random variables $x[n]$ with zero mean and constant variance σ_x^2. The ACVS and the PSD are given by

$$c_{xx}[\ell] = \sigma_x^2 \delta[\ell] \xrightarrow{\text{DTFT}} S_{xx}(\omega) = \sigma_x^2. \text{ for all } \omega \tag{13.126}$$

The fact that $S_{xx}(\omega)$ takes a constant value means that each frequency component contributes exactly the same amount to the total variance or power of the process. This property

is analogous to that of white light, where all frequencies (colors) are present in the same amount. For this reason, the purely random process specified by (13.126) is often referred to as "white noise" and it is denoted by

$$x[n] \sim \text{WN}(0, \sigma_x^2). \tag{13.127}$$

If the random variables $x[n]$ are normally distributed, that is, $x[n] \sim N(0, \sigma_x^2)$, the result is a white Gaussian noise process denoted by $x[n] \sim \text{WGN}(0, \sigma_x^2)$.

13.5.2 Linear processes

Suppose that the input to a stable LTI system is the white noise process (13.126). Using (13.104), (13.114), and (13.126) we can easily show that the ACRS and PSD of the resulting *linear process* are given by

$$r_{yy}[\ell] = \sigma_x^2 r_{hh}[\ell] \xrightarrow{\text{DTFT}} S_{yy}(\omega) = \sigma_x^2 |H(e^{j\omega})|^2. \tag{13.128}$$

We can use this approach to generate any process with a continuous PSD function.

If the system $h[n]$ is minimum-phase (that is, causal and causally invertible), the resulting linear process $y[n]$ is known as a *regular* process. The system specified by

$$y[n] = \sum_{k=0}^{\infty} h[k]x[n - k] \tag{13.129}$$

is known as a *synthesis* or *coloring* filter. Since both the system $h[n]$ and its inverse $h_\text{I}[n] = \mathcal{Z}^{-1}[1/H(z)]$ are causal and stable, we can recover the input white noise using the system

$$x[n] = \sum_{k=0}^{\infty} h_\text{I}[k]y[n - k], \tag{13.130}$$

which is known as an *analysis* or *whitening* filter. The processes $y[n]$ and $x[n]$ are linearly equivalent in the sense that each is linearly dependent on the other and its past. Finding the synthesis filter $h[n]$ from the process ACRS or PSD is known as *spectral factorization*.

A process is regular if its PSD satisfies the Paley–Wiener condition

$$\int_{-\pi}^{\pi} |\ln S_{yy}(\omega)| d\omega < \infty. \tag{13.131}$$

This condition is *not* satisfied if the PSD consists of lines or it is bandlimited. A complete treatment can be found in Papoulis and Pillai (2002).

13.5.3 Autoregressive moving average (ARMA) processes

The regular process model (13.129) is specified by an infinite number of parameters, that is, the values of the sequences $h[n], 0 \leq n \leq \infty$ or $r_{yy}[\ell], 0 \leq \ell \leq \infty$. A finite parameter

model can be obtained by considering regular processes generated by minimum-phase pole-zero systems. The result is an ARMA(p, q) process defined by the difference equation

$$y[n] = -\sum_{k=1}^{p} a_k y[n-k] + \sum_{k=0}^{q} b_k x[n-k], \qquad (13.132)$$

where $x[n] \sim \text{WN}(0, \sigma_x^2)$. The process $y[n]$ has zero mean, thus $r_{yy}[\ell] = c_{yy}[\ell]$. The PSD of an ARMA$(p, q)$ process is given by

$$S_{yy}(\omega) = \sigma_x^2 \left| H(e^{j\omega}) \right|^2 = \sigma_x^2 \left| \frac{\sum_{k=0}^{q} b_k e^{-j\omega k}}{1 + \sum_{k=1}^{p} a_k e^{-j\omega k}} \right|^2. \qquad (13.133)$$

For $q = 0$, we have an all-pole system which generates an *autoregressive process* of order p, denoted by AR(p). However, if $p = 0$, we have an all-zero system which generates a *moving average process* of order q, denoted by MA(q). For $q > 0$ and $p > 0$ we have a pole-zero system producing an ARMA(p, q) process.

A task of major practical interest is how to determine the coefficients $\{a_k, b_k\}$ of the ARMA model from the ACRS $r_{yy}[\ell]$. To find the relation between these quantities, we first multiply both sides of (13.132) by $y[n - \ell]$ and then we take the expectation. This yields

$$\text{E}(y[n]y[n-\ell]) = -\sum_{k=1}^{p} a_k \text{E}(y[n-k]y[n-\ell]) + \sum_{k=0}^{q} b_k \text{E}(x[n-k]y[n-\ell]),$$

or

$$r_{yy}[\ell] = -\sum_{k=1}^{p} a_k r_{yy}[\ell-k] + \sum_{k=0}^{q} b_k r_{xy}[\ell-k]. \qquad (13.134)$$

Pre-multiplying both sides of (13.129) by $x[n+\ell]$ and taking the expected value, we obtain

$$\text{E}(x[n+\ell]y[n]) = \sum_{m=0}^{\infty} h[m]\text{E}(x[n+\ell]x[n-m]). \qquad (13.135)$$

Using (13.126), (13.135) results in

$$r_{xy}[\ell] = \sigma_x^2 \sum_{m=0}^{\infty} h[m]\delta[\ell+m] = \sigma_x^2 h[-\ell]. \qquad (13.136)$$

Substituting (13.136) into (13.134) we finally obtain

$$r_{yy}[\ell] = -\sum_{k=1}^{p} a_k r_{yy}[\ell-k] + \sigma_x^2 \sum_{k=0}^{q} b_k h[k-\ell], \quad \text{all } \ell. \qquad (13.137)$$

Since $h[n]$ is a function of $\{a_k, b_k\}$, the ACRS $r_{yy}[\ell]$ is a nonlinear function of the ARMA model coefficients. This relation becomes linear in the case of AR processes; for this reason AR models are widely used in practical signal processing applications.

AR processes Indeed, if the process is AR(p), setting $q = 0$ in (13.137) yields

$$r_{yy}[\ell] = -\sum_{k=1}^{p} a_k r_{yy}[\ell - k] + \sigma_x^2 b_0 h[-\ell]. \ \ \text{all } \ell \tag{13.138}$$

The impulse response of an all-pole system satisfies the difference equation

$$h[n] = -\sum_{k=1}^{p} a_k h[n - k] + b_0 \delta[n], \ n \geq 0 \tag{13.139}$$

and $h[n] = 0$ for $n < 0$. From (13.139) we obtain $h[0] = b_0$ and without loss of generality we choose $b_0 = 1$. Equation (13.138) is true for all ℓ, but because $h[\ell] = 0$ for $\ell < 0$, $h[-\ell] = 0$ for $\ell > 0$, and we have

$$r_{yy}[\ell] = -\sum_{k=1}^{p} a_k r_{yy}[\ell - k], \ \ell > 0 \tag{13.140}$$

which is a recursive relation for $r_{yy}[\ell]$ in terms of past values and the filter coefficients a_k. Setting $\ell = 0$ and using $r_{yy}[-\ell] = r_{yy}[\ell]$, (13.138) yields

$$\sigma_x^2 = r_{yy}[0] + \sum_{k=1}^{p} a_k r_{yy}[k]. \tag{13.141}$$

Equations (13.140) for $\ell = 1, 2, \ldots, p$ comprise p equations that relate the p coefficients $a_1, a_2, \ldots, a_p$ to the first $p + 1$ ACRS coefficients $r_{yy}[0], r_{yy}[1], \ldots, r_{yy}[p]$. These p equations can be written in matrix form as

$$\begin{bmatrix} r_{yy}[0] & r_{yy}[1] & \cdots & r_{yy}[p-1] \\ r_{yy}[1] & r_{yy}[0] & \cdots & r_{yy}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{yy}[p-1] & r_{yy}[p-2] & \cdots & r_{yy}[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} r_{yy}[1] \\ r_{yy}[2] \\ \vdots \\ r_{yy}[p] \end{bmatrix}, \tag{13.142}$$

or more compactly as

$$R_y a = -r_y. \tag{13.143}$$

These equations are known as the *Yule–Walker equations* in the statistics literature. Because of the Toeplitz structure and the nature of the right hand side, the linear system (13.143) can be solved recursively by using the algorithm of Levinson–Durbin see Section 14.4.2. Having obtained a, we can compute the input noise variance using (13.141) as follows:

$$\sigma_x^2 = \sigma_y^2 + a^T r_y = \sigma_y^2 - r_y^T R_y^{-1} r_y \leq \sigma_y^2. \tag{13.144}$$

The last inequality results from the fact that the matrix R_y is nonnegative definite. If the Toeplitz matrix R_y is positive definite, the AR(p) model obtained from the solution of

(13.143) is minimum phase. The Levinson–Durbin algorithm, besides its computational efficiency, provides a powerful theoretical tool for analyzing the properties of AR process models and the development of lattice filter structures for the implementation of the analysis and synthesis filters. However, it is adequate for our purposes to solve (13.143) using the MATLAB operator `a = -Ry\ry` and then (13.144) is computed by `s2x = s2y+a'*ry`.

Example 13.6 AR(2) process

The difference equation for the AR(2) model is

$$y[n] = -a_1 y[n-1] - a_2 y[n-2] + x[n]. \tag{13.145}$$

The system function can be written as

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{z^2}{(z-p_1)(z-p_2)}. \tag{13.146}$$

The most interesting case occurs when the poles are complex conjugate, that is, $p_{1,2} = r\exp(\pm j\theta)$. If the poles are inside the unit circle, that is, $0 < r < 1$, the output process is stationary and regular. The Yule–Walker equations are

$$r_{yy}[0]a_1 + r_{yy}[1]a_2 = -r_{yy}[1], \tag{13.147}$$

$$r_{yy}[1]a_1 + r_{yy}[0]a_2 = -r_{yy}[2]. \tag{13.148}$$

Solving for a_1 and a_2, we get

$$a_1 = -\frac{r_{yy}[1](r_{yy}[0] - r_{yy}[2])}{r_{yy}^2[0] - r_{yy}^2[1]}, \quad a_2 = -\frac{r_{yy}[0]r_{yy}[2] - r_{yy}^2[1]}{r_{yy}^2[0] - r_{yy}^2[1]}. \tag{13.149}$$

The variance of the input white noise is determined by

$$\sigma_x^2 = r_{yy}[0] + a_1 r_{yy}[1] + a_2 r_{yy}[2]. \tag{13.150}$$

Equations (13.147), (13.148), and (13.150) can be used to determine $r_{yy}[0]$, $r_{yy}[1]$, and $r_{yy}[2]$ from a_1, a_2, and σ_x^2. The values of $h[n]$ for $n \geq 0$ and $r_{yy}[\ell]$ for $\ell > 2$ can be computed recursively using the formulas (13.139) and (13.140), respectively (see Tutorial Problem 16).

This behavior of the AR(2) model is illustrated in Figure 13.14 with $a_1 = -0.4944$ and $a_2 = 0.64$. The model has two complex conjugate poles with $r = 0.8$ and $\theta = \pm 2\pi/5$. The PSD has a single peak and displays a passband type of behavior. The impulse response is a damped sine wave while the autocorrelation is a damped cosine. The typical realization of the output shows clearly a pseudoperiodic behavior that is explained by the shape of the autocorrelation and the spectrum of the model. We also note that if the poles are complex conjugates, the autocorrelation has pseudoperiodic behavior. ∎
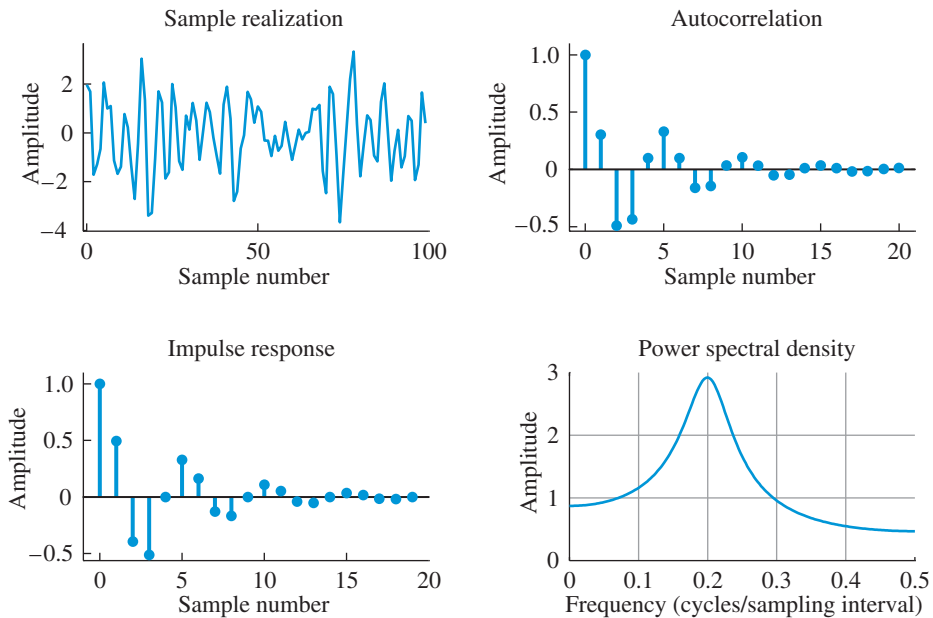
Figure 13.14 Sample realization, impulse response, ACRS, and PSD of an AR(2) random process with complex conjugate poles at $p_{1,2} = 0.8 \exp(\pm j2\pi/5)$.

13.5.4   Harmonic process models

A *harmonic process* is defined by the formula

$$x[n] = \sum_{k=1}^{p} A_k \cos(\omega_k n + \phi_k), \; \omega_k \neq 0 \tag{13.151}$$

where p, $A_1, \ldots, A_p$, and $\omega_1, \ldots, \omega_p$ are constants and $\phi_1, \ldots, \phi_p$ are pairwise independent random variables uniformly distributed in the interval $(0, 2\pi)$. It can be shown (see Tutorial Problem 17) that $x[n]$ is wide-sense stationary with mean value zero and ACRS given by

$$r_{xx}[\ell] = c_{xx}[\ell] = \frac{1}{2} \sum_{k=1}^{p} A_k^2 \cos \omega_k \ell. \tag{13.152}$$

We note that $r_{xx}[\ell]$ consists of a sum of "in-phase" cosines with the same frequencies as in the process $x[n]$. If $\omega_k/(2\pi)$ are rational numbers, the ACRS $r_{xx}[\ell]$ is periodic and can be expanded as a Fourier series. The coefficients of this series provide the PSD of $x[n]$. However, because $r_{xx}[\ell]$ is a linear superposition of cosines, it always has a line spectrum with $2p$ lines of strength $A_k^2/4$ at frequencies $\pm\omega_k$. In terms of the impulse function $\delta(\omega)$ we have

$$S_x(\omega) = \sum_{k=1}^{p} 2\pi \frac{A_k^2}{4} [\delta(\omega + \omega_k) + \delta(\omega - \omega_k)]. \tag{13.153}$$

The term 2π is absent if we use the frequency variable $f = \omega/2\pi$. If $r_{xx}[\ell]$ is periodic, then the lines are equidistant (that is, harmonically related), hence the name *harmonic process*.

13.5.5   The Wold decomposition theorem

The Wold decomposition theorem, which is mainly of theoretical interest, essentially says that any stationary discrete-time random process can be expressed as the sum of two uncorrelated processes, one regular process and one harmonic process. To obtain a heuristic interpretation we note that the most general form of PSD includes a continuous component (due to the regular process) and a discrete (or line) component (due to the harmonic process).

Learning summary

- Random signals are described mathematically using probabilistic models. For each probability model there is an underlying random experiment consisting of (a) the set of all possible outcomes (sample space S), (b) a collection of outcomes (events), and (c) a real number assigned to each event (probability) according to well-defined rules.

- Random variables, that is, mappings from the sample space S to real numbers, allow real-world events like random signals to be analyzed using probability theory. A random variable is completely characterized by probability functions, pdf or CDF. Both functions allow computations of probabilities of events (or intervals) on the real line. Two most important random variable models are the uniform, $U(a, b)$, and the normal, $N(m, \sigma^2)$ distributions.

- Moments like mean and variance are characteristic numbers which allow description of random variables when distribution functions are unavailable. These moments are computed using mathematical expectation $E[\cdot]$ operator. The mean value m provides a measure of the center of the distribution while square-root of variance σ^2 (or standard deviation, σ) provides a measure of the spread of random values from the mean.

- Relationships between more than one random situation are modeled by two jointly distributed random variables. In theory these can be described by their joint distributions but in practice we use their means, variances, and pairwise interactions called covariances (or correlations) which are computed using joint expectation. The covariances measure the extent of linear relationship between two random variables.

- Two random variables are statistically independent if their joint pdf factors into a product of their densities, $f(x, y) = f(x)f(y)$; they are uncorrelated (no linear dependence) if their covariance is zero, and orthogonal if their correlation is zero.

- Real-world signals which have random values are modeled using the concept of random processes. This concept can be thought of as a collection of random variables for each sample index or an ensemble of signals with an associated distribution. In practice, random processes are characterized by their mean sequences and autocovariance (or autocorrelation) sequences.

- Random processes that have time-invariant statistics of all order are called strict-sense stationary processes and are useful in practice. We focus on wide-sense stationary processes which are stationary in their mean and autocovariance (or autocorrelation). The resulting mean sequence is a constant while the autocovariance sequence, ACVS (or autocorrelation, ACRS) is a function of lag ℓ.

- A random process $x[n]$, when processed using an LTI system with impulse response $h[n]$ results in an output process $y[n]$. When $x[n]$ is stationary, $y[n]$ is also stationary with mean $m_y = m_x(\sum h[n])$ and the autocorrelation $r_{yy}[\ell] = h[\ell] * h[-\ell] * r_{xx}[\ell]$. The cross-correlation between output and input is $r_{yx}[\ell] = h[\ell] * r_{xx}[\ell]$.

- The Fourier transform of the ACRS is called power spectral density (PSD), $S_{xx}(\omega)$. It is a real and nonnegative function of ω and its integral over a band describes average power in the signal over that band of frequencies.

- A white noise process is a stationary process with zero mean and a constant variance or PSD. It is a useful abstraction and is used to generate other linear processes by filtering it through LTI systems. When the LTI system is described by a difference equation, the resulting output process is called an ARMA(p, q) process which has a finite number of parameters. Special cases of ARMA processes are AR(p) and MA(q) processes. When the PSD contains only impulses, the resulting process is called a harmonic process.

TERMS AND CONCEPTS

Autocorrelation sequence (ACRS)
Correlation between the samples of a stationary process as a function of lag ℓ is termed the autocorrelation sequence.

Autocovariance sequence (ACRS) Covariance between the samples of a stationary process as a function of lag ℓ is termed the autocovariance sequence.

Autoregressive moving average (ARMA) process A linear process resulting from a linear constant coefficient difference equation driven by a white noise process and denoted by ARMA(p, q).

Autoregressive (AR) process An ARMA process with $q = 0$, that is the generating difference equation has only the autoregressive part, and is denoted by AR(p).

Continuous random variable A random variable that takes as a value any real number from a specified set.

Correlation A measure of the affine relationship between two random variables, denoted by r_{xy}.

Correlation coefficient A normalized covariance between two random variables. It is denoted by ρ_{xy} and satisfies $-1 \le \rho_{xy} \le 1$.

Covariance A measure of the linear relationship between two random variables, denoted by $\text{cov}(x, y)$.

Covariance matrix Covariances between components of a random vector, organized as a square matrix.

Cumulative distribution function (CDF) A probability function $F(a)$ that describes the accumulated nature of probability at a, that is of a semi-infinite interval from $-\infty$ to a.

Discrete random variable A random variable that takes values from a finite or countably infinite set of values.

Events A subset of outcomes for which a probability is desired. An outcome may or may not be an event.

Expectation A statistical average of a function of a random variable with respect to its pdf.

Harmonic process A random process whose PSD contains impulses but no smooth spectrum. It is generated using appropriately modeled sinusoidal sequences.

Joint probability density function A 2D function that describes the distributed nature of probability for two joint random variables, denoted by $f(x, y)$.

Linear process A stationary random process generated by an LTI system driven by a white noise process. Its PSD is a continuous function.

Mean value A measure of the center or location of a probability distribution and

indicates the average value of a random variable.

Mean sequence A sequence formed by the mean values of the samples of a random process. For a stationary process, mean sequence is a constant.

Moving average process An ARMA process with $p = 0$, that is the generating difference equation has only the moving average part, and is denoted by MA(q).

Nonnegative definite matrix A matrix whose quadratic product is nonnegative. Autocorrelation and autocovariance matrices are nonnegative definite.

Normal distribution A bell shaped density function with parameters mean m and variance σ^2 described by the Gaussian function and denoted by $N(m, \sigma^2)$.

Normal random vector A random vector whose components are jointly normal with each other.

Outcome The result of performing a given random experiment.

Orthogonal random variables Two random variables with zero correlation between them. Describes perpendicular relationship between random variable as vectors.

Power spectral density (PSD) A nonnegative function of ω that is a Fourier transform of the ACRS. It gives an average power in the random sequence over a frequency band.

Probability A numerical value between 0 and 1, inclusive of both, assigned to an event, that describes the randomness of that event.

Probability density function (pdf) A probability function $f(x)$ that describes the distributed nature of probability over an interval, i.e. $\Pr[a_1 < x < a_2] = \int_{a_1}^{a_2} f(x)\,dx$.

Random experiment An experiment whose outcome cannot be predicted. It is used to describe events of a probabilistic nature.

Random variable A function from the sample space S to the real line so that all outcomes and events are real numbers.

Random process A model that describes signals which cannot be described mathematically and which exhibits random values each time. It can be thought of as a collection of functions (or sequences) with probabilistic attributes or as a temporal collection of random variables.

Regular process A linear process generated by a minimum-phase (that is, causal and causally invertible) LTI system driven by a white noise process.

Relative frequency A ratio of the number of outcomes favorable to an event to the total number of outcomes in the sample space.

Sample space Collection of all possible outcomes in a random experiment, denoted by S.

Spectral factorization A decomposition of the ACRS or PSD into a minimum-phase component to design a coloring filter.

Standard deviation A measure of spread, in the same units as the original observation, of a distribution about its mean value. It is the positive square root of the variance.

Strict-sense stationary process A process whose all-order joint density functions (or all-order statistics) are time-invariant.

Statistical independence Random variable x does not statistically affect random variable y in any way, that is $f(y|x) = f(y)$. It also implies that $f(x, y) = f(x)f(y)$.

Uncorrelated random variables Two random variables with zero covariance between them. Describes "linear independence."

Uniform distribution A density function that is constant over a finite interval, $[a, b]$, and is denoted by $U(a, b)$.

Variance A measure of spread or dispersion of a distribution about its mean value.

White noise process A sequence of uncorrelated random variables. Its PSD takes a constant value for all frequencies.

Wide-sense stationary process A process whose statistics are time-invariant only up to the second-order.

MATLAB functions and scripts

Name	Description	Page
epdf*	Computes the empirical probability density function	781
rand	Generates uniform random numbers	784
randn	Generates normal random numbers	785

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A simple introduction to probability and random variables, at the same level as in this book, is given in Ross (2004) and Hogg and Tanis (2005).
2. The standard references to probability theory and stochastic processes for electrical engineers are Papoulis and Pillai (2002), Stark and Woods (2002), and Leon-Garcia (2008); the last reference is closer to the level of this book.
3. A lucid theoretical introduction to the basic principles of probability, random variables, and random processes is provided by Davenport (1970) and Gray and Davisson (2004). The classic textbook by Davenport and Root (1987), published first in 1958, shows that the fundamental principles of random signal processing are still the same; only the emphasis from continuous-time to discrete-time has changed.
4. A concise yet thorough discussion of random variables and random sequences is given in Manolakis *et al.* (2005) with many signal processing examples. This reference provides a detailed discussion of the theory of ARMA models and their applications. A more theoretical discussion of ARMA processes is given by Box *et al.* (2008) and Stoica and Moses (2005).

Review questions

1. What is a random experiment and what are its components?
2. Describe probabilistic models that explain events of random behavior.
3. What is statistical regularity and why is it essential for the foundation of the mathematical theory of probability?
4. What is probability and how is it defined in various models?
5. Describe the concept of a random variable and explain why it is needed.
6. What are probability functions and how do they differ from probability?
7. Describe different types of random variable and their associated probability functions.
8. How is cumulative distribution function related to the probability density function and vice versa?
9. What are the basic properties of the cumulative distribution function?
10. What are the basic properties of the probability density function?
11. Why are statistical averages like mean and variance preferred in practice to probability functions?

12. What is mathematical expectation and how is it used to compute mean and variance of a random variable?

13. What quality of a random variable is described by the mean? What by the variance?

14. What are the mean and variance of a uniformly distributed random variable $X \sim U(a, b)$?

15. Explain why mean and variance of a Gaussian random variable are enough to describe it completely in a statistical sense.

16. Which probability function is used in practice to describe two random variables jointly?

17. What quality of two jointly distributed random variables is described by the correlation? What by the covariance?

18. When are two jointly distributed random variables statistically independent? Uncorrelated? Orthogonal?

19. What is a correlation coefficient and what is its range?

20. When the correlation between two random variables is zero, why are they said to be orthogonal?

21. When two random variables are statistically independent, they are also uncorrelated. True or false?

22. When two random variables are uncorrelated, they are also statistically independent. True or false?

23. When two random variables are uncorrelated, are they also orthogonal? If not, what condition is needed?

24. When two jointly Gaussian random variables are uncorrelated, they are also statistically independent. True or false?

25. Linear combination of Gaussian random variables may or may not result in a Gaussian distribution. True or false?

26. Describe the basic properties of a covariance (or correlation) matrix.

27. Explain the concept of stochastic process in terms of sample space, outcomes, and ensembles.

28. What is a stationary random process and how does it differ from the wide-sense stationary process?

29. Describe key important properties of the autocovariance and autocorrelation sequences.

30. A stationary random process is filtered through an LTI system resulting in an output process which may or may not be stationary. True or false?

31. When a stationary random process is filtered through an LTI system, the output autocorrelation sequence is obtained by convolving the input autocorrelation sequence with another sequence. What is this sequence and what does it describe about the system?

32. What is power spectral density of a random sequence and what information does it provide about the process.

33. Enumerate basic yet important properties of a power spectral density.

34. What makes a white-noise process "white"?

35. What is a regular process? When are processes $x[n]$ and $y[n]$ linearly equivalent?

36. Describe an ARMA process. How does one obtain an MA process from it? How does one obtain an AR process from it?

37. Describe a harmonic process. Does it always result in a periodic autocorrelation sequence?

Problems

Tutorial problems

1. Three fair dice, colored red, green, and blue, are tossed.
   (a) Determine the probability of getting all three face values equal to 3.
   (b) It is observed that one face value is 3. Now determine the probability of getting all three face values equal to 3.
   (c) It is further observed that the red face value is 3. Now determine the probability of getting all three face values equal to 3.
   (d) *Qualitatively* explain why the above three probabilities make sense.

2. The file `f16.mat` contains noise recorded at the copilot's seat of an F-16 airplane using a 16 bit A/D converter with $F_S = 19.98$ kHz. Use the first $N = 20\,000$ samples of this signal to reproduce Figure 13.4. Hint: Estimate the empirical pdf using function `epdf` with 50 bins.

3. Random variable X is described by the normal "mixture" model of the form $f(x) = \alpha_1 N(x; m_1, \sigma_1^2) + \alpha_2 N(x; m_2, \sigma_1^2)$.
   (a) Generate $10\,000$ samples of X using the model $f(x) = 0.5N(x; 0, 1) + 0.5N(x; 2, 1)$. Using the MATLAB function `epdf` with 50 bins, plot the empirical pdf of X. Determine the mean and standard deviation of X.
   (b) Repeat part (a) using $f(x) = 0.3N(x; 0, 1) + 0.7N(x; 2, 1)$.
   (c) Repeat part (a) using $f(x) = 0.3N(x; 0, 1) + 0.7N(x; 5, 3)$.
   (d) Based on your results in parts (a)–(c), comment on the information provided by the mean and standard deviation about the shape of the pdf.

4. Let X be a random variable with pdf $f_X(x)$.
   (a) Show that the random variable $Y = aX + b$ has pdf $f_Y(y) = \frac{1}{|a|} f_X \left(\frac{y-b}{a} \right)$. Hint: Determine $F_Y(y)$ and then use the property $f_Y(y) = \mathrm{d}F_Y(y)/\mathrm{d}y$.
   (b) Show that if $X \sim N(m, \sigma^2)$ then $Y = aX + b \sim N(am + b, a^2\sigma^2)$.
   (c) Generate $N = 10\,000$ samples of $X \sim N(0, 1)$ and compute the corresponding values of Y for $a = 2$ and $b = 3$.
   (d) Compute the empirical pdfs of X and Y using the MATLAB function `epdf` with 50 bins and compare them with the theoretical distributions predicted in part (b). Hint: Superimpose empirical and theoretical distribution graphs as in Figure 13.4(b).
   (e) Compute the mean and variance of Y using the MATLAB functions `mean` and `var` and verify your answers using results from part (b).

5. The book file `fatherson.txt` contains father-son height data in inches.
   (a) Plot a scatter-graph of the data to understand the relationship between father and son heights.
   (b) Using 100 bins, plot a normalized bar-graph for the father-height data.
   (c) Repeat part(b) for the son-height data.
   (d) Using 100 bins, plot normalized bar-graphs for the conditional father-height data when son's heights are 65 inches and 70 inches.
   (e) Using 100 bins, plot normalized bar-graphs for the conditional son-height data when father's heights are 65 inches and 70 inches.

6. Two random variables, X and Y, have a joint density function $f(x, y)$ which is equal to $\frac{1}{2}$ in the shaded region shown below.
   (a) Determine the marginal densities $f(x)$ and $f(y)$.
   (b) Show that X and Y are uncorrelated.
   (c) Show that X and Y are not independent.
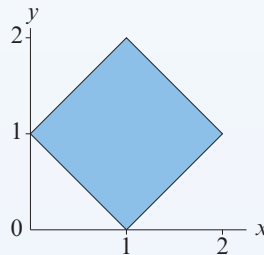
7. In this problem we will study 10 000 samples of bivariate Gaussian random numbers with zero means, unit variances, and an arbitrary correlation coefficient ρ.
   (a) Generate a scatter plot for $\rho = 0.9$.
   (b) Generate a scatter plot for $\rho = -0.9$.
   (c) Generate a scatter plot for $\rho = 0$.
   (d) Comment on your plots.
   The generation of random vectors with multivariate normal distribution (13.49) is discussed in Tutorial Problem 14.20.

8. Consider a 3×1 random vector $\underline{X}$ with a probability density function given by $f(x_1, x_2, x_3) = K \cdot x_1 x_2 x_3$ for $0 \leq x_3 \leq x_2 \leq x_1 \leq 1$ and 0 otherwise where K is a constant. Determine:
   (a) the mean vector $\boldsymbol{m}$,
   (b) the autocorrelation matrix $\boldsymbol{R}$,
   (c) the autocovariance matrix $\boldsymbol{C}$.

9. A random process $x[n]$ is characterized by

$$x[n] = A(\zeta) \cos\left[\Omega(\zeta)n + \Theta(\zeta)\right],$$

where random variables $A(\zeta)$, $\Omega(\zeta)$, and $\Theta(\zeta)$ are mutually independent. Random variables $A(\zeta) \sim U(0, 1)$ and $\Theta(\zeta) \sim U(-\pi, \pi)$ are of continuous type while $\Omega(\zeta)$ is of discrete type taking values 10 and 20 radians with equal probability.

(a) Determine the mean sequence $m_X[n]$.

(b) Determine the ACVS $c_X[m, n]$.

(c) Comment on the stationarity of the process $x[n]$.

10. Let $x[n]$ and $v[n]$ be two mutually independent IID random sequence with marginal pdfs:

$$f(x) = e^{-x}, \ x \geq 0 \text{ and } f(v) = 2e^{-2v}, \ v \geq 0.$$

Define a random sequence $y[n]$ by the difference equation

$$y[n] = x[n] + x[n - 1] + v[n].$$

The random sequence $y[n]$ can be thought of as the result of passing $x[n]$ through a first order moving average filter and then adding noise $v[n]$. Determine:

(a) the mean $m_y[n]$ of $y[n]$,

(b) the ACRS $r_y[m, n]$, and

(c) the marginal density $f(y)$ of $y[n]$ at a fixed n.

11. Show that the *average power* $E(Y^2[n])$ at the output of an FIR system with $h[n] = 0$ for $n < 0$ and $n \geq M$ is

$$E(Y^2[n]) = \sum_{k=0}^{M} \sum_{m=0}^{M} h[k]h[m]r_x[m - k] = \boldsymbol{h}^T \boldsymbol{R}_x \boldsymbol{h}.$$

Use the inverse DTFT relation to show that

$$E(Y^2[n]) = \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 S_{xx}(\omega) d\omega.$$

12. Let $x[n]$ be a WSS process applied to an LTI system with impulse response $h[n]$ to produce $y[n]$. Using the derivation leading to (13.100) and (13.104), show that the identical relations hold for the auto and cross-covariance sequences:

$$c_{xy}[\ell] = \sum_{k=-\infty}^{\infty} h[k]c_{xx}[\ell + k] = h[-\ell] * c_{xx}[\ell],$$

$$c_{yx}[\ell] = \sum_{k=-\infty}^{\infty} h[k]c_{xx}[\ell - k] = h[\ell] * c_{xx}[\ell],$$

$$c_{yy}[\ell] = \sum_{k=-\infty}^{\infty} h[k]c_{xy}[\ell - k] = h[\ell] * c_{xy}[\ell],$$

$$c_{yy}[\ell] = \sum_{m=-\infty}^{\infty} r_{hh}[m]c_{xx}[\ell - m] = r_{hh}[\ell] * c_{xx}[\ell].$$

13. Consider the mse objective function (13.56)

$$J(a, b) = E[(Y - aX - b)^2].$$

(a) Express $J(a, b)$ in terms of the parameters a, b, and the moments of X and Y.

(b) Using partial derivatives $\dfrac{\partial J}{\partial a}$ and $\dfrac{\partial J}{\partial b}$, determine the values of a and b by solving the equations $\partial J / \partial a = 0$ and $\partial J / \partial b = 0$ that minimize $J(a, b)$ to obtain optimum values given in (13.58) and (13.62).

14. Let $x[n]$ be a WSS process applied to an LTI system with impulse response $h[n]$ to produce $y[n]$. Using the derivation leading to (13.110) and (13.111), show that the identical relations hold for the Fourier transforms of the auto and cross-covariance sequences:

$$C_{xy}(z) = H(1/z)C_{xx}(z),$$
$$C_{yx}(z) = H(z)C_{xx}(z),$$
$$C_{yy}(z) = H(z)H(1/z)C_{xx}(z).$$

15. A random process $x[n]$ with ACRS $r_{xx}[\ell] = \frac{1}{2}^{|\ell|}$ is applied as an input to an LTI system with impulse response $h[n]$ to produce the output process $y[n]$. The cross-correlation between $y[n]$ and $x[n]$ was measured to be

$$r_{yx}[\ell] = \left\{ 108 \left(\tfrac{1}{3} \right)^{\ell} - 72 \left(\tfrac{1}{4} \right)^{\ell} - 72 \left(\tfrac{1}{2} \right)^{\ell} \right\} u[\ell].$$

Determine the impulse response $h[n]$.

16. Consider the following AR(2) process

$$y[n] = \frac{5}{6}y[n-1] - \frac{1}{6}y[n-2] + x[n],$$

where $x[n]$ is a zero-mean WGN process with variance equal to 2. Using (13.147), (13.148), and (13.150), determine $r_{xx}[\ell]$ for $\ell \geq 0$.

17. Consider the harmonic process given below:

$$x[n] = \sum_{k=1}^{p} A_k \cos(\omega_k n + \phi_k), \quad \omega_k \neq 0$$

where p, $A_1, \ldots, A_p$, and $\omega_1, \ldots, \omega_p$ are constants and $\phi_1, \ldots, \phi_p$ are pairwise independent random variables uniformly distributed in the interval $(0, 2\pi)$.

(a) Show that the mean of the above process is zero for all n.

(b) Show that its ACRS is given by

$$r_{xx}[\ell] = c_{xx}[\ell] = \frac{1}{2} \sum_{k=1}^{p} A_k^2 \cos \omega_k \ell.$$

Basic problems

18. A random experiment consists of observing the sum of the numbers showing up when two dice are thrown.

(a) Find the sample space S of the experiment.

(b) Determine the probabilities of the following events: $A = \{\text{sum} = 7\}$, $B = \{8 < \text{sum} \le 11\}$, and $C = \{10 < \text{sum}\}$.

19. Companies A, B, C, D, and E each send three delegates to a conference. A committee of four delegates, selected at random, is formed. Determine the following probabilities.

    (a) Company A is not represented on the committee.
    (b) Company A has exactly one representative on the committee.
    (c) Neither company A nor company E is represented on the committee.

20. The joint probability function of random variables X and Y is given by

    $$f_{X,Y}(x, y) = K_1(x + y)e^{-(x+y)}u(x)u(y),$$

    where $u(\cdot)$ is a unit step function.
    (a) Determine K_1.
    (b) Determine $f_{Y|X}(y|x)$ and $f_{X|Y}(x|y)$.
    (c) Are random variables X and Y independent? Justify your answers.

21. Let X be a random variable with pdf $f_X(x)$.
    (a) Show that the random variable $Y = X^2$ has pdf $f_Y(y) = \frac{1}{2\sqrt{y}}[f_X(\sqrt{y}) - f_X(\sqrt{-y})$.
        Hint: Determine $F_Y(y)$ and then use the property $f_Y(y) = dF_Y(y)/dy$.
    (b) Show that if $X \sim N(0, 1)$ then $Y = X^2 \sim \chi_1^2$, that is, a chi-square distribution with one degree of freedom.
    (c) Generate $N = 10\,000$ samples of $X \sim N(0, 1)$ and compute the corresponding values of Y.
    (d) Compute the empirical pdfs of X and Y using the MATLAB function `epdf` with 50 bins and compare them with the theoretical distributions predicted in part (b). Hint: Superimpose empirical and theoretical distribution graphs as in Figure 13.4(b).

22. Consider two jointly distributed random variables X and Y with pdf

    $$f(x, y) = \begin{cases} 8xy, & 0 \le x \le 1, 0 \le y \le x \\ 0. & \text{otherwise} \end{cases}$$

    (a) Determine $f(x), f(y), f(x|y)$, and $f(y|x)$.
    (b) Are X and Y independent?

23. A Gaussian voltage random variable X has a mean value of zero and variance equal to 9. The voltage X is applied to a square-law full wave diode detector with a transfer characteristic $y = 5x^2$. Find the mean and the variance of the output voltage random variable Y.

24. Let random variable $A \sim U[0, 1]$. Define for $n = 1, 2, \ldots$ a discrete-valued random sequence $x[n]$ as the binary expansion of A, that is,

    $$A = \sum_{n=1}^{\infty} x[n]2^{-n} = 0 \cdot b_1 b_2 b_3 \cdots,$$

    where $b_n = 0$ or 1.

(a) Determine the probability mass function (pmf) for the random variable $x[3]$.

(b) Determine the mean sequence $m_x[n]$.

(c) Determine the ACRS $r_x[m, n]$.

25. Let $x[n]$ be a random sequence with

$$P[x[n] = 1] = 0.6 \text{ and } P[x[n] = 0] = 0.4,$$

where components of $x[n]$ are statistically independent random variables. We define a *counting sequence* $y[n]$ as

$$y[n] = \sum_{i=1}^{n} x[i], \ n \geq 1.$$

(a) Determine the mean $m_y[n]$ and the variance $\sigma_y^2[n]$ of Y_n for $n \geq 1$.

(b) Determine the covariance function $c_y[m, n]$ for $m, n \geq 1$.

(c) Let $A = y[m] - y[n]$ be a random variable which describes the counting increment. Determine the variance of A for $m, n \geq 1$.

26. Let $w[n]$ be an IID discrete-valued random sequence with the marginal pmf

$$\Pr(w) = \begin{cases} \frac{1}{4}, & w = -4, 0 \\ \frac{1}{2}. & w = 4 \end{cases}.$$

Let $v[n]$ be another IID *continuous-valued* random sequence, independent of $w[n]$, with the marginal pdf

$$f(v) = \begin{cases} 1/12, & -5 \leq v \leq 7 \\ 0. & \text{otherwise} \end{cases}.$$

Define a new random sequence $x[n]$ by the equation

$$x[n] = w[n] + v[n - 1].$$

(a) Determine the mean $m_w[n]$ and the autocorrelation $r_w[m, n]$ of $w[n]$.

(b) Determine the mean $m_v[n]$ and the autocorrelation $r_v[m, n]$ of $v[n]$.

(c) Determine the cross-correlation sequence $r_{w,v}[(m, n]$ between $w[n]$ and $v[n]$.

(d) Determine the mean $m_X[n]$ of $x[n]$.

(e) Verify that the autocorrelation of $x[n]$ is given by $r_x[m, n] = 4 + 23\delta[m - n]$.

Assessment problems

27. A random variable Y has a probability density

$$f_Y(y) = \begin{cases} ae^{-by}, & y \geq 0 \\ 0. & y < 0 \end{cases}$$

(a) Determine the value of a in terms of b.

(b) Determine the conditional density function for Y, given that $Y > c > 0$.

(c) Determine the conditional distribution function for Y, given that $0 < c < Y < d$.

28. Two joint random variables X and Y are defined through the following construction:
  - The random variable Y is uniformly distributed between 1 and 2.
  - For $Y = y$ the random variable X is exponentially distributed with parameter y.

  (a) Determine the joint density $f_{X,Y}(x, y)$.
  (b) Determine and sketch the marginal densities $f_X(x)$ and $f_Y(y)$.
  (c) Determine and sketch the conditional densities $f_{X|Y}(x \mid y = 1.5)$ and $f_{Y|X}(y \mid x = 1)$.

29. Let $X = \sin(2\pi Z)$, $Y = \cos(2\pi Z)$, and $Z \sim U(0, 1)$. Clearly, by definition, the random variables X and Y are not independent.
  (a) Show that $\text{cov}(X, Y) = 0$, that is, the random variables are uncorrelated.
  (b) Generate 100 sample values of X and Y, plot their scatter diagram, and compute an estimate of their covariance.
  (c) Does the shape of the scatter diagram explain why the dependent random variables X and Y are uncorrelated?

30. Let $x_1 \sim N(m_1, \sigma_1^2)$, $x_2 \sim N(m_2, \sigma_2^2)$ and ρ be the correlation coefficient between them. Define $z_i = (x_i - m_i)/\sigma_i$, $i = 1, 2$. Show that

$$f(z_1, z_2) = \frac{1}{2\pi \sigma_1 \sigma_2 \sqrt{1 - \rho^2}} \exp\left[-\frac{z_1^2 - 2\rho z_1 z_2 + z_2^2}{2\sqrt{1 - \rho^2}}\right].$$

31. Let X_k, $k = 1, 2, 3, 4$, be four independent and identically distributed (IID) random variables, each uniformly distributed between $\left[-\frac{1}{2}, \frac{1}{2}\right]$, that is,

$$f_{X_k}(x) = \begin{cases} 1, & -\frac{1}{2} \le x \le \frac{1}{2} \\ 0, & \text{otherwise} \end{cases} \quad k = 1, 2, 3, 4.$$

  Let $Y_n = \sum_{k=1}^n X_k$, $n = 2, 3, 4$. Use the convolution integral approach to answer the first three parts below.
  (a) Determine and plot the pdf $f_{Y_2}(y)$ of Y_2.
  (b) Determine and plot the pdf $f_{Y_3}(y)$ of Y_3.
  (c) Determine and plot the pdf $f_{Y_4}(y)$ of Y_4.
  (d) Determine and plot the pdf of $N(0, \sigma^2)$ so that 99% of its area is between $[-2, 2]$.
  (e) From the shapes of the above four pdfs, formulate your conclusions regarding the sum of an infinite number of IID random variables.

32. A 2×1 random vector $\underline{X}$ has mean vector m and covariance matrix C given by

$$m = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, C = \begin{bmatrix} 4 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$

  This vector is transformed into a 3×1 random vector $\underline{Y}$ by the following linear transformation:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 2 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}.$$

Determine:

(a) the mean vector m_y,

(b) the autocovariance matrix C_y, and

(c) the cross-correlation matrix r_{xy}.

33. A random sequence $x[n]$ is defined by $x[n] = A\sin(\Omega n)$, $n \geq 0$, in which A and Ω are discrete random variables described by their joint probability mass function

$$
p_{A,\Omega}(a,\omega) = \begin{cases}
1/5, & a = \omega = 0 \\
1/5, & a = 1, \omega = \pi/2 \\
1/5, & a = 1, \omega = -\pi/2 \\
1/5, & a = -1, \omega = \pi/2 \\
1/5, & a = -1, \omega = -\pi/2 \\
0. & \text{otherwise}
\end{cases}
$$

Determine:

(a) the mean sequence $m_x[n]$,

(b) the autocorrelation $r_x[m, n]$, and

(c) whether the sequence x_n is wide-sense stationary, uncorrelated, or orthogonal random sequence.

34. A stationary random sequence $x[n]$ with mean $m_x = 4$ and ACVS

$$
c_x[n] = \begin{cases}
4 - |n|, & |n| \leq 3 \\
0, & \text{otherwise}
\end{cases}
$$

is applied as an input to an LTI system whose impulse response $h[n]$ is

$$
h[n] = u[n] - u[n-4],
$$

where $u[n]$ is a unit step sequence. The output of this system is another random sequence $y[n]$. Determine:

(a) the mean sequence $m_y[n]$,

(b) the cross-covariance $c_{xy}[m, n]$ between $x[m]$ and $y[n]$, and

(c) the autocovariance $c_y[m, n]$ of the output $y[n]$.

35. An uncorrelated random sequence $x[n]$, $n \geq 0$ with mean $m_x[n] = \left(\frac{1}{2}\right)^n u[n]$ and variance $\sigma_x^2[n] = \left(\frac{1}{3}\right)^n u[n]$ is applied as an input to an LTI system with impulse response $h[n] = \left(\frac{1}{4}\right)^n u[n]$.

Let $y[n]$ be the output sequence. Determine:

(a) the ACRS $r_x[m, n]$ of $x[n]$,

(b) the mean sequence $m_y[n]$ of $y[n]$, and

(c) the ACRS $c_y[m, n]$ of $y[n]$.

36. Let $x[n]$ and $y[n]$ be two jointly stationary white noise sequences with

$$
r_x[\ell] = 2\delta(\ell), \quad r_y[\ell] = 3\delta[\ell], \quad \text{and} \quad r_{xy}[\ell] = 4\delta[\ell].
$$

These two sequences are applied as inputs to the LTI system shown below.
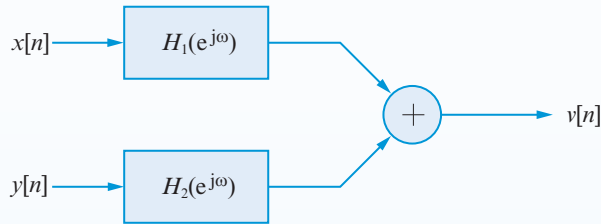
The system $H_1(e^{j\omega})$ is an ideal digital lowpass filter given by

$$H_1(e^{j\omega}) = \begin{cases} 1, & |\omega| \le \frac{\pi}{2} \\ 0, & |\omega| > \frac{\pi}{2} \end{cases}$$

while the system $H_2(e^{j\omega})$ is an ideal digital highpass filter given by

$$H_2(e^{j\omega}) = \begin{cases} 0, & |\omega| < \frac{\pi}{2} \\ 1. & |\omega| \ge \frac{\pi}{2} \end{cases}$$

Determine the power spectral density $S_u(\omega)$.

37. Let $x[n]$ be an independent random sequence which at each n is uniformly distributed over $[0, 2\sqrt{3}]$. It is processed through a digital differentiator given by the impulse response

$$h[n] = \delta[n-1] - \delta[n+1],$$

to obtain the output random sequence $y[n]$.
    (a) Determine the mean $m_x[n]$ and the variance $\sigma_x^2[n]$ of $x[n]$.
    (b) Determine the mean $m_y[n]$ of $y[n]$.
    (c) Determine the autocovariance $c_y[\ell]$.

38. Consider the following AR(4) process

$$y[n] = \frac{5}{6}y[n-1] - \frac{1}{6}y[n-2] + x[n], \tag{13.154}$$

where $x[n]$ is a zero-mean WGN process with variance equal to 2. Using (13.147), (13.148), and (13.150), determine $r_{xx}[\ell]$ for $\ell \ge 0$.

14 Random signal processing

To use stochastic process models in practical signal processing applications, we need to estimate their parameters from data. In the first part of this chapter we introduce some basic concepts and techniques from estimation theory and then we use them to estimate the mean, variance, ACRS, and PSD of a stationary random process model. In the second part, we discuss the design of optimum filters for detection of signals with known shape in the presence of additive noise (matched filters), optimum filters for estimation of signals corrupted by additive noise (Wiener filters), and finite memory linear predictors for signal modeling and spectral estimation applications. We conclude with a discussion of the Karhunen–Loève transform, which is an optimum finite orthogonal transform for representation of random signals.

Study objectives

After studying this chapter you should be able to:

- Compute estimates of the mean, variance, and covariance of random variables from a finite number of observations (data) and assess their quality based on the bias and variance of the estimators used.

- Estimate the mean, variance, ACRS sequence, and PSD function of a stationary process from a finite data set by properly choosing the estimator parameters to achieve the desired quality in terms of bias–variance trade-offs.

- Design FIR matched filters for detection of known signals corrupted by additive random noise, FIR Wiener filters that minimize the mean squared error between the output signal and a desired response, and finite memory linear predictors that minimize the mean squared prediction error.

- Understand the concept, computation, and optimality properties of the Karhunen–Loéve transform of a random vector and use it for data compression applications.

14.1 Estimation of mean, variance, and covariance

In Chapter 13 we assumed that the distributions of random variables or the ACRS and PSD of random processes are *known*. However, this is *not* the case in practical applications, where we can only collect finite sets of observations. Therefore, it is important to estimate the quantities of interest from observed data using sound statistical techniques; this is the subject of estimation theory.

14.1.1 Basic concepts and terminology

To introduce some basic ideas and terminology from estimation theory, suppose that we wish to estimate the mean μ of a random variable X from N observations $x_1, \ldots, x_N$. If we use the arithmetic average as an estimate, we have

$$\bar{x} = \frac{1}{N} \sum_{k=1}^{N} x_k. \tag{14.1}$$

If we use another set of observations $x_1', \ldots, x_N'$, we obtain another estimate $\bar{x}'$, which is very unlikely to be equal to $\bar{x}$. In general, we should expect different observations to yield different estimates. For some observations the value of $\bar{x}$ may turn out to be very close to μ, but for other observations $\bar{x}$ may deviate significantly from μ. The variability of the estimate $\bar{x}$ over different sets of observations can be described by viewing $\bar{x}$ as an observation of a random variable

$$\bar{X} = \frac{1}{N} \sum_{k=1}^{N} X_k, \tag{14.2}$$

where $X_1, \ldots, X_N$ are the random variables that yield the observations $x_1, \ldots, x_N$. The random variable $\bar{X}$ is called an *estimator* for μ, while $\bar{x}$, the numerical value calculated from a particular set of observations, is called an *estimate*. In other words, the estimator is the formula, while the value which it produces for a particular set of observations is the estimate. The distribution of an estimator is called *sampling distribution* to be distinguished from the distribution of the random variables used for the estimation.

In (14.1) and (14.2) we use different symbols to emphasize the difference between random variables and their observed values. Once the logical distinction between these two ideas is understood, it becomes unnecessarily cumbersome to retain the two sets of symbols. *From now on we will only use lower case symbols, but it should be clear from the context whether we are discussing a set of random variables or a particular set of numerical values.*

Consider an estimator $\hat{\theta}$ for a parameter θ; note that we usually put a hat ($\hat{\ }$) over a parameter to denote its estimate or estimator. The quality of an estimator $\hat{\theta}$ is evaluated by the properties of its distribution. A "good" estimator should have a distribution that is centered and highly concentrated around the true value. However, deriving the exact sampling distribution of an estimator is not an easy task. Thus, in practice, we

usually characterize estimators with three important properties: bias, variance, and mean square error.

Bias The bias of an estimator $\hat{\theta}$ of a parameter θ is defined by

$$B(\hat{\theta}) \triangleq \mathrm{E}(\hat{\theta}) - \theta. \tag{14.3}$$

When $B = 0$ the estimator is said to be *unbiased*; otherwise, it is said to be biased. An unbiased estimator gives the correct value on the average when used a large number of times.

Variance The variance of of an estimator

$$\mathrm{var}(\hat{\theta}) = \mathrm{E}\{[\hat{\theta} - \mathrm{E}(\hat{\theta})]^2\} \tag{14.4}$$

shows the spread of its values about its expected value; hence, in general, the variance should be small. A "good" estimator should have zero bias and the smallest possible variance. However, as we show next, the requirements for small bias and small variance are not necessarily compatible.

Mean square error The average deviation of the estimator $\hat{\theta}$ from the true value θ is measured by the mean square error

$$\mathrm{mse}(\hat{\theta}) = \mathrm{E}[(\hat{\theta} - \theta)^2]. \tag{14.5}$$

We may express the mse in terms of the variance and bias by writing

$$\begin{aligned}
\mathrm{mse}(\hat{\theta}) &= \mathrm{E}[(\hat{\theta} - \mathrm{E}(\hat{\theta}) + \mathrm{E}(\hat{\theta}) - \theta)^2] \\
&= \mathrm{E}[(\hat{\theta} - \mathrm{E}(\hat{\theta}) + B(\hat{\theta}))^2] \\
&= \mathrm{E}[\{\hat{\theta} - \mathrm{E}(\hat{\theta})\}^2] + B^2(\hat{\theta}) + 2B(\hat{\theta})\mathrm{E}[\hat{\theta} - \mathrm{E}(\hat{\theta})].
\end{aligned}$$

Noting that $\mathrm{E}[\hat{\theta} - \mathrm{E}(\hat{\theta})] = 0$, we obtain the fundamental expression

$$\mathrm{mse}(\hat{\theta}) = \mathrm{var}(\hat{\theta}) + B^2(\hat{\theta}). \tag{14.6}$$

Therefore, the mean square error takes into consideration both the variance and the bias of the estimator. There are occasions on which we are forced to trade off bias and variance of estimators. For example, an estimator with very low variance and some bias may be more desirable than an unbiased estimator with high variance (see Figure 14.1).

Any sensible estimator should give a better estimate as the number of observations N increases and, in the limiting case, should give the true value θ as $N \to \infty$. This property, called *consistency*, is clearly desirable for any estimator. A sufficient condition for an estimator $\hat{\theta}$ to be *consistent* is that the mse (14.6) or equivalently both its variance and bias converge to zero as the number of observations becomes very large.
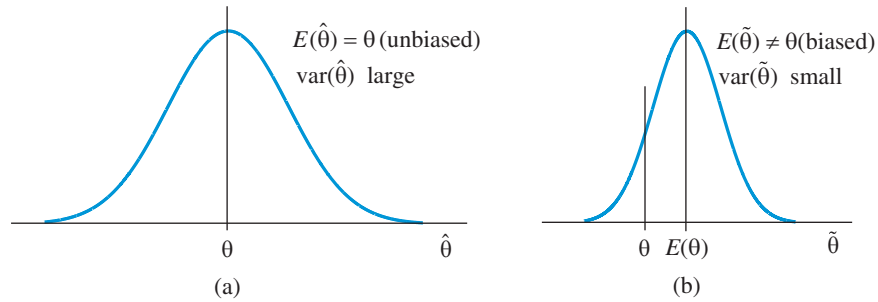
Figure 14.1 Sampling distribution of (a) unbiased, and (b) biased estimators of parameter θ. The biased estimator $\tilde{\theta}$ has smaller mse than the unbiased estimator $\hat{\theta}$.

14.1.2   Sample mean

In practice we usually estimate the mean value μ using the arithmetic average

$$\hat{m} = \frac{1}{N} \sum_{k=1}^{N} x_k. \tag{14.7}$$

This estimate, known as the sample mean, is evaluated using the MATLAB function `m=mean(x)`, where `x` is a vector containing the N observations.

Bias   The center of the sampling distribution of an estimator is usually determined by the expected value. Using (13.22) and (14.7), we obtain

$$\mathrm{E}(\hat{m}) = \frac{1}{N} \sum_{k=1}^{N} \mathrm{E}(x_k) = m, \tag{14.8}$$

that is, the expected value of the estimator (14.7) is equal to the true mean value. The sample mean (14.7) is therefore an unbiased estimator, that is, it gives the correct value on the average when used a large number of times.

Variance   While lack of bias is a desirable property for an estimator, it says nothing about the dispersion of individual estimates about the true value. To evaluate the "concentration" of estimator (14.7) about its mean we need to determine its variance. If we express (14.7) in vector form we have

$$\hat{m} = \frac{1}{N} \sum_{k=1}^{N} x_k \triangleq \frac{1}{N} \boldsymbol{u}^{\mathrm{T}} \boldsymbol{x}, \tag{14.9}$$

where $\boldsymbol{u}$ is an $N \times 1$ vector with unit components. Setting $\boldsymbol{a} = \boldsymbol{u}/N$ in (13.45), we obtain

$$\mathrm{var}(\hat{m}) = \frac{1}{N^2} \boldsymbol{u}^{\mathrm{T}} \boldsymbol{C} \boldsymbol{u}, \tag{14.10}$$

where C is the covariance matrix of random vector x. In order to proceed further, however, we need to know the covariance matrix of the observations. To illustrate the significance of correlation in estimation, we consider two interesting special cases.

If the random variables x_k are mutually uncorrelated, that is, $\mathrm{cov}(x_i, x_j) = 0$ for $i \neq j$ and $\mathrm{var}(x_k) = \sigma^2$, we have $C = \sigma^2 I$. Substitution into (14.10) yields

$$\mathrm{var}(\hat{m}) = \frac{\sigma^2}{N}. \tag{14.11}$$

Thus, the variance of the sample mean of N *pairwise uncorrelated* observations equals the variance of a single observation, divided by N. Therefore, *the estimate $\hat{m}$ for m can be made as precise as desired by taking a sufficiently large number of uncorrelated observations*. From (14.8) and (14.11) we conclude that the sample mean is a consistent estimator because $\mathrm{var}(\hat{m}) = 0$ as $N \to \infty$.

Suppose next that the random variables x_k are highly correlated; in particular, assume that $\mathrm{cov}(x_i, x_j) = \mathrm{var}(x_k) = \sigma^2$ for all values of i and j. In this case, (14.10) yields $\mathrm{var}(\hat{m}) = \sigma^2$; that is, the variance of the sample mean equals the variance of the individual observations. Therefore, we cannot improve the quality of the estimator by increasing the number of observations. *As a rule of thumb, as the correlation between the observations increases, we need more observations to maintain the quality of a given estimator.*

14.1.3   Sample variance

A reasonable choice for an estimator of the variance of a random variable is the *sample variance* defined by the formula

$$\hat{\sigma}^2 \triangleq \frac{1}{N} \sum_{k=1}^{N} (x_k - \hat{m})^2. \tag{14.12}$$

The sample variance is evaluated using MATLAB function s=var(x,1), where x is a vector containing the N observations. It can be shown that the mean of $\hat{\sigma}^2$ for the case of uncorrelated observations is given by (see Problem 23)

$$\mathrm{E}(\hat{\sigma}^2) = \frac{N-1}{N} \sigma^2, \tag{14.13}$$

which shows that (14.12) is a biased estimator of variance. An unbiased estimator $\hat{\sigma}_u^2$ can be obtained by replacing N in (14.12) by $(N-1)$. The MATLAB function s=var(x) computes the biased estimator, whereas function s=var(x,0) provides the unbiased estimator; for large values of N the two estimators are nearly the same. The variance of the biased estimator is given by (see Stuart and Ord (1991))

$$\mathrm{var}(\hat{\sigma}^2) = \frac{N-1}{N^3} \left[(N-1)m_4 - (N-3)\sigma^4 \right], \tag{14.14}$$

where $m_4 = \mathrm{E}[(x - m)^4]$. For normal distributions, where $m_4 = 3\sigma^4$, this expression is simplified to (see Leon-Garcia (2008))

$$\mathrm{var}(\hat{\sigma}^2) = \frac{2(N-1)}{N^2}\sigma^4. \tag{14.15}$$

If m_4 is finite, the variance tends to zero as N tends to infinity. Since the mean and variance of $\hat{\sigma}^2$ tend to zero as the number of observations increases, the sample variance is a consistent estimator.

14.1.4   Sample covariance

Since the covariance is defined by $\mathrm{cov}(x, y) = \mathrm{E}[(x - \mu_x)(y - \mu_y)]$, the covariance could be estimated by the average of the products of the deviations of x and y about their means, that is,

$$\hat{\sigma}_{xy} \triangleq \frac{1}{N}\sum_{k=1}^{N}(x_k - \hat{m}_x)(y_k - \hat{m}_y). \tag{14.16}$$

However, just like the sample variance, this estimator will be biased. To obtain an unbiased estimator we should divide the summation in (14.16) by $(N-1)$ instead of N. The *sample covariance* (14.16) is asymptotically unbiased and its variance goes to zero as N gets very large. Finally, we note that the *sample correlation coefficient* is defined by

$$\hat{\rho}_{xy} \triangleq \frac{\hat{\sigma}_{xy}}{\hat{\sigma}_x\hat{\sigma}_y} = \frac{\sum_{k=1}^{N}(x_k - \hat{m}_x)(y_k - \hat{m}_y)}{\sqrt{\sum_{k=1}^{N}(x_k - \hat{m}_x)^2 \sum_{k=1}^{N}(y_k - \hat{m}_y)^2}}. \tag{14.17}$$

Like its theoretical counterpart, $\hat{\rho}_{xy}$ ranges between -1 and $+1$ in value. However, the properties of $\hat{\rho}_{xy}$ are rather more difficult to find than those of $\hat{\sigma}_{xy}$ because it involves both products and ratios of random variables.

The sample covariance and the sample correlation coefficient are evaluated using the following MATLAB functions (see Tutorial Problem 3):

$$\texttt{C=cov(x,y,1),} \tag{14.18a}$$

$$\hat{\sigma}_x^2\texttt{=C(1,1),} \quad \hat{\sigma}_y^2\texttt{=C(2,2),} \quad \hat{\sigma}_{xy}\texttt{=C(1,2),} \tag{14.18b}$$

$$\texttt{rhoxy=corrcoef(x,y),} \quad \hat{\rho}_{xy}\texttt{=rhoxy(1,2).} \tag{14.18c}$$

14.2   Spectral analysis of stationary processes

The need for power spectral density estimation arises in a variety of applications, including measurement of correlations and power spectral densities for the design of optimum

filters, detection of narrowband signals in wideband noise, estimation of parameters of LTI systems by using a white noise excitation, and extraction of information about a physical system by looking at the absence or presence of power in specific frequency bands.

Estimation of mean, variance, ACRS, and PSD is meaningful *only* for data thought to have come from a stationary process. Since for nonstationary processes all these quantities change with time, any attempt to estimate them may lead to misleading results and conclusions. To understand the issues concerning estimation of second-order moments of wide-sense stationary random processes, we should always take into consideration the fact that a random process is not just a single sequence, but a family of an infinite number of sample sequences.

Ergodicity Since for a specific n, $x[n]$ is a random variable, we can estimate its mean as in Section 14.1.2. We observe N values $x[n, \zeta_k]$ (see Figure 13.9) and we estimate $m_x[n] = \mathrm{E}(x[n])$ by the sample mean

$$\hat{m}[n] = \frac{1}{N} \sum_{k=1}^{N} x[n, \zeta_k].$$

However, in most practical applications we know only a single realization of the process, say $x[n] \triangleq x[n, \zeta]$, and the question is whether we can estimate $\mu_x[n]$ using the time average

$$\langle x[n] \rangle_N = \frac{1}{N} \sum_{n=0}^{N-1} x[n].$$

Clearly, this is *not* possible if $\mathrm{E}(x[n])$ depends on n; however, it may be possible if the process is stationary. If $\langle x[n] \rangle_N$ tends to m_x as $N \to \infty$, we say that the stationary process $x[n]$ is *mean-ergodic*. Similarly, if

$$\langle x[n + \ell]x[n] \rangle_N = \frac{1}{N} \sum_{n=0}^{N-1} x[n + \ell]x[n]$$

tends to $r_x[\ell]$ as $N \to \infty$, the stationary process $x[n]$ is known as *correlation-ergodic*. In general, ergodic properties allow us to equate ensemble averages with infinite length time averages obtained from a single "representative" sequence of the ensemble. In practice, *an intuitive justification for ergodicity is whether the available sample sequence "looks" sufficiently random so that variations along the time axis can be assumed to represent typical variations over the ensemble.*

14.2.1   Estimation of mean, variance, and ACVS/ACRS

In practice, the first step in the analysis of random signals is estimation and removal of the mean value from the data. However, this may be a potentially misleading operation unless all systematic trends have been removed from the data to ensure the stationarity assumption (see Tutorial Problem 4).

Estimation of mean Suppose that we have N observations $\{x[n], 0 \leq n \leq N-1\}$ from a wide-sense stationary process having mean m, variance σ^2, and ACRS $r[\ell]$ or ACVS $\gamma[\ell]$. Since the mean is constant, a meaningful estimate is the sample mean defined by

$$\hat{m} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]. \tag{14.19}$$

Since $E(x[n]) = m$ for every n, the expectation of $\hat{m}$ is given by

$$E(\hat{m}) = \frac{1}{N} \sum_{n=0}^{N-1} E(x[n]) = m, \tag{14.20}$$

which shows that the sample mean is an unbiased estimator of m. Determining the variance of $\hat{m}$, using (14.10), requires the covariance matrix $\mathbf{C}$. Since the process is stationary, the matrix $\mathbf{C}$ is Toeplitz with elements $\text{cov}(x[i], x[j]) = c(|i-j|)$. Adding the elements across each diagonal of $\mathbf{C}$ in (14.10) yields the relation (recall that $\rho[\ell] = c[\ell]/\sigma^2$)

$$\text{var}(\hat{m}) = \frac{\sigma^2}{N} \left[1 + 2 \sum_{\ell=1}^{N-1} \left(1 - \frac{\ell}{N} \right) \rho[\ell] \right]. \tag{14.21}$$

If the process is white noise, we obtain the usual result $\text{var}(\hat{m}) = \sigma^2/N$. However, if the process is highly correlated $\text{var}(\hat{m})$ may differ considerably from σ^2/N.

To understand the implications of (14.21) we consider a process with ACVS $c[\ell] = a^{|\ell|}$, $|a| < 1$. Then, for large N, (14.21) reduces to (see Tutorial Problem 5)

$$\text{var}(\hat{m}) = \frac{\sigma^2}{N} \left(\frac{1+a}{1-a} \right). \tag{14.22}$$

In this case, we can define an "equivalent number of uncorrelated observations" N_{eq} so that $\text{var}(\hat{m}) = \sigma^2/N_{\text{eq}}$. This yields $N_{\text{eq}} = N(1-a)/(1+a)$. For example, for $a = 0.9$ we have $\text{var}(\hat{m}) \simeq \sigma^2/(N/20)$; that is, we need 20 times more observations to obtain an estimate with the same variance. When $a > 0$ we need more data to estimate the mean with the same accuracy; on the other hand, we need fewer data when $a < 0$. In other words, negatively correlated data "contain" more information as far as estimation of the mean value is concerned.

Estimation of ACVS/ACRS Given N observations $x[0], x[1], \ldots, x[N-1]$, we can form $N - \ell$ pairs of observations, namely,

$$(x[0], x[\ell]), (x[1], x[\ell+1]), \ldots, (x[N-\ell-1], x[N-1]), \tag{14.23}$$

where each pair is separated by ℓ sampling intervals. Regarding the first observation in each pair as one variable, and the second observation as a second variable, we can generate a scatter plot to visually access the correlation between samples, say $x[n]$ and $x[n+\ell]$, that are ℓ-points apart. The sample ACVS is usually estimated by

$$\hat{c}[\ell] = \frac{1}{N} \sum_{n=0}^{N-\ell-1} (x[n] - \hat{m})(x[n+\ell] - \hat{m}), \ 0 \leq |\ell| \leq N-1 \tag{14.24}$$

which is then used to evaluate the sample autocorrelation coefficient sequence

$$\hat{\rho}[\ell] = \frac{\hat{c}[\ell]}{\hat{\sigma}^2} = \frac{\hat{c}[\ell]}{\hat{c}[0]}. \tag{14.25}$$

If we ignore the effect of estimating μ, the expectation of (14.24) is

$$\mathrm{E}(\hat{c}[\ell]) = \frac{1}{N} \sum_{n=0}^{N-\ell-1} c[\ell] = \left(1 - \frac{|\ell|}{N}\right) c[\ell]. \tag{14.26}$$

Since for each ℓ, $|\ell|/N \to 0$ as $N \to \infty$, $\hat{c}[\ell]$ is an asymptotically unbiased estimator of ACVS. Estimating the variance of $\hat{c}[\ell]$ is complicated because it involves fourth-order moments. For large N, the variance of the estimator $\hat{c}[\ell]$ is given by (see Jenkins and Watts (1968))

$$\mathrm{var}(\hat{c}[\ell]) \simeq \frac{\sigma^4}{N} \sum_{k=-\infty}^{\infty} (\rho^2[k] + \rho[k+\ell]\rho[k-\ell]). \tag{14.27}$$

From (14.26) and (14.27) it follows that (14.24) is a consistent estimator of $c[\ell]$ because its bias and variance tend to zero as the number of observations increases. This indicates that successive values of $\hat{c}[\ell]$ may be highly correlated and that $\hat{c}[\ell]$ may fail to die out even if it is expected to do so. This makes the interpretation of sample ACVS graphs quite challenging because we do not know whether the variation is real or statistical. This is illustrated in Figure 14.2 with the estimated ACVS of a white Gaussian noise process. Recall that the theoretical values are $c[0] = \sigma^2$ and $c[\ell] = 0$ for $\ell \neq 0$.
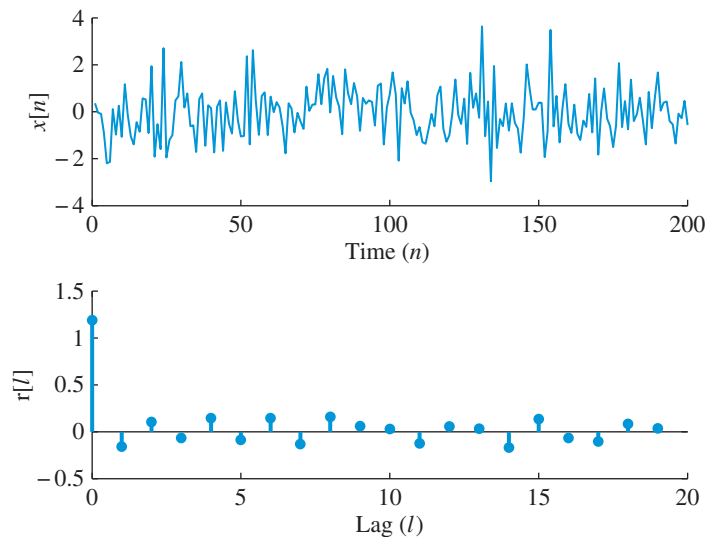
Figure 14.2 Data from a white Gaussian noise process and its sample ACVS.

```
function r=acrs(x,L)
% Computes the ACRS r[m] for 0<= m <= L
% r=acrs(x-mean(x),L) yields the ACVS
N=length(x);
x1=zeros(N+L-1,1);
x2=x1;
x1(1:N,1)=x;
for m=1:L
    x2=zeros(N+L-1,1);
    x2(m:N+m-1,1)=x;
    r(m)=x1'*x2;
end
r=r(:)/N;
```

Figure 14.3 Computation of ACRS using (14.28).

In practice, usually, either we assume a zero-mean process or we remove the estimated mean value from the data. Then, we evaluate the sample ACRS by

$$\hat{r}[\ell] = \frac{1}{N} \sum_{n=0}^{N-\ell-1} x[n]x[n+\ell], \ 0 \le \ell \le N-1 \tag{14.28}$$

where $\hat{r}[\ell] = 0$ for $\ell \ge N$ and $\hat{r}[-\ell] = \hat{r}[\ell]$ for all $\ell < 0$. The mean and variance of $\hat{r}[\ell]$ can be easily obtained from (14.26) and (14.27). The ACVS and the ACRS can be evaluated using the MATLAB function r=acrs(x,L) shown in Figure 14.3. In practice we calculate $\hat{c}[\ell]$ for $0 \le \ell \le L-1$ for $L \ll N$. According to Box *et al.* (2008), a good rule of thumb is that N should be at least 50 and that $L \le N/4$.

Estimation of variance The variance of a stationary process is estimated by

$$\hat{\sigma}^2 = \hat{c}[0] = \frac{1}{N} \sum_{n=0}^{N-1} (x[n] - \hat{m})^2. \tag{14.29}$$

The variance of this estimator is obtained from (14.27) by setting $\ell = 0$,

$$\text{var}(\hat{\sigma}^2) \simeq \frac{2}{N} \sum_{k=-\infty}^{\infty} c^2[k] = \frac{2\sigma^4}{N} \sum_{k=-\infty}^{\infty} \rho^2[k]. \tag{14.30}$$

If the autocorrelation coefficient is given by $\rho[\ell] = a^{|\ell|}$, (14.30) yields

$$\text{var}(\hat{\sigma}^2) \simeq \frac{2\sigma^4}{N} \left(\frac{1+a^2}{1-a^2}\right). \tag{14.31}$$

Therefore, comparing to (14.22), we observe that the "equivalent number of uncorrelated observations" is given by $N_{eq} = N(1 - a^2)/(1 + a^2)$ [see the discussion following equation (14.22)]. Note that in contrast to (14.22) we need the same N_{eq} for both positive and negative values of a. The implications of (14.31) are investigated in Tutorial Problem 6.

<table>
<tr><td>14.2.2</td><td></td></tr>
</table>

The periodogram

The PSD of a zero-mean wide-sense stationary process with absolutely summable ACRS is a continuous ("smooth") function of ω defined by (see Section 13.4.4)

$$S(\omega) = \sum_{\ell=-\infty}^{\infty} r[\ell] e^{-j\omega\ell}. \tag{14.32}$$

Suppose now that we wish to estimate the function $S(\omega)$ from a finite set of observations $\{x[n], 0 \le n \le N-1\}$ of a single realization. A natural estimate of the PSD can by obtained by replacing the true ACRS $r[\ell]$ in (14.32) by the sample ACRS $\hat{r}[\ell]$, given by (14.28), for all available lags. The result is the following estimator

$$I(\omega) \triangleq \sum_{\ell=-(N-1)}^{N-1} \hat{r}[\ell] e^{-j\omega\ell}. \tag{14.33}$$

Since $\hat{r}[\ell]$ at the unavailable lags is arbitrarily set to zero, this estimator will give large errors when $r[\ell]$ differs significantly from zero for $|\ell| > N$. As we show next, the estimator $I(\omega)$ can be expressed directly in terms of the observations by

$$I(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \right|^2. \tag{14.34}$$

The estimator $I(\omega)$ is known as the *periodogram*, although it is a function of frequency and not the period. The periodogram was introduced by Schuster in 1898 in his efforts to search for hidden periodicities in solar sunspot data (see Percival and Walden (1993)). To prove the equivalence of (14.33) and (14.34), we first note that

$$I(\omega) = \frac{1}{N} \left(\sum_{n=0}^{N-1} x[n] e^{-j\omega n} \right) \left(\sum_{m=0}^{N-1} x[m] e^{-j\omega m} \right)^*$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} x[n] x[m] e^{-j\omega(n-m)}. \tag{14.35}$$

The terms in the double summation (14.35) can be arranged in matrix form as illustrated next for the special case of $N = 4$:

$$
\begin{bmatrix}
x[0]x[0] & x[0]x[1]e^{-j\omega} & x[0]x[2]e^{-j2\omega} & x[0]x[3]e^{-j3\omega} \\
x[1]x[0]e^{j\omega} & x[1]x[1] & x[1]x[2]e^{-j\omega} & x[1]x[3]e^{-j2\omega} \\
x[2]x[0]e^{j2\omega} & x[2]x[1]e^{-j\omega} & x[2]x[2] & x[2]x[3]e^{-j\omega} \\
x[3]x[0]e^{j3\omega} & x[3]x[1]e^{-j2\omega} & x[3]x[2]e^{-j\omega} & x[3]x[3]
\end{bmatrix}.
\tag{14.36}
$$

Careful inspection of this matrix shows that (14.34) computes $I(\omega)$ by adding the elements columnwise or rowwise, whereas (14.33) computes $I(\omega)$ by adding the elements along the diagonals parallel to the main diagonal.

The equivalence of (14.33) and (14.34) allows us to use the most convenient form to derive the properties of the periodogram. From (14.34) we have that $I(\omega) \geq 0$, which implies that $\hat{r}[\ell]$ is nonnegative definite, that is, $\boldsymbol{R}_p$ is nonnegative definite for every p; see Papoulis and Pillai (2002). Since $\hat{r}[\ell]$ and $I(\omega)$ form a Fourier transform pair, we have

$$
\hat{r}[\ell] = \frac{1}{2\pi} \int_{-\pi}^{\pi} I(\omega) e^{j\omega\ell} d\omega.
\tag{14.37}
$$

For $\ell = 0$, we obtain an expression for the average power in terms of the periodogram

$$
\hat{r}[0] = \frac{1}{N} \sum_{n=0}^{N-1} x^2[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} I(\omega) d\omega.
\tag{14.38}
$$

Thus, the periodogram shows how the average power of the data segment is distributed as a function of frequency.

Computation The periodogram $I(\omega)$ is a continuous function of ω; hence we can compute it only at a discrete set of frequencies. We usually evaluate $I(\omega)$ at the set of frequencies $\omega_k = 2\pi k/K$, $k = 0, 1, \ldots, K - 1$, $K \geq N$. This is done by the MATLAB function $I=\texttt{psdper}(\texttt{x},\texttt{K})$, which computes a K-point DFT of the data segment (see Figure 14.4).

```
function I=psdper(x,K)
% K-point FFT >= N
N=length(x);
X=fft(x,K);
I=X.*conj(X)/N;
I(1)=I(2); % Avoid DC bias
I=I(:);
```

Figure 14.4 Computation of periodogram $I(\omega)$ using the FFT.

```
function r=acrsfft(x,L)
% r=acrsfft(x-mean(x),L) yields the ACVS
N=length(x);
Q=nextpow2(N+L);
X=fft(x,2^Q);
r0=real(ifft(X.*conj(X)));
r=r0(1:L)/N;
```

Figure 14.5 Computation of ACRS using the FFT.

The integral in (14.38) can be approximated by the sum below:

$$\hat{r}[0] \simeq \frac{1}{2\pi} \sum_{k=0}^{K-1} I\left(\frac{2\pi}{K}k\right) \frac{2\pi}{K} = \frac{1}{K} \sum_{k=0}^{K-1} I\left(\frac{2\pi}{K}k\right) = \texttt{sum(I)/K}. \qquad (14.39)$$

Since $\hat{r}[\ell]$ is the inverse DFT of $I(2\pi k/N)$ we can use the FFT algorithm to compute $\hat{r}[\ell]$ using the function `r=acrsfft(x,L)` shown in Figure 14.5. To avoid time-domain aliasing the length of the DFT should be at least $N + L$ points (see Tutorial Problem 7).

14.2.3   Statistical properties of the periodogram

It would be natural to expect that the periodogram $I(\omega)$, which is a Fourier transform of the sample ACRS $\hat{r}[\ell]$, should inherit the asymptotically unbiased and consistent properties of $\hat{r}[\ell]$. However, as we illustrate in the following example, $I(\omega)$ does *not* posses all these desirable statistical properties.

Example 14.1   Periodogram of white noise process

In this example, we investigate the behavior of the periodogram when the data come from a realization of a white Gaussian noise process $x[n] \sim \text{WGN}(0, \sigma^2)$ with $\sigma^2 = 2$. The PSD of $x[n]$ is $S(\omega) = \sigma^2 = 2$, $-\pi < \omega \leq \pi$. We generate a segment of 512 samples and we compute the periodogram $I_N(\omega_k)$ from the segment consisting of the first 128 samples, the first 256, the first 384, and the entire record of 512 samples. Typical graphs of $I_N(\omega_k)$ and the theoretical PSD $S(\omega) = \sigma^2$ are shown in Figure 14.6.

A simple inspection of Figure 14.6 reveals that the curves representing $I_N(\omega)$ have an erratic and wildly fluctuating form and bear little resemblance to the theoretical PSD. These fluctuations would make it difficult to conclude on the basis of this diagram that the true PSD corresponds to a white noise process. However, what is more disturbing is that these fluctuations do not seem to decrease as the length of the data segment increases.

Since $S(\omega)$ has the same value σ^2 at all frequencies, we can statistically analyze the fluctuations of $I_N(\omega_k)$ by evaluating their mean and variance over frequency. The results in Table 14.1 show that the mean values for each periodogram are close to two, the value of the true PSD; however, the variances do *not* decrease as N increases. In fact, it appears that the standard deviation is close to two, the value to be estimated. Therefore, the periodogram $I_N(\omega)$ is not a good PSD estimator for a white noise process. It is safe to say that if $I_N(\omega)$

Table 14.1 Behavior of $I(\omega)$ of white noise as the segment length is increased.

N	128	256	384	512
Mean of $I(\omega)$	2.3581	2.0370	2.1205	2.0170
Variance of $I(\omega)$	5.6406	4.1541	3.6799	4.5522
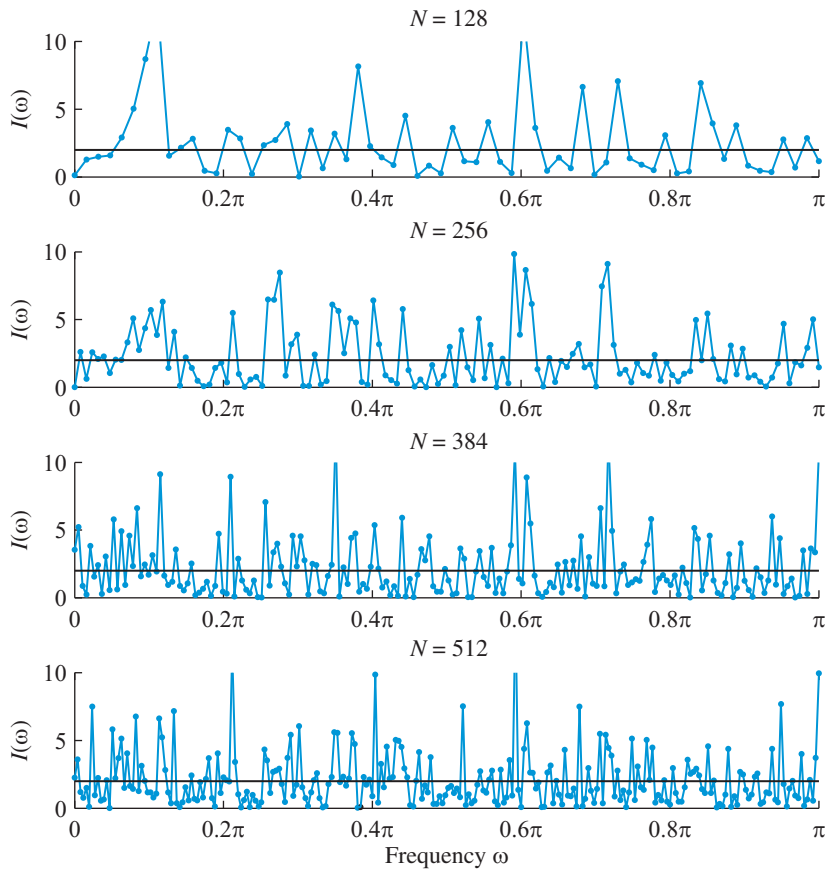
Figure 14.6 Periodogram estimates of a white Gaussian noise process using segments of different length. The horizontal lines represent the true PSD of the white noise process.

is not a good estimator for a flat (that is, constant) spectrum, it cannot be expected to be a good estimator for more complicated spectra. ∎

Since for each value of ω, the estimator $I(\omega)$ is a random variable, the erratic behavior of the periodogram, which is illustrated in Figure 14.6, can be explained by considering its mean, variance, and covariance.

Mean of the periodogram Taking the mathematical expectation of (14.33) and using (14.26) (recall that $\hat{c}[\ell] = \hat{r}[\ell]$ for $m = 0$), we obtain the relation

$$\mathrm{E}[I(\omega)] = \sum_{\ell=-(N-1)}^{N-1} \mathrm{E}(\hat{r}[\ell])\mathrm{e}^{-\mathrm{j}\omega\ell} = \sum_{\ell=-(N-1)}^{N-1} \left(1 - \frac{|\ell|}{N}\right) r[\ell]\mathrm{e}^{-\mathrm{j}\omega\ell}. \tag{14.40}$$

Since $\mathrm{E}[I(\omega)] \neq S(\omega)$, the periodogram is a *biased* estimator of $S(\omega)$. However, for each $|\ell|$, the factor $(1 - |\ell|/N) \to 1$ as $N \to \infty$. Hence, we have

$$\lim_{N\to\infty} \mathrm{E}[I(\omega)] = S(\omega), \tag{14.41}$$

that is, *the periodogram is an asymptotically unbiased estimator of $S(\omega)$.*

If we define the zero-phase triangular or Bartlett window

$$w_B[\ell] \triangleq \begin{cases} 1 - \frac{|\ell|}{N}, & 0 \le |\ell| \le N - 1 \\ 0, & |\ell| \ge N \end{cases} \tag{14.42}$$

and we use the frequency convolution theorem (4.152), we can express (14.40) as

$$\mathrm{E}[I(\omega)] = \sum_{\ell=-(N-1)}^{N-1} w_B[\ell]r[\ell]\mathrm{e}^{-\mathrm{j}\omega\ell} \tag{14.43}$$

$$= \frac{1}{2\pi} \int_{-\pi}^{\pi} S(\theta)W_B(\omega - \theta)\mathrm{d}\theta, \tag{14.44}$$

where the Fourier transform of $w_B[\ell]$ is given by

$$W_B(\omega) \triangleq \frac{1}{N} \left(\frac{\sin \omega N/2}{\sin \omega/2}\right)^2. \tag{14.45}$$

Thus, the expected value of the periodogram is obtained by convolving the true PSD $S(\omega)$ with the Fourier transform $W_B(\omega)$ of the Bartlett window (see Figure 14.7). For $\mathrm{E}[I(\omega)]$ to be as close as possible to $S(\omega)$, $W_B(\omega)$ should be a close approximation to a unit impulse $\delta(\omega)$. The mainlobe of $W_B(\omega)$ has an approximate 3 dB bandwidth of $2\pi/N$ rads. Therefore, $W_B(\omega)$ is a poor approximation of $\delta(\omega)$ for small values of N; however, $W_B(\omega)$ approaches a periodic impulse train as N approaches infinity. The average periodogram $\mathrm{E}[I(\omega)]$ approaches asymptotically, as $N \to \infty$, the true PSD $S(\omega)$.

There are two types of bias: *smearing* and *leakage*. The predominant effect of the main lobe of $W_B(\omega)$ is to *smear* or *smooth* the periodogram. Since the mainlobe has an "effective" width of $2\pi/N$ rads, two equal amplitude sinusoids with frequencies less than $2\pi/N$ rads apart cannot be separated. In this sense, we say that the *spectral resolution* of the periodogram estimator is approximately $2\pi/N$ rads. The major effect of the sidelobes is to transfer power from frequency bands that contain large amounts of signal
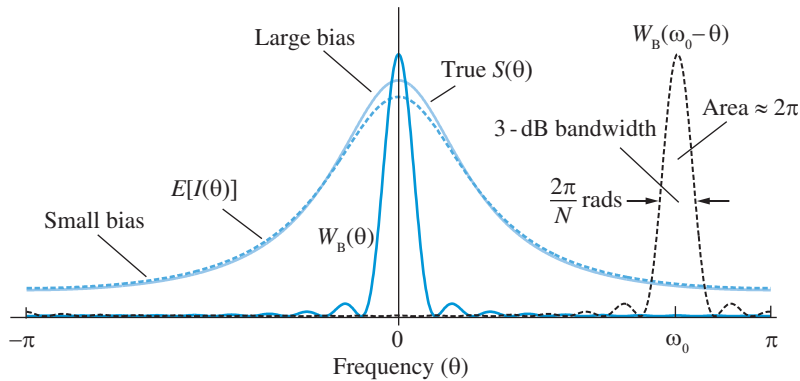
Figure 14.7 Generation of bias by smoothing the PSD in the frequency domain. $E[I(\omega)]$ is determined by evaluating the DTFT of $w_B[\ell]r[\ell]$.

power into bands that contain little or no power. This effect is known as *leakage*. Since a large mean value produces a strong frequency component at $\omega = 0$ rads, we always *remove the sample mean* from the data to avoid leakage. This procedure affects mainly the values of $\hat{S}(\omega)$ in the neighborhood of $\omega = 0$. In practice, we avoid this problem by setting $\hat{S}(0)$ equal to the next available value $\hat{S}(\omega_1)$. For a detailed discussion of leakage see Section 7.6.1.

Smearing and leakage are especially critical for PSDs with strong peaks and valleys. For relatively "flat" PSDs these effects are less significant. Indeed, for a white noise process with a maximally "flat" PSD $S(\omega) = \sigma^2$, we have

$$E[I(\omega)] = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_B(\theta)S(\omega - \theta)d\theta = \sigma^2 \left[\frac{1}{2\pi} \int_{-\pi}^{\pi} W_B(\theta)d\theta \right]. \tag{14.46}$$

Therefore, if we normalize the Fourier transform $W_B(\omega)$ so that

$$w_B[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} W_B(\theta)d\theta \triangleq 1, \tag{14.47}$$

it follows that *for a white noise process the periodogram is an unbiased estimator regardless of the value of N*. To explain this fact, we note that due to the lack of peaks and valleys in $S(\omega)$ the convolution operation (14.46) does not introduce any distortion (smearing or leakage). If $S(\omega)$ does not change significantly within the mainlobe of $W_B(\omega)$, we have

$$E[I(\omega)] = W_B(\omega) \otimes S(\omega) \simeq S(\omega)\frac{1}{2\pi} \int_{-\pi}^{\pi} W_B(\theta)d\theta, \tag{14.48}$$

which shows that normalizing the window by (14.47) is a requirement for $I(\omega)$ to be an unbiased estimator of $S(\omega)$.

Variance and covariance of the periodogram To obtain the variance of $I(\omega)$, we first exploit the symmetry of $\hat{r}[\ell]$ to express the periodogram (14.33) as

$$I(\omega) = \hat{r}[0] + \sum_{\ell=1}^{N-1} (2\cos\omega\ell)\,\hat{r}[\ell]. \tag{14.49}$$

This relation shows that the evaluation of $I(\omega)$ involves N sample autocorrelations. Although the variance of each sample autocorrelation is proportional to σ^4/N (see (14.15)) for large N, the cumulative effect of the N terms in (14.49) produces a variance which is proportional to σ^4. For a white noise process $S(\omega) = \sigma^2$, therefore var$[I(\omega)] \propto S^2(\omega)$. We note that although the sample autocorrelations $\hat{r}[\ell]$ are not, in general, uncorrelated, the basic effect is the same.

It turns out, see Jenkins and Watts (1968), that this result holds for most random processes of practical interest. Indeed, for sufficiently large values of N, we have

$$\text{var}[I(\omega)] \approx S^2(\omega), \quad 0 < \omega < \pi \tag{14.50}$$

which is doubled at $\omega = 0$ or π. Since the variance of $I(\omega)$ at frequencies $\omega = 0, \pi$ is double, we sometimes ignore the values of the PSD estimate at these frequencies. The main implication of (14.50) is that *the periodogram is* not *a good estimator of PSD because, independently of N, the standard deviation of the estimator is as large as the quantity to be estimated.* In other words, the periodogram is not a consistent estimator; that is, its distribution does not tend to cluster more closely around the true PSD as N increases. Thus the definition of the PSD by $S(\omega) = \lim_{N\to\infty} I(\omega)$ is not valid because even if $\lim_{N\to\infty} \text{E}[(\omega)] = S(\omega)$, the variance of $I(\omega)$ does not tend to zero as $N \to \infty$.

The covariance between values of the periodogram at harmonically spaced frequencies is given by, see Jenkins and Watts (1968),

$$\text{cov}\left[I\left(2\pi k_1/N\right), I\left(2\pi k_2/N\right)\right] \approx 0, \; k_1 \neq k_2 \tag{14.51}$$

which states that closely spaced values of the periodogram are uncorrelated. As the segment length N increases, the distance between neighboring frequency samples that are uncorrelated decreases; however, as we have seen in (14.50), the variance remains the same. These conditions cause rapid fluctuations of the periodogram as the segment length increases.

14.2.4   The modified periodogram

Figure 14.7 suggests that we could possibly reduce bias by replacing $W_B(\omega)$ with another window $W_c(\omega)$ having a "better" shape. To understand how the Bartlett window enters into the computation of $\text{E}[I(\omega)]$, we define the *modified periodogram*

$$\tilde{I}(\omega) \triangleq \frac{1}{N} \left| \sum_{n=0}^{N-1} w[n]x[n]e^{-j\omega n} \right|^2, \tag{14.52}$$

where $w[n]$ is a *data window* of duration N. The purpose of using a data window is to reduce the spectral leakage caused by strong narrowband components by lowering the level of sidelobes. If we define the windowed signal

$$v[n] \triangleq w[n]x[n], \tag{14.53}$$

we can express (14.52) as

$$\tilde{I}(\omega) = \sum_{\ell=-(N-1)}^{N-1} \hat{r}_v[\ell]e^{-j\omega\ell}, \tag{14.54}$$

where

$$\hat{r}_v[\ell] = \frac{1}{N} \sum_{n=0}^{N-\ell-1} w[n+\ell]w[n]x[n+\ell]x[n]. \tag{14.55}$$

Taking the expectation of (14.55) we obtain

$$E(\hat{r}_v[\ell]) = \frac{1}{N} \sum_{n=0}^{N-\ell-1} w[n+\ell]w[n]E(x[n+\ell]x[n]) = w_c[\ell]r[\ell], \tag{14.56}$$

where

$$w_c[\ell] = \frac{1}{N} \sum_{n=0}^{N-\ell-1} w[n+\ell]w[n] = \frac{1}{N}w[\ell] * w[-\ell]. \tag{14.57}$$

The expected value of the modified periodogram is

$$E[\tilde{I}(\omega)] = \sum_{\ell=-(N-1)}^{N-1} w_c[\ell]r[\ell]e^{-j\omega\ell} = W_c(\omega) \otimes S(\omega), \tag{14.58}$$

which is a generalization of (14.44) to arbitrary windows. Indeed, if $w[n]$ is the rectangular window then $w_c[n]$ becomes the Bartlett window. The window $w_c[\ell]$ is known as a *correlation* or *lag window* because it is applied into the ACRS; in contrast, the data window $w[n]$ is applied into the observed sequence.

There are two ways to normalize the data window that lead to the same result. First, we note that the inverse Fourier transform of (14.58) for $\ell = 0$ yields

$$w_c[0]r[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} E[\tilde{I}(\omega)]d\omega. \tag{14.59}$$

```
function I=psdmodper(x,K)
% K-point FFT >= N
N=length(x);
w=hann(N); % choose window
w=w/(norm(w)/sqrt(N)); % sum w^2[k]=N
X=fft(x(:).*w(:),K);
I=X.*conj(X)/N;
I(1)=I(2); % Avoid DC bias
I=I(:);
```

Figure 14.8 Computation of the modified periodogram PSD estimate.

If we choose $w_c[0] = 1$, the area under the modified periodogram provides an unbiased estimator of the process power. Alternatively, from (14.56) we note that for the quantity $\hat{r}_v[0] = w_c[0]r[0]$ to be an unbiased estimator of power, we require that $w_c[0] = 1$. Using (14.57) we obtain the following normalization for the equivalent data window

$$\sum_{n=0}^{N-1} w^2(n) = N. \tag{14.60}$$

This condition ensures that the periodogram of $x[n]$ is asymptotically unbiased, (see the discussion following (14.45)). We emphasize that in contrast to (14.38), which holds for every single realization, equation (14.59) holds on the average. The modified periodogram (14.52) can be computed using the MATLAB function `psdmodper` shown in Figure 14.8.

Example 14.2   Illustration of leakage in the periodogram
We consider an AR(2) process and an AR(4) process defined by the following difference equations

$$x[n] = 0.75x[n-1] - 0.5x[n-2] + v[n],$$
$$x[n] = 2.7607x[n-1] - 3.8106x[n-2] + 2.6535x[n-3]$$
$$- 0.9238x[n-4] + v[n],$$

where $v(n) \sim \text{WGN}(0, 1)$. Both processes have been used extensively in the literature for power spectrum estimation studies, see Percival and Walden (1993). Their true power spectral densities are obtained using (13.133). For simulation purposes, $N = 1024$ samples of each process were generated. The dynamic range of the two spectra, that is, the ratio $10\log_{10}[\max_{\omega} S_x(\omega)/\min_{\omega} S_x(\omega)]$, is about 15 and 65 dB, respectively.

From the sample realizations, periodograms and modified periodograms, based on the Hanning window, were computed by using (14.52) at $K = 1024$ frequencies. These are shown in Figure 14.9. The periodograms for the AR(2) and AR(4) processes, respectively, are shown in the left column while the modified periodograms are shown in the right column of the figure. These plots illustrate that the periodogram is a biased estimator of the
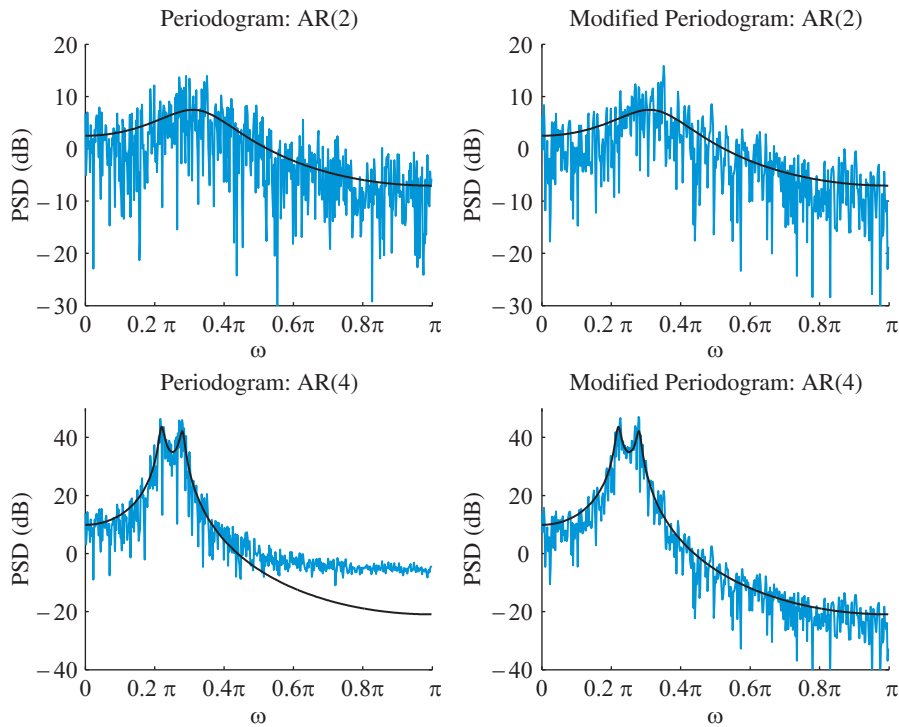
Figure 14.9 Properties of periodogram as a power spectrum estimator. When the PSD has a large dynamic range, windowing helps to avoid leakage. The smooth curves show the PSD and the fluctuating curves show the periodogram.

power spectrum. In the case of the AR(2) process, since the spectrum has a small dynamic range (15 dB), the bias in the periodogram estimate is not obvious; furthermore, the windowing in the modified periodogram did not show much improvement. On the other hand, the AR(4) spectrum has a large dynamic range, and hence the bias is clearly visible at high frequencies. This bias is clearly reduced by windowing the data in the modified periodogram. In both cases, the random fluctuations are not reduced by the data windowing operation. ■

Failure of periodogram as a PSD estimator From (14.50) and (14.51), which are the key formulas for understanding the statistical behavior of periodograms, we conclude that:

- The variance of the periodogram, which is of the order of $S^2(\omega)$, does not decrease as the number N of observations increases.
- The values of the periodogram for ω equal to $2\pi k/N$ are uncorrelated. As N increases, the spacing between contiguous uncorrelated values decreases.

These two properties demonstrate that as a function of ω, the periodogram $I(\omega)$, has an *erratic* and *wildly fluctuating* form (see Figure 14.6). Therefore, the "rough" periodogram $I(\omega)$ is a *very poor* estimator of the "smooth" PSD function $S(\omega)$. When $S(\omega)$ is continuous

the mean and variance of $I(\omega)$ are proportional to $S(\omega)$ and $S^2(\omega)$, respectively, and hence $I(\omega)$ may produce spurious peaks in regions where $S(\omega)$ is relatively large.

We shall attempt to overcome this problem by smoothing the periodogram, that is, by replacing $I(\omega)$ by a weighted sum of neighboring values or by averaging multiple periodograms from the same process. Both approaches render the estimator well-behaved in the sense of having a small variance; however, to obtain the reduction of variance we have to pay a price in the form of increasing the bias of the estimator. Due to stationarity, the two approaches should provide comparable results under similar circumstances.

<h3>14.2.5   The Blackman–Tukey method: smoothing a single periodogram</h3>

The reason why the variance of $I(\omega)$ does not tend to zero is because, loosely speaking, the computation of (14.49) involves "too many" sample autocorrelations. One way to obtain an estimator with reduced variance is simply to omit some of the terms in (14.49). This idea suggests using an expression of the form

$$\hat{S}(\omega) = \sum_{\ell=-(L-1)}^{L-1} \hat{r}_x[\ell] e^{-j\omega\ell}, \tag{14.61}$$

where L is some integer significantly smaller than N. Since $\hat{S}(\omega)$ contains L sample autocorrelations, whereas $I(\omega)$ contains N sample autocorrelations, we would expect that

$$\mathrm{var}[\hat{S}(\omega)] \simeq \frac{L}{N} \mathrm{var}\,[I(\omega)]. \tag{14.62}$$

The bias of (14.61) is given by

$$\mathrm{E}[\hat{S}(\omega)] = \sum_{\ell=-(L-1)}^{L-1} \left(1 - \frac{|\ell|}{N}\right) r_x[\ell] e^{-j\omega\ell}. \tag{14.63}$$

Hence, if we make the value of L depend on N in such a way that (a) $L \to \infty$ as $N \to \infty$, and (b) $(L/N) \to 0$ as $N \to \infty$, then both the bias and variance of $\hat{S}(\omega)$ will asymptotically tend to zero. Therefore, $\hat{S}(\omega)$, for each value of ω, is a consistent estimator of $S(\omega)$. In practice, we can easily satisfy these requirements by choosing L to be some multiple of $\sqrt{N}$. A useful starting point is $L \simeq 2\sqrt{N}$.

The Blackman–Tukey PSD estimator   We note that the estimator $\hat{S}(\omega)$ can be considered as a special case of the more general form of estimator,

$$\hat{S}_{\mathrm{BT}}(\omega) \triangleq \sum_{\ell=-(L-1)}^{L-1} w_c[\ell] \hat{r}_x[\ell] e^{-j\omega\ell}, \tag{14.64}$$

where the correlation or lag window $w_c[\ell]$ is chosen so that sample autocorrelations with higher lags (which are less accurate) are weighted less. This approach to spectral estimation

was introduced by Blackman and Tukey (1958). The lag window must be real nonnegative, symmetric, and nonincreasing with $|\ell|$, that is

$$0 \le w_c[\ell] \le w_c[0] = 1$$
$$w_c[-\ell] = w_c[\ell]$$
$$w_c[\ell] = 0, \; L \le |\ell|. \; L \le N - 1 \tag{14.65}$$

Note that the symmetry property of $w_c[\ell]$ ensures that the estimated PSD is real. The normalization $w_c[0] = 1$ ensures that the area under $\hat{S}_{BT}(\omega)$ is equal to the estimated power $\hat{r}[0]$. Indeed, the inverse Fourier transform of (14.64) for $\ell = 0$ yields

$$w_c[0]\hat{r}[0] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{S}_{BT}(\omega) d\omega, \tag{14.66}$$

which leads to the requirement $w_c[0] = 1$. To understand the effects of the correlation window on the Blackman–Tukey PSD estimator, we express (14.64) in the frequency domain. The result is

$$\hat{S}_{BT}(\omega) = \frac{1}{2\pi} \int_{-\pi}^{\pi} I(\theta) W_c(\omega - \theta) d\theta, \tag{14.67}$$

where $W_c(\omega)$ is the Fourier transform of $w_c[\ell]$. To guarantee that $\hat{S}_{BT}(\omega) \ge 0$, we require that

$$W_c(\omega) \ge 0 \quad \text{for} \; -\pi < \omega \le \pi. \tag{14.68}$$

This condition is satisfied by the Bartlett and Parzen windows. The Parzen window, which is defined by

$$w_P[\ell] = \begin{cases} 1 - 6\left(\frac{|\ell|}{L}\right)^2 + 6\left(\frac{|\ell|}{L}\right)^3, & |\ell| \le \frac{L}{2} \\ 2\left(1 - \frac{|\ell|}{L}\right)^3, & \frac{L}{2} < |\ell| \le L \\ 0, & |\ell| > L \end{cases} \tag{14.69}$$

has a Fourier transform given by (see Basic Problem 30)

$$W_P(\omega) = \frac{3L}{4} \left[\frac{\sin(\omega L/4)}{\sin(\omega/4)}\right]^4. \tag{14.70}$$

Since, for most windows in common use, $W_c(\omega)$ has a dominant narrow peak at $\omega = 0$, the integration in (14.67) produces a weighted average of the values of $I(\omega)$ with the largest weights attached to values in the neighborhood of $\theta = \omega$. This process can be interpreted as filtering the "signal" $I(\omega)$ with a filter having "impulse response" $W_c(\omega)$ to produce an output $\hat{S}_{BT}(\omega)$ (see Figure 14.10). Therefore, weighting the sample autocorrelation in (14.64) so as to reduce the contributions from the "tail" is equivalent to smoothing the periodogram by a weighted integral of the form (14.67) (see Tutorial Problem 8).
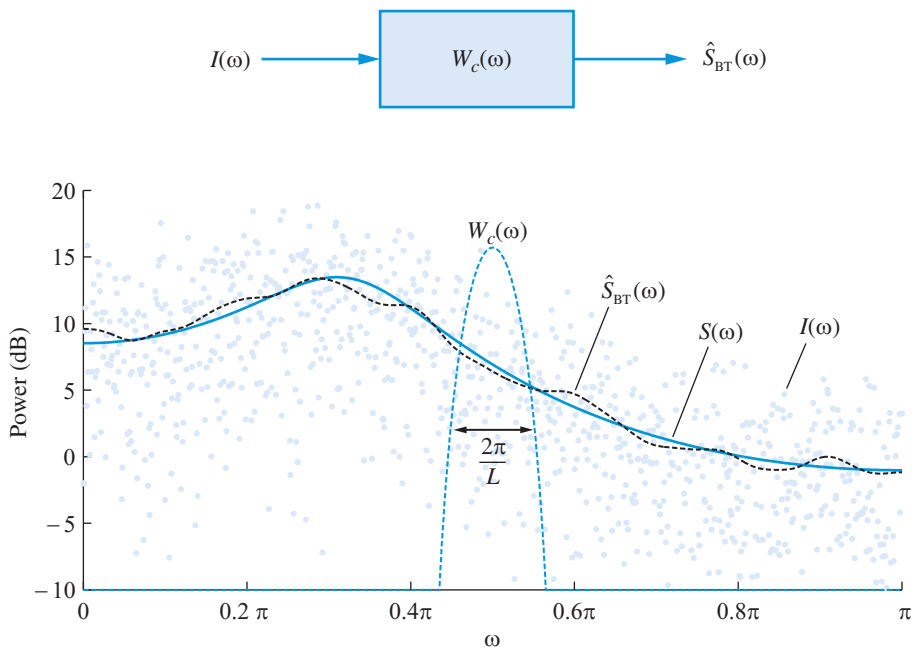
Figure 14.10 Interpretation of the Blackman–Tukey method as a linear filtering operation in the frequency domain. Increasing the width of the mainlobe of $W_c(\omega)$ decreases the variance of the periodogram at the expense of decreasing the resolution. Indeed, no peaks of $S(\omega)$ narrower than the mainlobe of $W_c(\omega)$ can be "seen" in $\hat{S}_{BT}(\omega)$. For most windows the width of the mainlobe is about $2\pi/L$ rads.

Mean of $\hat{S}_{BT}(\omega)$ Taking the expectation of (14.64) and using (14.26) yields

$$E[\hat{S}_{BT}(\omega)] = \sum_{\ell=-(L-1)}^{L-1} \left(1 - \frac{|\ell|}{N}\right) r[\ell] w_c[\ell] e^{-j\omega\ell}, \tag{14.71a}$$

$$= W_B(\omega) \otimes W_c(\omega) \otimes S(\omega), \tag{14.71b}$$

which shows that $\hat{S}_{BT}(\omega)$ is a biased estimator of $S(\omega)$. The lag windows in (14.71a) are normalized so that $w_B[0] = w_c[0] = 1$. Under this condition, if both L and N tend to infinity, then $W_c(\omega)$ and $W_B(\omega)$ become periodic impulse trains and the convolution (14.71b) reproduces the true PSD $S(\omega)$. Thus, $\hat{S}_{BT}(\omega)$ is an asymptotically unbiased estimator of $S(\omega)$. For $N \gg L$, which is typically the case, we see from Figure 14.11 that $w_B[\ell] w_c[\ell] \simeq w_c[\ell]$; thus, the smoothing of $S(\omega)$ is mainly determined by lag window $W_c(\omega)$, which has a wider mainlobe.

In practical spectral estimation, the exact shape of the lag window is relatively unimportant; the crucial choice is the length L of the window $w_c[\ell]$ or equivalently the width $2\pi/L$ of $W_c(\omega)$. Thus, the subject of designing optimal windows for spectral estimation is mainly of theoretical interest because the potential of improvement over the Tukey and Parzen windows appears slight, see Priestley (1981).
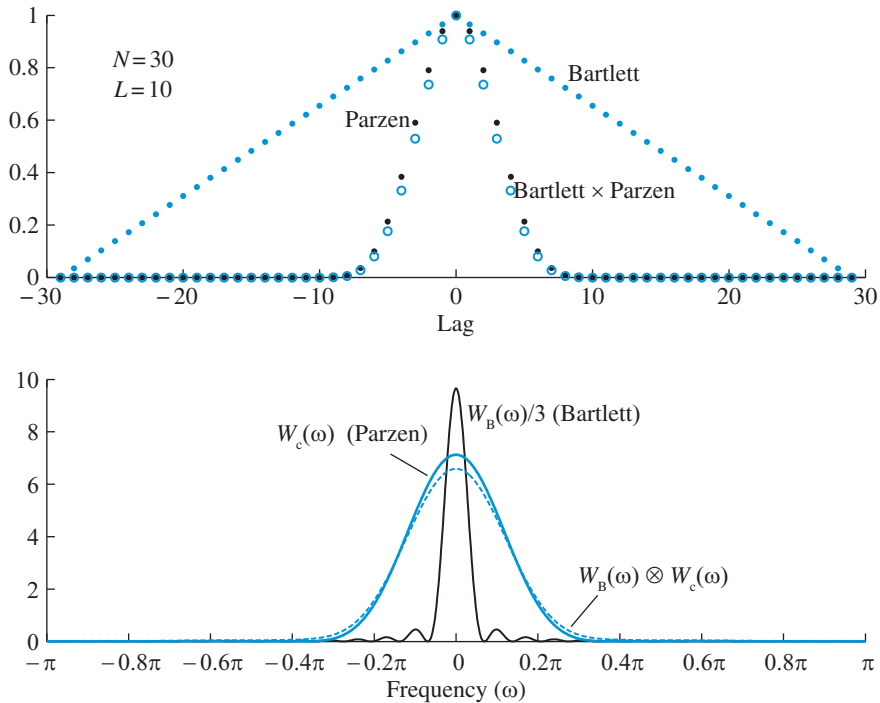
Figure 14.11 The role of windowing in the Blackman–Tukey method of PSD estimation. The finite length of the data segment introduces the Bartlett window; the Parzen window is utilized to smooth the periodogram.

Variance of $\hat{S}_{BT}(\omega)$ It can be shown, see Jenkins and Watts (1968), that approximately

$$\text{var}[\hat{S}_{BT}(\omega)] \simeq \left(\frac{1}{N} \sum_{\ell=-(L-1)}^{L-1} w_c^2[\ell] \right) S^2(\omega), \tag{14.72}$$

again doubled at $\omega = 0, \pi$. From (14.65) we conclude that $\sum_\ell w_c^2[\ell] \leq 2L$; therefore, the variance of $\hat{S}_{BT}(\omega)$ is upper bounded by $(2L/N)S_x^2(\omega)$.

The spectral resolution of the Blackman–Tukey estimator, which determines its bias, is of the order of $2\pi/L$ rads, whereas its variance is of the order of $(L/N)S^2(\omega)$. Therefore, *for a fixed data length N there is a trade-off between bias and variance*. We cannot reduce both bias and variance simultaneously; increasing L to reduce bias leads to an increase in variance, and vice versa. This can be expressed as

$$\text{Bias} \times \text{Variance} = \text{Constant}. \tag{14.73}$$

Computation of $\hat{S}_{BT}(\omega)$ using the DFT In practice, the Blackman–Tukey PSD estimate is computed by using a K-point DFT as follows:

1. Estimate the ACRS $r[\ell]$ using the function `acrs` or `acrsfft` (FFT-based computation is more efficient for $L > 100$; see Tutorial Problem 7):

```
function S=psdbt(x,L,K)
% BT PSD estimator of S(2*pi*k/K)
N=length(x);
w=hann(N); % Data window
w=w/(norm(w)/sqrt(N)); % sum w^2[k]=N
x=x.*w; % Data windowing
r=acrsfft(x,L);
wc=parzenwin(2*L-1); % Lag window
rw=r.*wc(L:2*L-1); % Lag windowing
g=zeros(K,1);
g(1:L)=rw;
g(K-L+2:K)=flipud(rw(2:L));
G=fft(g,K); % f even => F real
S=2*real(G(1:K/2)); S(1)=S(2);
```

Figure 14.12 Computation of Blackman–Tukey PSD estimate.

$$\hat{r}[\ell] = \hat{r}[-\ell] = \frac{1}{N} \sum_{n=0}^{N+\ell-1} x[n+\ell]x[n]. \quad \ell = 0, 1, \ldots, L-1 \tag{14.74}$$

To minimize the effects of leakage we typically use the ACRS of the windowed segment $v[n] = w[n]x[n]$, $0 \le n \le N-1$, where $w[n]$ is a Hann window.

2. Choose a number $K = 2^\kappa > L$ and form the sequence

$$g[\ell] = \begin{cases} \hat{r}[\ell]w_c(\ell), & 0 \le \ell \le L-1 \\ 0, & L \le \ell \le K-L \\ \hat{r}[K-\ell]w_c(K-\ell). & K-L+1 \le \ell \le K-1 \end{cases} \tag{14.75}$$

3. Compute the PSD estimate as the K-point DFT of the sequence $g[\ell]$:

$$\hat{S}_{\text{BT}}\left(\frac{2\pi}{K}k\right) = G[k] = \text{DFT}\{g[\ell]\}. \, k = 0, 1, \ldots, K-1 \tag{14.76}$$

This approach is implemented by the MATLAB function $S=\text{psdbt}(x,L,K)$ shown in Figure 14.12. The values of $\hat{S}_{\text{BT}}(\omega)$ are scaled such that its integral from 0 to π provides an estimate of the process variance.

Example 14.3   Window closing
In many applications we search for periodic components in wideband noise. A natural model for such observations is the harmonic process model (13.151), which for a multiple sinusoids takes the form

$$x[n] = \sum_{k=1}^{p} A_k \cos(\omega_k n + \phi_k) + v[n], \tag{14.77}$$

where A_k and ω_k are constants, ϕ_k are independent random variables uniformly distributed on $(0, 2\pi)$, and $v[n] \sim \text{WGN}(0, \sigma_v^2)$. The random variables ϕ_k and $v[n]$ are assumed mutually uncorrelated. In the absence of noise, for each particular realization the variables ϕ_k are constant; hence each individual realization is essentially a deterministic sequence which can be expressed in the frequency domain using a discrete-time Fourier series. The superposition of the noise term on the right hand side of (14.77) yields a more realistic model by accounting for random observation errors.

The ACRS of the noisy harmonic process (14.77) can be easily shown to be

$$r_x[\ell] = \sum_{k=1}^{p} \frac{A_k^2}{2} \cos(\omega_k \ell) + \sigma_v^2 \delta[\ell]. \tag{14.78}$$

Since $v[n]$ has a purely continuous PSD and $A \cos(\omega_k n)$ has a discrete PSD, we say that the process $x[n]$ has a *mixed* PSD given by

$$S_x(\omega) = 2\pi \sum_{k=1}^{p} \frac{A^2}{4} [\delta(\omega + \omega_k) + \delta(\omega - \omega_k)] + \sigma_v^2. \tag{14.79}$$

Consider now a special case of (14.77) with $A_1 = A_2 = 1$, $A_3 = 1/4$, $\omega_1 = 0.35\pi$, $\omega_2 = 0.40\pi$, $\omega_3 = 0.80\pi$, and $\sigma_v^2 = 1$. We generate a realization of $N = 2000$ samples and we compute the Blackman–Tukey PSD estimate using lag windows of length $L = 50, 100$, and 200 samples. The results shown in Figure 14.13 demonstrate the trade-off between bias and variance. A low value of L will give an idea where the large peaks in $S(\omega)$ are,
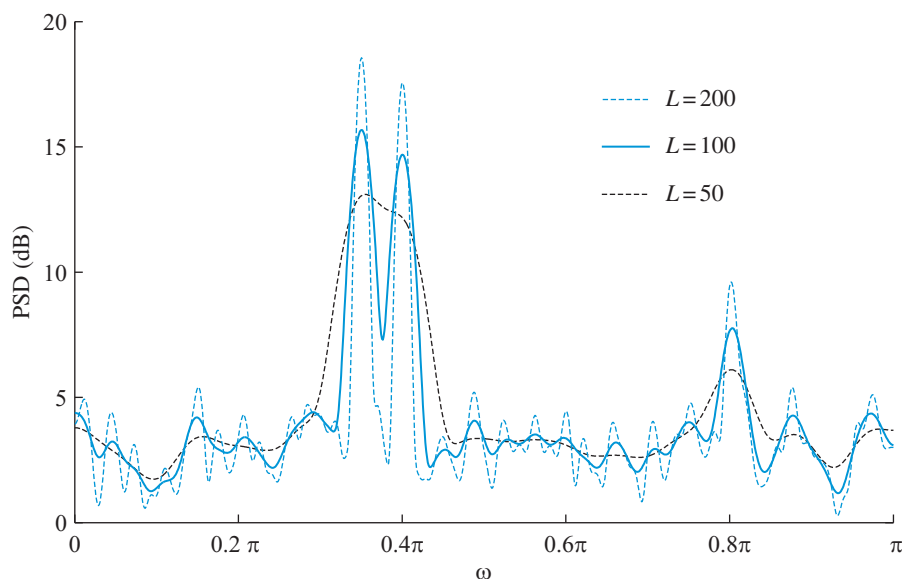
Figure 14.13 Power spectrum estimation of three sinusoids in white noise using the Blackman–Tukey method. Note the trade-off between bias and variance as the length of the lag window L changes.

but the curve is likely to be too smooth. A high value is likely to produce a curve showing a large number of peaks, some of which may be spurious. We usually find a compromise starting with a low value of L (large mainlobe) and increasing L (decreasing the mainlobe) until we do not see significant changes in the estimate. This approach is known as *window closing*, see Jenkins and Watts (1968). In Figure 14.13 the value $L = 100$ appears to provide a good balance between resolution and statistical variability. We note that for the two closely spaced sinusoids to be resolved, the length of the window should satisfy the condition $\omega_2 - \omega_1 > 2\pi/L$ or $L > 40$ samples. We emphasize that this condition is nothing more than a "rule of thumb," which simply states that the resolution is inversely proportional to the length of the data record. More accurate evaluations of L based on more precise definitions of spectral resolution have minor significance in practical applications. See Tutorial Problem 9 for more details. ∎

14.2.6   The Bartlett–Welch method: averaging multiple periodograms

The periodogram $I(\omega)$, for each value of ω, is a random variable. Therefore, we could improve the statistical properties of the periodogram by averaging the values of the periodogram from multiple realizations of a stationary process. Since, in practice, we seldom have segments from multiple realizations of a process, we can only use segments from the same realization.

A widely-used approach introduced by Bartlett (1953) is to subdivide the observed data record into M nonoverlapping segments, find the periodogram of each segment, and finally evaluate the average of the periodograms obtained. If the number of data samples N is equal to ML, where M is the number of segments and L is their length, the mth segment $(1 \leq m \leq M)$ is defined by

$$x_m[n] = x[n + (m-1)L]. \ 0 \leq n \leq L - 1 \tag{14.80}$$

The periodogram of the mth segment is evaluated by

$$I_m(\omega) = \frac{1}{L} \left| \sum_{n=0}^{L-1} x_m[n] e^{-j\omega n} \right|^2. \tag{14.81}$$

The Bartlett estimator is obtained by averaging the M periodograms $I_m(\omega)$, that is,

$$\hat{S}_B(\omega) = \frac{1}{M} \sum_{m=1}^{M} I_m(\omega). \tag{14.82}$$

We emphasize that by definition the estimate $\hat{S}_B(\omega)$ is always nonnegative. To investigate the bias and variance of $\hat{S}_B(\omega)$, we assume that $r[\ell]$ is very small for $|\ell| > L$. This implies that the segments (14.80) can be assumed to be approximately uncorrelated. We also assume that, due to stationarity, all $I_m(\omega)$ are identically distributed. Under these assumptions, the mean value of $\hat{S}_B(\omega)$ is

$$E[\hat{S}_B(\omega)] = \frac{1}{M} \sum_{m=1}^{M} E[I_m(\omega)] = E[I_m(\omega)], \quad \text{for any } m. \tag{14.83}$$

The mean value of $I_m(\omega)$ is given by (14.40) and (14.44) with N replaced by L. Since $L = N/M$, the bias of $\hat{S}_B(\omega)$ will approximately increase by a factor of M compared to the bias of $I(\omega)$, the periodogram of the entire set of N observations.

The variance of the average periodogram $\hat{S}_B(\omega)$ is given by (see (14.50))

$$\text{var}[\hat{S}_B(\omega)] = \frac{1}{M} \sum_{m=1}^{M} \text{var}[I_m(\omega)] \simeq \frac{1}{M} S^2(\omega), \tag{14.84}$$

because for each ω, $I_m(\omega)$, $1 \le m \le M$ are independent and identically distributed random variables. Clearly, as M increases, the variance tends to zero. Thus, $\hat{S}_B(\omega)$ provides an asymptotically unbiased and consistent estimate of $S(\omega)$. If N is fixed and $N = ML$, we see that increasing M to reduce the variance (or equivalently obtain a smoother estimate) results in a decrease in L, that is, a reduction in resolution (or equivalently an increase in bias). Thus, *for a fixed data length there is a trade-off between bias and variance*. However, if the data length N increases, both L and M can be allowed to increase, so that as $N \to \infty$, the bias and variance of the average periodogram can approach zero. Therefore, $\hat{S}_B(\omega)$ is an asymptotically unbiased and consistent estimator of $S(\omega)$.

Welch's method The method of Welch (1969) extends the Bartlett approach by overlapping the segments and by windowing each segment. The purpose of using a *data window* is to reduce the spectral leakage caused by strong narrowband components by lowering the level of sidelobes. Overlapping the segments yields some extra variance reduction due to the increased number of averaged periodograms. The M (possibly overlapping) segments are defined by

$$x_m[n] = x[n + (m-1)D], \; 0 \le n \le L - 1 \tag{14.85}$$

where D is an *offset* distance. If $D < L$, the segments overlap; and for $D = L$, the segments are contiguous. The modified periodograms are given by

$$\tilde{I}_m(\omega) \triangleq \frac{1}{L} \left| \sum_{n=0}^{L-1} w[n] x_m[n] e^{-j\omega n} \right|^2, \tag{14.86}$$

where $w[n]$ is a data window of duration L, which does not have to be symmetric. The Welch estimator is given by the average of the M modified periodograms

$$\hat{S}_W(\omega) = \frac{1}{M} \sum_{m=1}^{M} \tilde{I}_m(\omega). \tag{14.87}$$

The data window is normalized so that $\sum_{n=0}^{L-1} w^2(n) = L$ which assures that the periodogram of each segment $x_m[n]$ is asymptotically unbiased (see the discussion following equation (14.59)).

Welch showed that the shape of the window may reduce leakage but does not affect the variance formula (14.84). However, *overlapping the segments by fifty percent reduces the variance by about a factor of two*, owing to doubling the number of segments. More overlap does not result in additional reduction of variance because the data segments become less

```
function S=psdwelch(x,L,K)
% Welch PSD estimator of S(2*pi*k/K)
M=fix((length(x)-L/2)/(L/2)) % 50% overlap
time=(1:L)';
I=zeros(K,1);
w=hanning(L); % Choose window
w=w/(norm(w)/sqrt(L)); % sum w^2[k]=L
for m=1:M
   %xw=w.*detrend(x(time)); % detrenting
   xw=w.*x(time); % data windowing
   X=fft(xw,K);
   I=I+X.*conj(X);
   time=time+L/2;
end
I=I/(M*L); S=2*I(1:K/2); S(1)=S(2);
```

Figure 14.14 Computation of Welch PSD estimate.

and less independent. Clearly, nonoverlapping segments can be totally uncorrelated only for white noise processes. However, the data segments can be considered approximately uncorrelated if they do not have sharp spectral peaks or if their autocorrelation sequence decays fast.

Computation In practice, we can compute $\hat{S}_W(\omega)$ at K equally spaced frequencies $\omega_k = 2\pi k/K, 0 \le k \le K - 1$, easily and efficiently by using the DFT. A straightforward implementation of Welch's PSD estimator is provided by the MATLAB function `psdwelch` shown in Figure 14.14.

In the signal processing toolbox the Welch method is implemented by

$$[Sw,omega] = pwelch(x,window(L),Noverlap,Nfft,Fs),$$

where `window` is the name of any MATLAB-provided window function (for example, `hann`); `Nfft` is the size of the DFT, which is chosen to be larger than `L` to obtain a high-density spectrum; `Fs` is the sampling frequency, which is used for plotting purposes; and `Noverlap` specifies the number of overlapping samples. If the `rectwin` window is used along with `Noverlap=0`, then we obtain Bartlett's method of periodogram averaging. (Note that `Noverlap` is different from the offset parameter D given above.) If `Noverlap=L/2` is used, then we obtain Welch's averaged periodogram method with 50 percent overlap.

Example 14.4   Welch's method for the AR(4) process
Consider Welch's PSD estimation method for the AR(4) process introduced in Example 14.2 for $N = 2000$, 50 percent overlap, and a Hanning window. Three different values for the overlapping segment length L were considered; $L = 50$ ($M = 79$ segments), $L = 100$
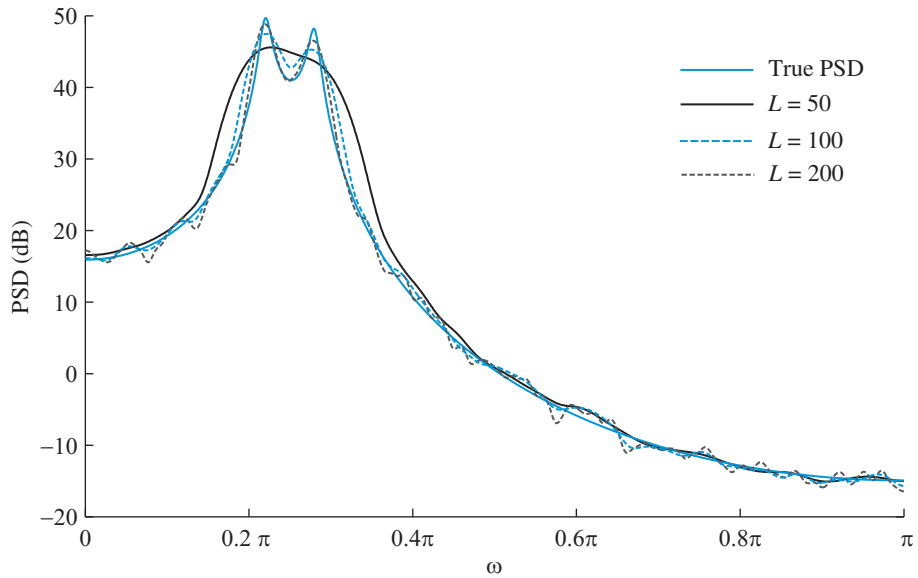
Figure 14.15 Power spectrum estimation of an AR(4) process using the Welch method of averaged periodograms. Note the trade-off between bias and variance as the length L of the overlapping segments changes.

($M = 39$ segments), and $L = 200$ ($M = 19$ segments). The results shown in Figure 14.15 clearly demonstrate the trade-off between bias and variance (for more details see Tutorial Problem 10). We emphasize that the choice of segment length L is a very challenging task and requires a good understanding of the underlying theory and sufficient practical experience.   ■

14.3   Optimum linear filters

Relationship (13.114) in Section 13.4.4 shows that we can use the techniques developed in Chapter 10 and Chapter 11 to design filters that can separate random signals with non-overlapping power spectral densities. However, in many applications, a signal of interest (either deterministic or random) is corrupted by unwanted noise or interference of random origin at the same frequency band. Such applications can be better served with linear filters designed using statistical criteria of optimality. In this section, we discuss the design of FIR systems that (a) maximize the output signal-to-noise ratio at a specified time, when the form of the signal is known, or (b) minimize the mean square error.

14.3.1   Filters that maximize the output signal-to-noise ratio

Many radar and digital communications systems involve the transmission of a deterministic signal of known form in noise. For example, a radar system operates by transmitting a finite

Figure 14.16 Principle of operation of a radar system.
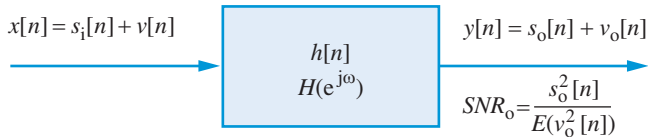
Figure 14.17 Input and output signals in a matched filter.

duration electromagnetic signal $s(t)$. If an object happens to be in the way then part of the signal is reflected and finds its way back to the radar (see Figure 14.16). In this case, the signal $x(t)$ measured by the radar receiver is the sum of the reflected signal $s_i(t) = as(t-t_0)$, where a is an attenuation factor and t_0 the round-trip delay, and the unwanted random noise $v(t)$. The other possibility is that there is no object in the way of the transmitted pulse, that is, the received signal is just noise. In discrete-time, these two possibilities can be stated as

$$x[n] = \begin{cases} s_i[n] + v[n], & \text{when the signal is present} \\ v[n], & \text{when the signal is absent} \end{cases} \tag{14.88}$$

where $s[n]$, $0 \le n \le p-1$ is a finite duration signal with fixed known shape, $s_i[n] = as[n-D]$ is the reflected signal, and $v[n]$ is a realization of a zero-mean wide-sense stationary noise process with variance σ_v^2 and ACRS $r_v[\ell]$. The parameter D is the round-trip delay in number of sampling intervals. The received signal is passed through a filter with impulse response $h[n]$ to yield an output $y[n]$ (see Figure 14.17). We wish to determine an FIR filter

$$y[n] = \sum_{k=0}^{p-1} h[k]x[n-k], \tag{14.89}$$

that will help, as much as possible, to decide which of the two possibilities, namely signal plus noise or just noise alone, actually occurs at a given instant.

To decide whether a signal plus noise or just noise alone is present at a certain instant of time, say time $n = n_0$, we require the output of the filter at that time to be greater when the signal is present than if it were absent. In other words, we wish to make decisions by recognizing peaks in the output signal $y[n]$. This objective can be achieved by maximizing the *output* signal-to-noise ratio (SNR), defined by

$$\text{SNR}_o = \frac{(\text{Value of filtered signal at } n = n_0)^2}{\text{Power of filtered noise}} = \frac{s_o^2[n_0]}{E(v_o^2[n_0])}, \tag{14.90}$$

at the decision time $n = n_0$. If we substitute the "signal present" case of (14.88) into (14.89) and set $n_0 = p + D - 1$, the output signal can be written as

$$y[n_0] = a\boldsymbol{h}^\mathsf{T}\boldsymbol{s} + \boldsymbol{h}^\mathsf{T}\boldsymbol{v}[n_0], \tag{14.91}$$

where

$$\boldsymbol{h} \triangleq \begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[p-1] \end{bmatrix}, \quad \boldsymbol{s} \triangleq \begin{bmatrix} s[p-1] \\ s[p-2] \\ \vdots \\ s[0] \end{bmatrix}, \quad \boldsymbol{v}[n_0] \triangleq \begin{bmatrix} v[P+D-1] \\ v[P+D-2] \\ \vdots \\ v[D] \end{bmatrix}. \tag{14.92}$$

The power of output signal $s_o[n] = a\boldsymbol{h}^\mathsf{T}\boldsymbol{s}$ at $n = n_0$ is $s_o^2[n_0] = a^2(\boldsymbol{h}^\mathsf{T}\boldsymbol{s})^2$. The average power of the output noise $v_o[n] = \boldsymbol{h}^\mathsf{T}\boldsymbol{v}[n_0]$ is given by the quadratic form

$$\mathrm{E}(v_o^2[n]) = \boldsymbol{h}^\mathsf{T}\boldsymbol{R}_v\boldsymbol{h}, \tag{14.93}$$

which follows from (13.47). Since $v[n]$ is wide-sense stationary with zero mean value, the noise correlation matrix $\boldsymbol{R}_v$ is symmetric and Toeplitz. The output SNR (14.90) is given by the formula

$$\mathrm{SNR_o} = \frac{s_o^2[n_0]}{\mathrm{E}(v_o^2[n_0])} = a^2 \frac{(\boldsymbol{h}^\mathsf{T}\boldsymbol{s})^2}{\boldsymbol{h}^\mathsf{T}\boldsymbol{R}_v\boldsymbol{h}}. \tag{14.94}$$

Our objective is to find the impulse response $\boldsymbol{h}$ so that the above ratio is maximum. We find the solution, known as a *matched filter*, using the Cauchy–Schwartz inequality

$$(\boldsymbol{a}^\mathsf{T}\boldsymbol{b})^2 \leq (\boldsymbol{a}^\mathsf{T}\boldsymbol{a})(\boldsymbol{b}^\mathsf{T}\boldsymbol{b}). \tag{14.95}$$

The equality holds if $\boldsymbol{a} = \kappa\boldsymbol{b}$, for any constant κ. To proceed, we note that every symmetric matrix $\boldsymbol{R}_v$ can be decomposed as $\boldsymbol{R}_v = \boldsymbol{R}_v^{1/2}\boldsymbol{R}_v^{1/2}$, where $\boldsymbol{R}_v^{1/2}$ is a symmetric matrix known as the square-root of $\boldsymbol{R}_v$ (see Strang 2006). Then, the output SNR (14.94) can be expressed as

$$\mathrm{SNR_o} = a^2 \frac{(\boldsymbol{h}^\mathsf{T}\boldsymbol{R}_v^{1/2}\boldsymbol{R}_v^{-1/2}\boldsymbol{s})^2}{\boldsymbol{h}^\mathsf{T}\boldsymbol{R}_v^{1/2}\boldsymbol{R}_v^{1/2}\boldsymbol{h}} = a^2 \frac{(\tilde{\boldsymbol{h}}^\mathsf{T}\tilde{\boldsymbol{s}})^2}{\tilde{\boldsymbol{h}}^\mathsf{T}\tilde{\boldsymbol{h}}} \leq a^2\tilde{\boldsymbol{s}}^\mathsf{T}\tilde{\boldsymbol{s}}, \tag{14.96}$$

where $\tilde{\boldsymbol{h}} \triangleq \boldsymbol{R}_v^{1/2}\boldsymbol{h}$ and $\tilde{\boldsymbol{s}} \triangleq \boldsymbol{R}_v^{-1/2}\boldsymbol{s}$. The last inequality in (14.96) follows from (14.95). The output SNR attains its maximum value $a^2\tilde{\boldsymbol{s}}^\mathsf{T}\tilde{\boldsymbol{s}}$ by choosing $\tilde{\boldsymbol{h}} = \kappa\tilde{\boldsymbol{s}}$ or $\boldsymbol{R}_v^{1/2}\boldsymbol{h} = \kappa\boldsymbol{R}_v^{-1/2}\boldsymbol{s}$. Multiplying both sides of the last equation by $\boldsymbol{R}_v^{-1/2}$ yields

$$\boldsymbol{h} = \kappa\boldsymbol{R}_v^{-1}\boldsymbol{s}. \tag{14.97}$$

The maximum possible value of the output SNR is given by

$$\text{SNR}_o = a^2 \tilde{s}^T \tilde{s} = a^2 s^T R_v^{-1} s. \tag{14.98}$$

The design of the matched filter requires the values $r_v[0], r_v[1], \ldots, r_v[p-1]$ of the noise ACRS and the shape s of the transmitted signal. The maximum SNR (14.98) is obtained for any choice of constant κ; however, we usually choose κ by requiring that (a) $h^T s = 1$, which yields $\kappa = 1/s^T R_v^{-1} s$, or (b) $\text{E}(v_0^2[n]) = h^T R_v h = 1$, which yields $\kappa = 1/\sqrt{s^T R_v^{-1} s}$.

To better understand the meaning of (14.97) and (14.98) we consider a white noise process with correlation matrix $R_v = \sigma_v^2 I$. Then (14.98) and (14.99) are simplified as follows:

$$h_w = \frac{\kappa}{\sigma_v^2} s, \quad \text{SNR}_w = \frac{a^2}{\sigma_v^2} \sum_{k=0}^{p-1} s^2[k] \triangleq a^2 \frac{E_s}{\sigma_v^2}. \tag{14.99}$$

We note that the best performance, as measured by the output SNR, depends on the received signal energy and the noise power. For a given pulse shape s and attenuation a, the signal energy E_s is maximized if all samples of the transmitted pulse are used by the filter. This explains the maximization of output SNR at time $n_0 = p + D - 1$. These issues are illustrated in Figure 14.18; for more details see Problem 37 and Johnson and Dudgeon (1993). In practice, finding the time delay D depends on the particular application. The term matched filter was introduced because, in the case of white noise, the impulse response is "matched" to the shape of the signal being sought. In fact, the output of the matched filter is proportional to the correlation between the signal segment stored in the filter memory and the signal of interest (see Problem 35).

14.3.2   Filters that minimize the output mean square error

For pedagogical reasons we start with the following problem: estimate or "guess" the value of a random variable y from the observations of a related set of random variables $x_1, x_2, \ldots, x_p$ using the linear estimator

$$\hat{y} = \sum_{k=1}^{p} h_k x_k = h^T x. \tag{14.100}$$

The error between the true and estimated values is given by

$$e \triangleq y - \hat{y} = y - \sum_{k=1}^{p} h_k x_k = y - h^T x. \tag{14.101}$$

We wish to determine the coefficients $h_1, h_2, \ldots, h_p$ that minimize the mse $\text{E}(e^2)$. Using (14.100) we can easily show that
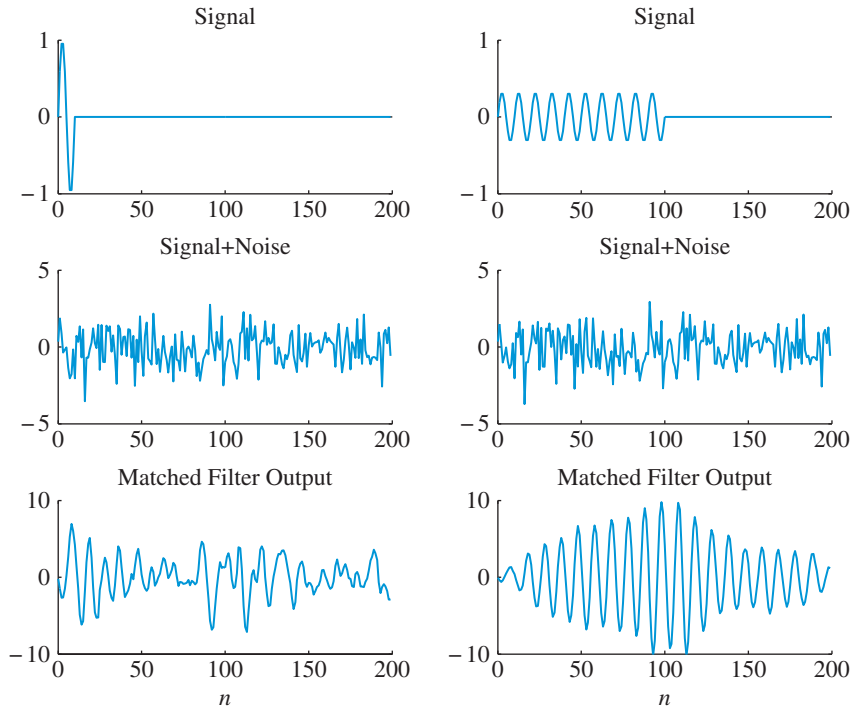
Figure 14.18 The operation of a matched filter in white noise. The two signals have different length ($p = 10$ and $p = 100$, respectively) but the same energy. We note that the peak response of the matched filter occurs at $n_0 = p - 1$ because there is no time delay ($D = 0$).

$$e^2 = \left(y - \sum_{k=1}^{p} h_k x_k \right) \left(y - \sum_{m=1}^{p} h_m x_m \right), \tag{14.102}$$

$$= y^2 - 2 \sum_{k=1}^{p} h_k x_k y + \sum_{k=1}^{p} \sum_{m=1}^{p} h_k h_m x_k x_m. \tag{14.103}$$

Taking the expectation of both sides of (14.103) yields

$$\mathrm{E}(e^2) = \mathrm{E}(y^2) - 2 \sum_{k=1}^{p} h_k \mathrm{E}(x_k y) + \sum_{k=1}^{p} \sum_{m=1}^{p} h_k h_m \mathrm{E}(x_k x_m). \tag{14.104}$$

If we define the correlation matrix $\boldsymbol{R}_x$ of random vector $\boldsymbol{x}$ and the cross-correlation vector $\boldsymbol{g}$ between $\boldsymbol{x}$ and y by

$$\boldsymbol{R}_x \triangleq \begin{bmatrix} \mathrm{E}(x_1 x_1) & \mathrm{E}(x_1 x_2) & \cdots & \mathrm{E}(x_1 x_p) \\ \mathrm{E}(x_2 x_1) & \mathrm{E}(x_2 x_2) & \cdots & \mathrm{E}(x_2 x_p) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{E}(x_p x_1) & \mathrm{E}(x_p x_2) & \cdots & \mathrm{E}(x_p x_p) \end{bmatrix} \text{ and } \boldsymbol{g} \triangleq \begin{bmatrix} \mathrm{E}(x_1 y) \\ \mathrm{E}(x_2 y) \\ \vdots \\ \mathrm{E}(x_p y) \end{bmatrix}, \tag{14.105}$$

we can express (14.103) in matrix form as follows:

$$J(\boldsymbol{h}) \triangleq \mathrm{E}(e^2) = \mathrm{E}(y^2) - 2\boldsymbol{h}^\mathrm{T}\boldsymbol{g} + \boldsymbol{h}^\mathrm{T}\boldsymbol{R}_x\boldsymbol{h}. \tag{14.106}$$

From (13.47) and (14.100) we find that $\mathrm{E}(\hat{y}^2) = \boldsymbol{h}^\mathrm{T}\boldsymbol{R}_x\boldsymbol{h}$. Since $\mathrm{E}(\hat{y}^2) \geq 0$ we conclude that $\boldsymbol{h}^\mathrm{T}\boldsymbol{R}_x\boldsymbol{h} \geq 0$, that is, the correlation matrix is always nonnegative definite.

Setting the partial derivatives of $\mathrm{E}(e^2)$ with respect to each coefficient $h_1, h_2, \ldots, h_p$ equal to zero, we obtain

$$\frac{\partial \mathrm{E}(e^2)}{\partial h_i} = \mathrm{E}\left(\frac{\partial e^2}{\partial h_i}\right) = \mathrm{E}\left(2e\frac{\partial e}{\partial h_i}\right) = -2\mathrm{E}(ex_i) = 0. \; 1 \leq i \leq p \tag{14.107}$$

This yields a set of simultaneous equations that specifies the optimum value for each coefficient. Indeed, using (14.101) and (14.107) we obtain the so-called *normal equations*

$$\mathrm{E}(x_1x_1)h_1 + \mathrm{E}(x_1x_2)h_2 + \cdots + \mathrm{E}(x_1x_p)h_p = \mathrm{E}(x_1y),$$
$$\mathrm{E}(x_2x_1)h_1 + \mathrm{E}(x_2x_2)h_2 + \cdots + \mathrm{E}(x_2x_p)h_p = \mathrm{E}(x_2y),$$
$$\vdots$$
$$\mathrm{E}(x_px_1)h_1 + \mathrm{E}(x_px_2)h_2 + \cdots + \mathrm{E}(x_px_p)h_p = \mathrm{E}(x_py),$$

which can be written in matrix form as follows:

$$\boldsymbol{R}_x\boldsymbol{h} = \boldsymbol{g}. \tag{14.108}$$

The solution of (14.108) provides the optimum mse estimator

$$\boldsymbol{h}_\mathrm{o} = \boldsymbol{R}_x^{-1}\boldsymbol{g}, \tag{14.109}$$

provided that the correlation matrix is invertible. The minimum mse, found by substituting (14.109) into (14.106), is given by

$$J_\mathrm{o} \triangleq J(\boldsymbol{h}_\mathrm{o}) = \mathrm{E}(y^2) - \boldsymbol{g}^\mathrm{T}\boldsymbol{R}_x^{-1}\boldsymbol{g}. \tag{14.110}$$

To verify that (14.110) is indeed the optimum solution, suppose that $\boldsymbol{h} = \boldsymbol{h}_\mathrm{o} + \boldsymbol{\delta}$. Then using (14.106) and (14.110) we obtain

$$J(\boldsymbol{h}) = J(\boldsymbol{h}_\mathrm{o}) + \boldsymbol{\delta}^\mathrm{T}\boldsymbol{R}_x\boldsymbol{\delta}. \tag{14.111}$$

If $\boldsymbol{R}_x$ is positive definite, which is the case for most well-behaved problems, then for any $\boldsymbol{\delta} \neq \boldsymbol{0}$, the quadratic form $\boldsymbol{\delta}^\mathrm{T}\boldsymbol{R}_x\boldsymbol{\delta} > 0$. Since $J(\boldsymbol{h}_0 + \boldsymbol{\delta}) > J(\boldsymbol{h}_0)$ for all $\boldsymbol{\delta} \neq \boldsymbol{0}$, $J(\boldsymbol{h}_0)$ is
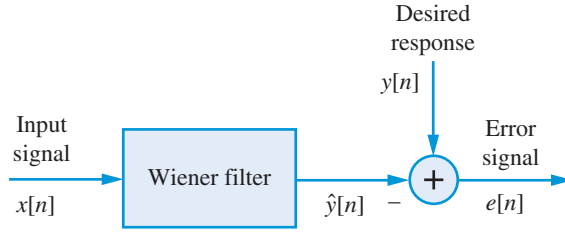
Figure 14.19 Representation of the Wiener filtering problem.

the minimum mse. Therefore, any deviation from the optimum results in an *excess mse* $\delta^T R_x \delta$ which depends *only* on the correlation matrix of the input random variables and the distance δ between the actual vector h and the optimum vector h_o. Thus, the solution h_o of the normal equations (14.108) provides the *optimum linear mse estimator* provided that the correlation matrix R_x of the input random variables is positive definite. The conditions $E(ex_i) = 0$ imply that the optimum estimation error is orthogonal to every random variable $x_1, x_2, \ldots, x_p$ used for the estimation. For this reason (14.107) is known as the *orthogonality principle* and (14.108) as the *normal equations*.

We next discuss an important special case of linear mse estimation which is widely used in many signal processing applications.

Optimum FIR filtering Suppose that we wish to estimate the value of a random process $y[n]$ using observations from a related process $x[n]$ with the FIR filter

$$\hat{y}[n] = \sum_{k=1}^{p} h_k x[n+1-k] \tag{14.112}$$

by minimizing the mse, $E(e^2[n])$, between the *desired response* $y[n]$ and the *actual response* $\hat{y}[n]$ of the filter (see Figure 14.19). Comparison of (14.112) to (14.100) shows that $y = y[n]$ and $x_k = x[n+1-k]$, $1 \le k \le p$. If we assume that the processes $x[n]$ and $y[n]$ are jointly wide-sense stationary, we have $E(x_k x_m) = E(x[n+1-k]x[n+1-m]) = r[|m-k|]$ and $E(x_k y) = E(x[n+1-k]y[n]) = r_{xy}[1-k] = r_{yx}[k-1]$. Therefore, the normal equations (14.108) take the form

$$\begin{bmatrix} r_x[0] & r_x[1] & \cdots & r_x[p-1] \\ r_x[1] & r_x[0] & \cdots & r_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_x[p-1] & r_x[p-2] & \cdots & r_x[0] \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_p \end{bmatrix} = \begin{bmatrix} r_{yx}[0] \\ r_{yx}[1] \\ \vdots \\ r_{yx}[p-1] \end{bmatrix}, \tag{14.113}$$

or $R_x h_o = g$, where the matrix R_x is symmetric Toeplitz. Since R_x and g do *not* depend on the time index n, the optimum filter coefficients h_o do not depend on n as well. Therefore, for stationary processes the optimum filter is time-invariant. If we set $h_o[k-1] = h_k$, $1 \le k \le p$, the normal equations can be written as

$$\sum_{k=0}^{p-1} h_o[k] r_x[m-k] = r_{yx}[m], \quad 0 \le m \le p-1. \tag{14.114}$$

The minimum value of mse $E(e^2[n])$, where $e[n] = y[n] - \hat{y}[n]$, is given by

$$J_o = r_y[0] - \mathbf{h}_o^T \mathbf{g} = r_y[0] - \sum_{k=0}^{p-1} h_o[k] r_{yx}[k]. \tag{14.115}$$

The filter defined by (14.113) is known as a *Wiener filter* because it is the discrete-time equivalent of the continuous-time optimum filter introduced by Norbert Wiener (see Bode and Shannon 1950). The most important feature of the Wiener filter is that its design does not depend upon the particular realizations $x[n]$ and $y[n]$ of the input and desired response stochastic processes, but upon the first p values of the correlation sequences $r_x[m]$ and $r_{yx}[m]$. Therefore, the same Wiener filter can be used for all random signals with the same second-order moments.

Example 14.5 Extraction of signal from noise

Suppose that we wish to design a Wiener filter for the extraction of a useful signal $y[n]$ from observations of the noise distorted signal

$$x[n] = y[n] + v[n]. \tag{14.116}$$

If we assume that (a) the noise process $v[n]$ is uncorrelated with the desired process $y[n]$, and (b) that $x[n]$ and $v[n]$ are wide-sense stationary, we obtain

$$r_x[m] = r_y[m] + r_v[m] \qquad \Rightarrow \mathbf{R}_x = \mathbf{R}_y + \mathbf{R}_v, \tag{14.117}$$

$$r_{yx}[m] = r_y[m] \qquad \Rightarrow \mathbf{g} = [r_y[0] \ r_y[1] \ \ldots \ r_y[p-1]]^T. \tag{14.118}$$

The optimum filter is found by solving (14.108) after substituting $\mathbf{R}_x$ and $\mathbf{g}$ from (14.117) and (14.118). To understand the operation of the optimum filter, we assume that the indices k and m in (14.114) take on an infinite range, that is

$$\sum_{k=-\infty}^{\infty} h_o[k] r_x[m-k] = r_{yx}[m]. \quad -\infty \le m \le \infty \tag{14.119}$$

Since the left hand side is the convolution between $h_o[m]$ and $r_x[m]$, we have that $h_o[m] * r_x[m] = r_{yx}[m]$. This is a convolution equation which can be solved *analytically* using transform techniques. Indeed, taking the Fourier transform of both sides and solving for the frequency response $H_o(\omega)$ of the optimum filter yields

$$H_o(\omega) = \frac{S_{yx}(\omega)}{S_x(\omega)} = \frac{S_y(\omega)}{S_y(\omega) + S_v(\omega)}. \tag{14.120}$$

The last relation follows by using the Fourier transform of (14.119) and (14.117). We note that for frequency bands where $S_y(\omega) \gg S_v(\omega)$, that is for bands with large SNR, we have $H_o(\omega) \approx 1$. In contrast, if $S_y(\omega) \ll S_v(\omega)$, that is for bands with low SNR, we have $H_o(\omega) \approx 0$. Thus, the optimum filter "passes" the input signal at bands with high SNR and "blocks" the input at bands with low SNR, as we would expect intuitively. ∎

Using Parseval's theorem (see Tutorial Problem 16) we can express the minimum mse in the frequency domain as follows:

$$J_o = r_y[0] - \sum_{k=-\infty}^{\infty} h_o[k]r_{yx}[k] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left[1 - \frac{|S_{yx}(\omega)|^2}{S_y(\omega)S_x(\omega)} \right] S_y(\omega)d\omega. \qquad (14.121)$$

The quantity $C(\omega) \triangleq |S_{yx}(\omega)|^2/S_y(\omega)S_x(\omega)$, which is called the *magnitude square coherence*, can be thought of as a sort of correlation coefficient in the frequency domain. Since $0 \le C(\omega) \le 1$, the optimum filter reduces the mse in bands with high coherence between the input and desired response processes (see Tutorial Problem 16 for further explanations).

14.4 Linear prediction and all-pole signal modeling

Linear prediction is a special case of Wiener filtering which finds applications in a wide range of areas, including speech processing, spectrum estimation, and image processing. In this section, we provide an introduction to linear prediction with emphasis on its application in parametric signal modeling.

14.4.1 Linear prediction and AR modeling

Suppose that we know the values of p consecutive samples of a wide-sense stationary process, say $x[n-1], x[n-2], \ldots, x[n-p]$, and we wish to estimate the value of the "future" sample $x[n]$ using the linear estimator

$$\hat{x}[n] = \sum_{k=1}^{p} h_k x[n-k] = \boldsymbol{h}^{\mathrm{T}} \boldsymbol{x}[n-1], \qquad (14.122)$$

where $\boldsymbol{h} \triangleq [h_1 \ h_2 \ \ldots \ h_p]^{\mathrm{T}}$ and $\boldsymbol{x}[n-1] \triangleq [x[n-1] \ x[n-2] \ \ldots \ x[n-p]]^{\mathrm{T}}$. The optimum coefficient vector $\boldsymbol{h}$ is obtained by minimizing the mse

$$J = \mathrm{E}(e^2[n]) = \mathrm{E}\{(x[n] - \hat{x}[n])^2\}. \qquad (14.123)$$

To this end, we note that (14.122) is a special case of (14.100) with $x_1 = x[n-1], x_2 = x[n-2], \ldots, x_p = x[n-p]$ and $y = x[n]$. Thus, the optimum one-step *forward linear predictor* given the finite past is given by (14.108). Since the process $x[n]$ is wide-sense stationary, the elements of matrix $\boldsymbol{R}_x$ and vector $\boldsymbol{g}$ are given by (for $k, m = 1, 2, \ldots, p$)

$$\mathrm{E}(x_k x_m) = \mathrm{E}(x[n-k]x[n-m]) = r(|k-m|), \qquad (14.124a)$$

$$g_k = \mathrm{E}(x_k y) = \mathrm{E}(x[n-k]x[n]) = r[k], \qquad (14.124b)$$

where for simplicity, we have dropped the subscript "x" from the ACRS $r_x[\ell]$. Thus, the normal equations for the optimum linear predictor are

$$Rh = r, \tag{14.125}$$

where R, which is a symmetric Toeplitz matrix, and the vector r are defined by

$$R \triangleq \begin{bmatrix} r[0] & r[1] & \cdots & r[p-1] \\ r[1] & r[0] & \cdots & r[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r[p-1] & r[p-2] & \cdots & r[0] \end{bmatrix} \quad \text{and} \quad r \triangleq \begin{bmatrix} r[1] \\ r[2] \\ \vdots \\ r[p] \end{bmatrix}. \tag{14.126}$$

From (14.110) and (14.125) it follows that the minimum mse is given by

$$J_0 = r[0] - h^{\mathsf{T}}r = r[0] - r^{\mathsf{T}}R^{-1}r. \tag{14.127}$$

We note from (14.124)–(14.126) that the optimum linear predictor is specified by the first $p+1$ samples of the ACRS of the process $x[n]$.

Suppose now that $x[n]$ is an AR(p) process specified by (see Section 13.5.3)

$$x[n] = -\sum_{k=1}^{p} a_k x[n-k] + z[n] = -a^{\mathsf{T}}x[n-1] + z[n], \tag{14.128}$$

where $z[n] \sim \text{WN}(0, \sigma_z^2)$. The coefficients $a_1, a_2, \ldots, a_p$ are specified by the following set of Yule–Walker equations:

$$Ra = -r, \tag{14.129}$$

whereas the variance of the input noise process is given by

$$\sigma_z^2 = r[0] + a^{\mathsf{T}}r = r[0] - r^{\mathsf{T}}R^{-1}r. \tag{14.130}$$

The actual value $x[n]$ of the sample to be predicted by (14.122) can be written as

$$x[n] = \sum_{k=1}^{p} h_k x[n-k] + e[n] = h^{\mathsf{T}}x[n-1] + e[n]. \tag{14.131}$$

Careful inspection of (14.128)–(14.131) shows that, if we set $h = -a$, the pth order optimum linear predictor and the AR(p) model of a wide-sense stationary process with zero mean value are described by the same set of equations. Adopting this convention leads to a unified presentation of linear prediction and AR modeling.

With the new notation, the prediction error can be expressed as

$$e[n] = x[n] + \sum_{k=1}^{p} a_k x[n-k] = x[n] + a^{\mathsf{T}}x[n-1], \tag{14.132}$$

which shows that sequence $e[n]$ is the output of a filter with system function

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}. \tag{14.133}$$

The system $A(z)$ is known as the *prediction error filter* or *analysis filter*. If the input $x[n]$ is an AR(p) process, the output of $A(z)$ is a white noise process $z[n]$; thus, $A(z)$ is often referred to as a *whitening filter*. The system $H(z) = 1/A(z)$ is also called *coloring filter* because it shapes the spectrum of the white noise process $z[n]$ to produce the color process $x[n]$.

14.4.2   The Levinson–Durbin algorithm

Since R is symmetric and positive definite, the Yule–Walker equations (14.129) can be solved using the Cholesky decomposition with a complexity of $O(p^3)$ computations. However, because of the Toeplitz structure and the special nature of the right hand side, we can solve (14.129) recursively, with a complexity of $O(p^2)$ operations, using the algorithm of Levinson–Durbin.

The development of the Levinson–Durbin algorithm requires the use of order-specific notation. Thus, the mth-order optimum linear predictor is specified by

$$e_m[n] = x[n] + a_m^T x_m[n-1], \tag{14.134a}$$

$$a_m = -R_m^{-1} r_m, \tag{14.134b}$$

$$J_m = r[0] + a_m^T r_m = r[0] - r_m^T R_m^{-1} r_m, \tag{14.134c}$$

where

$$a_m \triangleq [a_1^{(m)} \ a_2^{(m)} \ \ldots \ a_m^{(m)}]^T, \tag{14.135a}$$

$$x_m[n] \triangleq [x[n] \ x[n-1] \ \ldots \ x[n-p+1]]^T, \tag{14.135b}$$

$$r_m \triangleq [r[1] \ r[2] \ \ldots \ r[m]]^T, \tag{14.135c}$$

and R_m is an $m \times m$ symmetric Toeplitz matrix. The solution of (14.134b) for $m = 1$ is easily obtained by $a_1^{(1)} = -r[1]/r[0]$. Thus, if we can determine a_{m+1} from a_m we can develop an order-recursive algorithm for the solution of (14.134b).

To simplify the derivation, we use a matrix J_m (exchange matrix) defined by

$$J_m \triangleq \begin{bmatrix} 0 & 0 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & \ldots & 0 \\ 1 & 0 & \ldots & 0 \end{bmatrix}. \tag{14.136}$$

Note carefully that the bold-faced J is used for the exchange matrix while normal-faced J is used for the mse. The exchange matrix is symmetric $(J_m^T = J_m)$ and has the following properties:

$$J_m^T J_m = J_m J_m^T = I \quad \text{or} \quad J_m^{-1} = J_m^T. \tag{14.137}$$

The operation AJ flips A horizontally, whereas JA flips A vertically. It is easy to show that symmetric Toeplitz matrices have the following remarkable property:

$$R_m J_m = J_m R_m. \tag{14.138}$$

Suppose that we know the mth-order optimum predictor a_m specified by

$$R_m a_m = -r_m, \tag{14.139}$$

and we wish to use it to compute the $(m + 1)$th-order optimum predictor

$$R_{m+1} a_{m+1} = -r_{m+1}. \tag{14.140}$$

With the help of a simple example we can easily see that (14.140) can be partitioned as

$$\begin{bmatrix} R_m & J_m r_m \\ r_m^T J_m & r[0] \end{bmatrix} \begin{bmatrix} c_m \\ k_{m+1} \end{bmatrix} = - \begin{bmatrix} r_m \\ r[m+1] \end{bmatrix}. \tag{14.141}$$

Carrying out the matrix multiplication leads to the following set of equations:

$$R_m c_m + J_m r_m k_{m+1} = -r_m, \tag{14.142a}$$

$$r_m^T J_m c_m + r[0] k_{m+1} = -r[m+1]. \tag{14.142b}$$

Multiplying both sides of (14.139) from the left by J_m and using (14.138) we obtain

$$R_m J_m a_m = -J_m r_m. \tag{14.143}$$

Substitution of (14.143) and (14.139) into (14.142a) yields

$$R_m c_m - R_m J_m a_m k_{m+1} = R_m a_m. \tag{14.144}$$

Multiplying both sides of (14.144) by R_m^{-1} and rearranging its terms, we obtain

$$c_m = a_m + J_m a_m k_{m+1}. \tag{14.145}$$

Substituting (14.145) into (14.142b) and solving for k_{m+1} yields

$$k_{m+1} = -\frac{\beta_{m+1}}{J_m}, \tag{14.146}$$

where J_m is the minimum mse (14.134c) and the quantity β_{m+1} is defined by

$$\beta_{m+1} \triangleq r_m^T J_m a_m + r[m+1]. \tag{14.147}$$

We note that k_{m+1}, the last coefficient of a_{m+1}, is completely determined by a_m. Having determined k_{m+1}, we can compute the first m coefficients of a_{m+1} from (14.144). This leads to the well-known Levinson–Durbin recursion

$$a_{m+1} = \begin{bmatrix} a_m \\ 0 \end{bmatrix} + \begin{bmatrix} J_m a_m \\ 1 \end{bmatrix} k_{m+1}. \tag{14.148}$$

```
function [a,k,Jo]=levdur(r,p)
% Input: r(m), 0<=m<=p
% Output:
a=zeros(p+1,1);  % Prediction error filter
k=zeros(p,1);    % Lattice parameters
Jo=zeros(p+1,1); % LP MMSE for 0<=m<=M
% Initialization
J=r(1); Jo(1)=J;
beta=r(2); k(1)=-beta/J; a(1)=k(1);
J=J+beta*k(1); Jo(2)=J;
% Recursion
for m=2:p
  beta=(r(2:m))'*flipud(a(1:m-1))+r(m+1);
  k(m)=-beta/J;
  a(1:m)=[(a(1:m-1))' 0]'+[(flipud(a(1:m-1)))' 1]'*k(m);
  J=J+beta*k(m); Jo(m+1)=J;
end
a(2:p+1)=a(1:p); a(1)=1;
```

Figure 14.20 MATLAB function for the Levinson–Durbin algorithm.

Using (14.134c) and the Levinson–Durbin recursion we can show that (see Problem 39)

$$J_{m+1} = J_m + \beta_{m+1}k_{m+1} = (1 - k_{m+1}^2)J_m. \qquad (14.149)$$

Thus, the minimum mse for the pth-order linear predictor can be written as

$$J_p = J_{p-1} + \beta_p k_p = (1 - k_p^2)J_{p-1}. \qquad (14.150)$$

The Levinson–Durbin algorithm is initialized by noting that

$$k_1 = a_1^{(1)} = -\frac{r[1]}{r[0]} = \frac{\beta_1}{J_0}, \qquad (14.151)$$

which implies that $\beta_1 = r[1]$ and $J_0 = r[0]$. Then, for $m = 1, 2, \ldots, p - 1$ we compute J_m, β_{m+1}, k_{m+1}, and a_{m+1} using (14.149), (14.147), (14.146), and (14.148). Finally, we compute J_p using (14.149). A MATLAB function for the Levinson–Durbin algorithm is given in Figure 14.20. The Levinson–Durbin algorithm, which requires $p^2 + O(p)$ operations, provides *all* optimum predictors for order $m = 1, 2, \ldots, p$.

14.4.3   Lattice structures for linear prediction

The structure of Levinson–Durbin recursion (14.148) suggests the possibility of developing an order-recursive relation for the forward prediction error

$$e_{m+1}^f[n] \triangleq e_{m+1}[n] = x[n] + a_{m+1}^T x_{m+1}[n-1]. \qquad (14.152)$$

We first note that the input vector $x_{m+1}[n]$ can be partitioned as follows

$$x_{m+1}[n] = \begin{bmatrix} x_m[n] \\ x[n-m] \end{bmatrix} = \begin{bmatrix} x[n] \\ x_m[n-1] \end{bmatrix}. \tag{14.153}$$

For reasons to be seen below, we define the reverse coefficient vector

$$b_m \triangleq J_m a_m, \tag{14.154}$$

and rewrite the Levinson–Durbin recursion as

$$a_{m+1} = \begin{bmatrix} a_m \\ 0 \end{bmatrix} + \begin{bmatrix} b_m \\ 1 \end{bmatrix} k_{m+1}. \tag{14.155}$$

Using (14.155) and the first partitioning in (14.153), after some simple matrix manipulations, the forward prediction error can be expressed as follows:

$$e^f_{m+1}[n] = \left\{ x[n] + a_m^T x_m[n-1] \right\} + \left\{ b_m^T x_m[n-1] + x[n-1-m] \right\} k_{m+1}.$$

Since $e^f_m[n] = x[n] - a_m^T(-x_m[n-1])$, the forward predictor a_m uses the samples $x_k = -x[n-k]$, $k = 1, 2, \ldots m$ to estimate the sample $x[n]$. In a similar fashion, the linear estimator b_m uses the same set of samples to estimate the already known past sample $y = x[n-1-m]$. Since the present sample is $x[n-1]$, we use the same time index to denote the estimation error

$$e^b_m[n-1] \triangleq b_m^T x_m[n-1] + x[n-1-m], \tag{14.156}$$

which is known as *backward prediction error*. In both cases, the term prediction is used with estimation rather than *forecasting the future* or *guessing the past* in mind.

Using (14.152), (14.155), and (14.156) we obtain the following order-recursion:

$$e^f_{m+1}[n] = e^f_m[n] + k_{m+1} e^b_m[n-1]. \tag{14.157}$$

The optimum backward linear predictor $b_m = [b_1^{(m)} \quad b_2^{(m)} \quad \ldots \quad b_m^{(m)}]^T$ and the corresponding minimum mse J^b_m are given by (see Problem 38)

$$R_m b_m = -J_m r_m, \tag{14.158a}$$

$$J^b_m = r[0] + b_m^T J_m r_m, \tag{14.158b}$$

$$= r[0] - r_m^T R_m^{-1} r_m = J_m. \tag{14.158c}$$

Comparing (14.139) to (14.158a) we obtain $b_m = J_m a_m$, which explains the definition (14.154). Thus, the forward and backward optimum linear predictors have time-reversed impulse responses and the same minimum mse. This remarkable property follows from the stationarity property and the choice of mse as a criterion of performance.

If we flip (14.148) and use (14.154), we obtain a Levinson–Durbin recursion for b_m

$$b_{m+1} = \begin{bmatrix} 0 \\ b_m \end{bmatrix} + \begin{bmatrix} 1 \\ a_m \end{bmatrix} k_{m+1}. \tag{14.159}$$
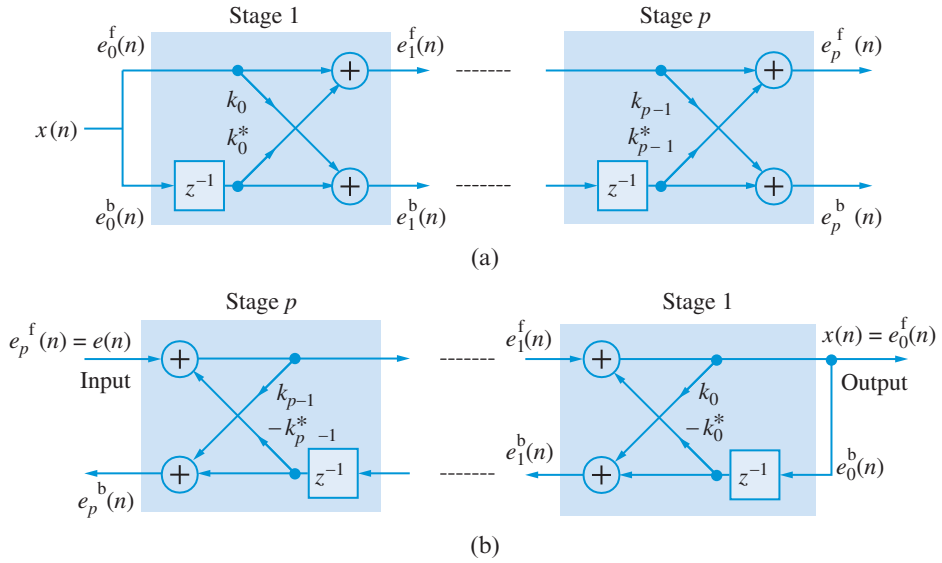
Figure 14.21 Lattice structures for (a) the all-zero prediction error filter, and (b) the all-pole synthesis filter.

We can now use (14.159) and the second partitioning in (14.153) to obtain an order-recursion for the backward prediction error (see Problem 53)

$$e^{b}_{m+1}[n] = e^{b}_{m}[n-1] + k_{m+1}e^{f}_{m}[n]. \tag{14.160}$$

Careful inspection of (14.157) and (14.160) shows that they describe the mth-stage of an all-zero lattice filter (see Section 9.4). Therefore, we can use an all-zero lattice filter to implement the pth-order prediction error filter $A_p(z)$ and an all-pole lattice filter to implement its inverse filter $H(z) = 1/A_p(z)$. These structures are illustrated in Figure 14.21. More implementation details can be found in Section 9.4 using the following correspondences: $f_m[n] \rightarrow e^{f}_{m}[n]$ and $g_m[n] \rightarrow e^{b}_{m}[n]$. The lattice coefficients $k_1, k_2, \ldots, k_p$, which are called *reflection coefficients* in the speech processing literature, are obtained by the Levinson–Durbin algorithm. Algorithms for the conversion between lattice and direct form coefficients are given in Section 9.4.

Statistical interpretation An interesting statistical interpretation of the lattice parameters, which provides additional insight into linear prediction theory, can be obtained as follows. According to the orthogonality principle (14.107), the optimum backward prediction error $e^{b}_{m}[n-1]$ is orthogonal to the vector $\boldsymbol{x}_m[n-1]$ used for the estimation, that is, $E(\boldsymbol{x}_m[n-1]e^{b}_{m}[n-1]) = \boldsymbol{0}$. Using this property and some obvious definitions, we have

$$
\begin{aligned}
E(e^{f}_{m}[n]e^{b}_{m}[n-1]) &= E\{(x[n] + \boldsymbol{a}^{T}_{m}\boldsymbol{x}_m[n-1])e^{b}_{m}[n-1]\} \\
&= E(x[n]e^{b}_{m}[n-1]) \\
&= E\{x[n](x[n-m-1] + \boldsymbol{b}^{T}_{m}\boldsymbol{x}_m[n-1])\} \\
&= r[m+1] + \boldsymbol{b}^{T}_{m}\boldsymbol{r}_m = \beta_{m+1}. \tag{14.161}
\end{aligned}
$$

The quantity β_{m+1} provides the *partial correlation (PARCOR)* between $x[n]$ and $x[n-m-1]$ after the influence of intermediate samples $x[n-1], \ldots, x[n-m]$ has been removed. Since the forward and backward mses are equal for stationary processes, that is,

$$J_m = \mathrm{E}\{(e_m^{\mathrm{f}}[n])^2\} = \mathrm{E}\{(e_m^{\mathrm{b}}[n-1])^2\}, \tag{14.162}$$

using (14.161) and (14.162) we obtain the *normalized PARCOR coefficient*

$$k_{m+1} = -\frac{\beta_{m+1}}{J_m} = -\frac{E(e_m^{\mathrm{f}}[n]e_m^{\mathrm{b}}[n-1])}{\sqrt{E\{(e_m^{\mathrm{f}}[n])^2\}}\sqrt{E\{(e_m^{\mathrm{b}}[n-1])^2\}}}. \tag{14.163}$$

Since k_{m+1} is the correlation coefficient between $e_m^{\mathrm{f}}[n]$ and $e_m^{\mathrm{b}}[n-1]$, we have

$$-1 \le k_{m+1} \le 1. \tag{14.164}$$

Since $J_m = (1 - k_m^2)J_{m-1}$ is strictly greater than zero for predictors of all orders, it follows that $-1 < k_m < 1$ for all m; this stricter inequality requires that $\boldsymbol{R}_p$ is positive definite. As we have already mentioned in Section 9.4, this condition guarantees that the prediction error filter is minimum phase, that is it has its zeros inside the unit circle.

14.4.4   Linear prediction in practice

So far we have assumed that we know the values $r[0], r[1], \ldots, r[p]$ of the ACRS; however, in practice, only a set of signal samples, say, $x[0], x[1], \ldots, x[N-1]$, is available. In this case, we determine the coefficients of the linear predictor

$$e[n] = x[n] + \sum_{k=1}^{p} a_k x[n-k] \tag{14.165}$$

by minimizing the sum of squared errors $e^2[n]$ over a finite range $N_1 \le n \le N_2$. We first note that the sum of squared errors can be expressed as follows:

$$S \triangleq \sum_{n=N_1}^{N_2} e^2[n] = \gamma_{00} + 2\sum_{k=1}^{p} a_k \gamma_{k0} + \sum_{k=1}^{p}\sum_{m=1}^{p} a_k a_m \gamma_{km}, \tag{14.166}$$

where

$$\gamma_{km} \triangleq \sum_{n=N_1}^{N_2} x[n-k]x[n-m]. \tag{14.167}$$

Minimization of (14.166) shows that the coefficients of the least squares linear predictor and the corresponding least squares error are given by (see Problem 54)

$$\Gamma \hat{a} = -\gamma, \tag{14.168}$$

and

$$S = \gamma_{00} + \boldsymbol{\gamma}^{\mathrm{T}}\hat{\boldsymbol{a}} = \gamma_{00} - \boldsymbol{\gamma}^{\mathrm{T}}\boldsymbol{\Gamma}^{-1}\boldsymbol{\gamma}, \tag{14.169}$$

where the symmetric matrix $\boldsymbol{\Gamma}$ and the vector $\boldsymbol{\gamma}$ are defined by

$$\boldsymbol{\Gamma} \triangleq \begin{bmatrix} \gamma_{11} & \gamma_{12} & \cdots & \gamma_{1p} \\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{p1} & \gamma_{p2} & \cdots & \gamma_{pp} \end{bmatrix}, \quad \boldsymbol{\gamma} = \begin{bmatrix} \gamma_{10} \\ \gamma_{20} \\ \vdots \\ \gamma_{p0} \end{bmatrix}. \tag{14.170}$$

The choice of the range of summation in (14.166) has crucial implications for the structure of matrix $\boldsymbol{\Gamma}$ and the properties of the resulting linear predictor. To best explain the two most important choices, we consider an example for $p = 3$ and $N = 8$. Writing (14.165) in matrix form for $n = 0, 1, \ldots, N + p - 1$, we have

$$\begin{bmatrix} e[0] \\ e[1] \\ e[2] \\ e[3] \\ e[4] \\ e[5] \\ e[6] \\ e[7] \\ e[8] \\ e[9] \\ e[10] \end{bmatrix} = \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ x[0] & 0 & 0 \\ x[1] & x[0] & 0 \\ x[2] & x[1] & x[0] \\ x[3] & x[2] & x[1] \\ x[4] & x[3] & x[2] \\ x[5] & x[4] & x[3] \\ x[6] & x[5] & x[4] \\ X[7] & x[6] & x[5] \\ 0 & x[7] & x[6] \\ 0 & 0 & x[7] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}. \tag{14.171}$$

Let us denote by $\tilde{\boldsymbol{x}}_0$ the column vector and by $\tilde{\boldsymbol{x}}_k$ the kth column of the matrix on the right hand side of (14.171). Careful inspection of (14.167) and (14.171) shows that

$$\gamma_{km} = \tilde{\boldsymbol{x}}_k^{\mathrm{T}}\tilde{\boldsymbol{x}}_m. \tag{14.172}$$

Windowing method If we choose $N_1 = 0$ and $N_2 = N + p - 1$, that is, if we use all rows in (14.171), we have to set to zero the unavailable signal samples. This process is equivalent to applying a rectangular window to the original signal; however, in most practical applications we use a Hamming or Hann window. In this case, the vector $\tilde{\boldsymbol{x}}_0$ contains the available N signal samples followed by p zeros; the vector $\tilde{\boldsymbol{x}}_k, k = 1, 2, \ldots, p$ is obtained by circularly shifting the elements of $\tilde{\boldsymbol{x}}_0$ down by k positions, and as a result the quantity γ_{km} depends on the difference $|k - m|$ only. From (14.171), (14.171), and (14.28) we conclude that

$$\gamma_{km} = N\hat{r}[\ell] = \sum_{n=0}^{N-\ell-1} x[n]x[n + \ell], \quad \ell = |m - k|. \tag{14.173}$$

Comparison of (14.170) and (14.173) shows that $(1/N)\boldsymbol{\Gamma}$ is an estimate of the Toeplitz correlation matrix $\boldsymbol{R}$ and $(1/N)\boldsymbol{\gamma}$ is an estimate of the correlation vector $\boldsymbol{r}$. Dividing both sides of (14.166) and (14.168) by N, we obtain

$$\hat{\boldsymbol{R}}\hat{\boldsymbol{a}} = -\hat{\boldsymbol{r}}, \tag{14.174}$$

and

$$\hat{J} = \frac{1}{N} \sum_{n=0}^{N+p-1} e^2[n] = r[\hat{0}] + \hat{\boldsymbol{r}}^T \hat{\boldsymbol{a}}. \tag{14.175}$$

Since $\hat{\boldsymbol{R}}$ is symmetric Toeplitz and positive definite, the prediction error filter is guaranteed to be minimum phase. Furthermore, we can solve (14.174) using the Levinson–Durbin algorithm and implement the direct and inverse filters using lattice structures. The lattice coefficients can also be computed directly from the data by replacing the expectations in (14.163) by time averages. The approach suggested by Itakura and Saito (1971) uses the formula

$$k_m^{(I)} = -\frac{\sum_{n=0}^{N+p-1} e_m^f[n]e_m^b[n-1]}{\sqrt{\sum_{n=0}^{N+p-1} (e_m^f[n])^2 \sum_{n=0}^{N+p-1} (e_m^b[n-1])^2}}, \tag{14.176}$$

after the $(m-1)$th-stage of the lattice; the coefficient k_1 is obtained by noting that $e_0^f[n] = e_0^b[n] = x[n]$. Several other estimates are discussed in Makhoul (1977).

Nonwindowing method If we set $N_1 = p$ and $N_2 = N - 1$, we only use the blue rows in (14.171); thus, we do *not* have to force the unavailable signal sample values to zero. The matrix $\boldsymbol{\Gamma}$ is symmetric but not Toeplitz; thus, we cannot use the Levinson–Durbin algorithm to solve the normal equations or to derive a lattice structure. Furthermore, the prediction error filter derived by the nonwindowing method is not guaranteed to have minimum phase.

The windowing and nonwindowing methods are known in signal processing literature as the *autocorrelation* and *covariance methods*, respectively [Makhoul (1975)]; we avoid these terms because they can lead to misleading statistical interpretations. We next provide an application of the windowing method, which is the most widely used method of linear prediction, to all-pole speech signal modeling.

All-pole speech signal modeling Speech production involves three processes: generation of the sound excitation, articulation by the vocal tract, and radiation from the lips and/or nostrils. If the excitation is a quasi-periodic train of air pressure pulses, produced by vibration of the vocal cords, the result is a voiced (quasi-periodic) sound, such as "e." Unvoiced or fricative (noise-like) sounds, such as "g," are produced by first creating a constriction in the vocal tract, usually toward the mouth end. Then we generate turbulence by forcing air through the constriction at a sufficiently high velocity. The resulting excitation is a broadband noiselike waveform.

The spectrum of the excitation is shaped by the vocal tract tube, which has a frequency response that resembles the resonances of organ pipes or wind instruments. The resonant frequencies of the vocal tract tube are known as *formant frequencies*, or simply
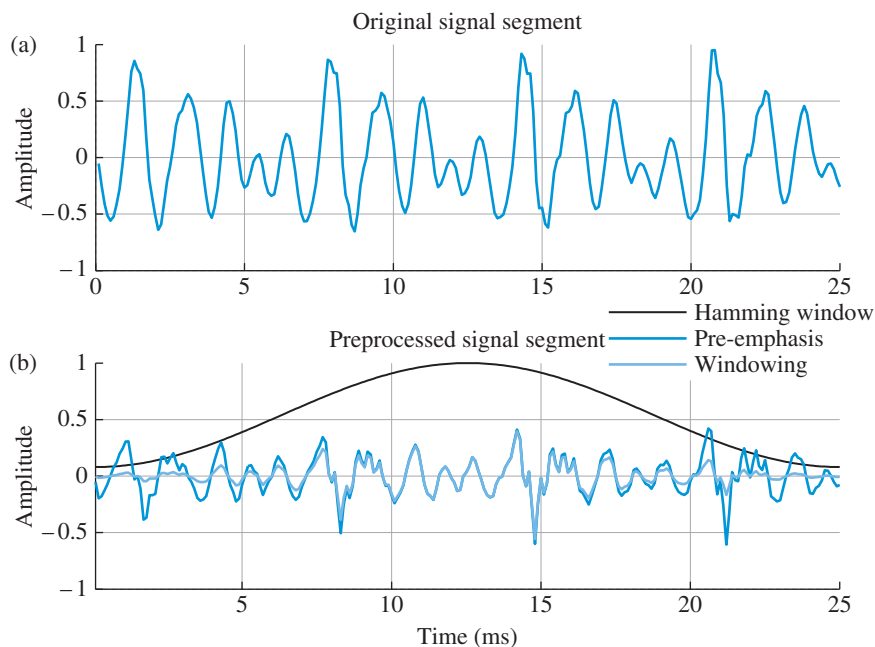
Figure 14.22 Effects of pre-emphasis and Hamming windowing on a segment of speech signal; these are typical preprocessing steps in linear prediction analysis.

formants. Changing the shape of the vocal tract changes its frequency response and results in generation of different sounds. Since the shape of the vocal tract changes slowly during continuous speech, we usually assume that it remains almost constant over intervals of the order of 10 ms. More details about speech signal generation and processing can be found in Rabiner and Schafer 2010, O'Shaughnessy 1987, and Rabiner and Juang 1993.

The top plot in Figure 14.22 shows a segment from a voiced speech signal sampled at $F_s = 10$ kHz. This signal is first filtered by the highpass filter $H(z) = 1 - 0.95z^{-1}$ to reduce the dynamic range of its spectrum (pre-emphasis) and then windowed using a Hamming window to reduce the discontinuities at the beginning and end of the segment; the resulting signals are shown in the bottom plot of Figure 14.22. The all-pole model estimated by the windowing method can be used to estimate the spectrum of the speech segment using the formula

$$\hat{S}_p(e^{j\omega}) = \frac{\hat{\sigma}_z^2}{|1 + \sum_{k=1}^{p} a_k e^{-j\omega k}|^2}, \tag{14.177}$$

where $\hat{\sigma}_z^2 = \sum_{n=0}^{N+p-1} e^2[n]/(N + p)$ is the estimated variance of the excitation white noise (see Section 13.5.3). Figure 14.23 shows the periodogram and all-pole model spectral densities for $p = 6$ and $p = 12$. We note that the all-pole model with $p = 12$ provides a good estimate of the envelop of the periodogram; this property, which is known as spectral matching, is discussed in Makhoul (1975). More details about this example are given in Tutorial Problem 21.
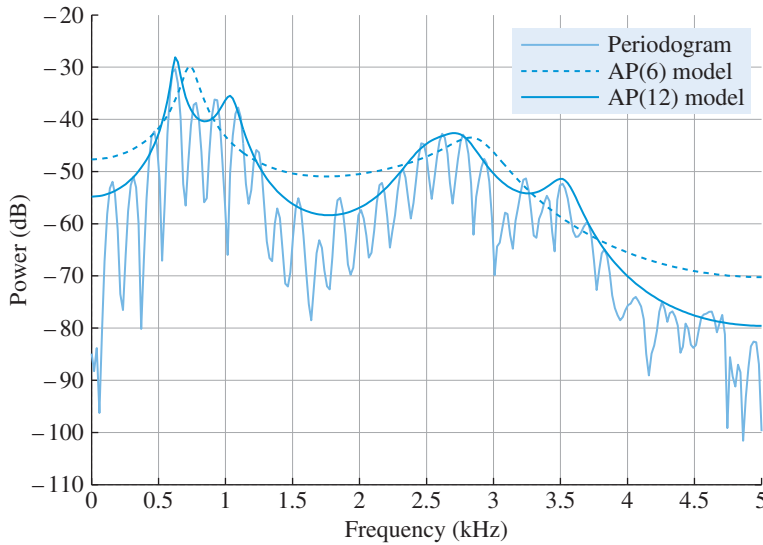
Figure 14.23 Comparison of periodogram and all-pole spectra for a segment of voiced speech signal; all-pole spectra "try" to match the envelop- of the periodogram.

14.5   Optimum orthogonal transforms

As we discussed in Section 14.3 there are some applications, involving random signals, which require the design of FIR filters using statistical criteria of performance. In this section we introduce a $p \times p$ orthogonal transform for random signals, known as the *Karhunen–Loève transform* (KLT) or *principal component transform* which can be thought of as the statistical counterpart of the DFT.

Orthogonal transforms Consider a linear transform that maps a set of random variables $x_1, x_2, \ldots, x_p$ to a new set of random variables $y_1, y_2, \ldots, y_p$ (transform coefficients) using the following set of linear equations

$$y_k = \sum_{i=1}^{p} a_{ik} x_i = a_{1k} x_1 + \cdots + a_{pk} x_p = a_k^{\mathrm{T}} x, \quad k = 1, \ldots, p. \tag{14.178}$$

The forward transform (14.178) can be expressed in compact matrix form as

$$
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_p \end{bmatrix} =
\begin{bmatrix} \longleftarrow & a_1^{\mathrm{T}} & \longrightarrow \\ \longleftarrow & a_2^{\mathrm{T}} & \longrightarrow \\ & \vdots & \\ \longleftarrow & a_p^{\mathrm{T}} & \longrightarrow \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \quad \text{or} \quad y = A^{\mathrm{T}} x, \tag{14.179}
$$

where A is a $p \times p$ transform matrix defined by

$$A \triangleq \begin{bmatrix} a_1 & a_2 & \ldots & a_p \end{bmatrix}. \tag{14.180}$$

To simplify the computation of the inverse transform, we require that the matrix A is orthogonal, that is,

$$A^T = A^{-1}. \tag{14.181}$$

This implies that the columns of A are orthonormal vectors, that is,

$$A^T A = I \quad \text{or} \quad a_i^T a_j = \begin{cases} 1, & i = j \\ 0. & i \neq j \end{cases} \tag{14.182}$$

Therefore, $(A^T)^{-1} = A$ and the inverse transform is easily evaluated by

$$x = Ay = \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ a_1 & a_2 & \cdots & a_p \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_p \end{bmatrix} = \sum_{k=1}^{p} c_k a_k. \tag{14.183}$$

The inverse transform provides an expansion of the input signal vector x into orthogonal components using as a basis the columns of transform matrix A. The coefficient y_k measures the similarity (correlation) of x with the basis vector a_k.

All orthogonal transforms preserve the energy of the input signal vector. Indeed, using (14.183) and (14.182), we obtain

$$\sum_{k=1}^{p} x_k^2 = x^T x = y^T A^T A y = y^T y = \sum_{k=1}^{p} y_k^2, \tag{14.184}$$

which can be considered as a generalization of Parseval's theorem. The result of an orthogonal transformation is a solid rotation of the coordinate system (see Tutorial Problem 17).

The Karhunen–Loève transform We wish to find the orthogonal transform that provides a representation of the input signal vector which is optimum with respect to the mse criterion. To this end, suppose that we retain the first m coefficients and we replace the $(p - m)$ remaining ones with preselected constants b_k. This yields the approximation

$$\hat{x}_m = \sum_{k=1}^{m} y_k a_k + \sum_{k=m+1}^{p} b_k a_k. \tag{14.185}$$

The error introduced by this approximation is given by

$$e_m = x - \hat{x}_m = \sum_{k=m+1}^{p} (y_k - b_k) a_k. \tag{14.186}$$

To find the average error energy, we first note that

$$e_m^T e_m = \sum_{i=m+1}^{p} \sum_{j=m+1}^{p} (y_i - b_i) a_i^T a_j (y_j - b_j) = \sum_{i=m+1}^{p} (y_i - b_i)^2, \tag{14.187}$$

because $a_i^T a_j = 0$ for $i \neq j$ (orthogonality property). Therefore the mse is

$$J_m \triangleq \mathrm{E}(e_m^T e_m) = \sum_{i=m+1}^{p} \mathrm{E}[(c_i - b_i)^2]. \tag{14.188}$$

The coefficients b_i that minimize J_m are obtained from the equations

$$\frac{\partial J_m}{\partial b_i} = -2[\mathrm{E}(c_i) - b_i] = 0, \quad i = m+1, \ldots, p, \tag{14.189}$$

which yields the optimum values

$$b_i = \mathrm{E}(c_i) = \mathrm{E}(a_i^T x) = a_i^T \mathrm{E}(x) = a_i^T m. \tag{14.190}$$

Thus setting $m = 0$ yields $b_i = 0$, and the corresponding terms in (14.185) may be simply omitted. For this reason we usually remove the mean from the data, compute the transform of the zero mean data, and then add the mean to the reconstructed data.

Since $b_i = \mathrm{E}(y_i)$, we have $\mathrm{E}[(y_i - b_i)^2] = \mathrm{var}(y_i)$. Thus, using (14.178) and (13.45), we obtain

$$\mathrm{var}(c_i) = a_i^T C a_i, \tag{14.191}$$

where C is the covariance matrix of x. Hence, the mse (14.188) can be written as

$$J_m = \sum_{i=m+1}^{p} \mathrm{var}(y_i) = \sum_{i=m+1}^{p} a_i^T C a_i. \tag{14.192}$$

Because $a_i^T C a_i \geq 0$ we can minimize equation (14.192) by minimizing each term of the summation. To minimize (14.192) under the orthogonality constraints, (14.182), we use the method of Lagrange multipliers, see for example Strang (1986). The constrained minimization of (14.192) is equivalent to the unconstrained minimization of the Lagrangian function

$$V = \sum_{i=m+1}^{p} \left[a_i^T C a_i + \lambda_i (1 - a_i^T a_i) \right] \tag{14.193}$$

with respect to a_i, where λ_i is the Lagrange multiplier. The partial derivative with respect to a_i is a $p \times 1$ vector defined by

$$\frac{\partial V}{\partial a_i} \triangleq \left[\frac{\partial V}{\partial a_{1i}} \quad \frac{\partial V}{\partial a_{2i}} \quad \cdots \quad \frac{\partial V}{\partial a_{pi}} \right]^T. \tag{14.194}$$

Now, it can be shown that (see Tutorial Problem 18)

$$\frac{\partial a_i^T C a_i}{\partial a_i} = 2C a_i \quad \text{and} \quad \frac{\partial a_i^T a_i}{\partial a_i} = 2 a_i. \tag{14.195}$$

Differentiating (14.193) with respect to $\boldsymbol{a}_i$, setting the result equal to zero, and solving, gives

$$\frac{\partial V}{\partial \boldsymbol{a}_i} = 2\mathbf{C}\boldsymbol{a}_i - 2\lambda_i \boldsymbol{a}_i = \mathbf{0}, \tag{14.196}$$

and finally

$$\mathbf{C}\boldsymbol{a}_i = \lambda_i \boldsymbol{a}_i. \tag{14.197}$$

Equation (14.197), by definition, implies that $\boldsymbol{a}_i$ is an eigenvector of the covariance matrix $\mathbf{C}$ and λ_i is the corresponding eigenvalue. Since the covariance matrix is real and symmetric it will always have real eigenvalues. Substitution of (14.197) into (14.192) shows that the resulting mse is given by

$$J_m = \sum_{i=m+1}^{p} \boldsymbol{a}_i^{\mathrm{T}} \lambda_i \boldsymbol{a}_i = \sum_{i=m+1}^{p} \lambda_i. \quad 1 \leq m \leq p \tag{14.198}$$

Therefore, the minimum mse is obtained by using the coefficients $y_1, y_2, \ldots, y_m$ corresponding to the m larger eigenvalues.

The covariance of the optimum coefficients y_i and y_j is given by

$$\mathrm{cov}(y_i, y_j) = \boldsymbol{a}_i^{\mathrm{T}} \mathbf{C} \boldsymbol{a}_j = \boldsymbol{a}_i^{\mathrm{T}} \lambda_j \boldsymbol{a}_j = \begin{cases} \lambda_i, & i = j \\ 0, & i \neq j \end{cases} \tag{14.199}$$

which implies that the optimum transform coefficients are uncorrelated.

The total variance of the random variables $x_1, \ldots, x_p$, which is equal to the sum of the diagonal elements (known as trace) of $\mathbf{C}$ is (see Tutorial Problem 19)

$$\sum_{k=1}^{p} \mathrm{var}(x_k) = \sum_{k=1}^{p} \mathrm{var}(y_k), \tag{14.200}$$

which states that the optimum transform redistributes the total variance of the input random variables by packing the maximum variance in $m \leq p$ components of the input vector for every value of m (optimum packing property).

In conclusion, the orthogonal transform (14.179) that minimizes the approximation mse (14.188) is determined by the eigenvectors of the covariance matrix of the input random vector, see (14.197). This transform was first introduced by Hotelling, who used the name *method of principal components*. The analogous transform for continuous time signals was obtained by Karhunen and Loève. In signal processing we use the term *Karhunen–Loève transform* or *KLT* in abbreviated form.

Computation of A Since the KLT coefficients are uncorrelated their covariance matrix is diagonal. Therefore,

$$\mathrm{cov}(\boldsymbol{y}) \triangleq \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p). \tag{14.201}$$
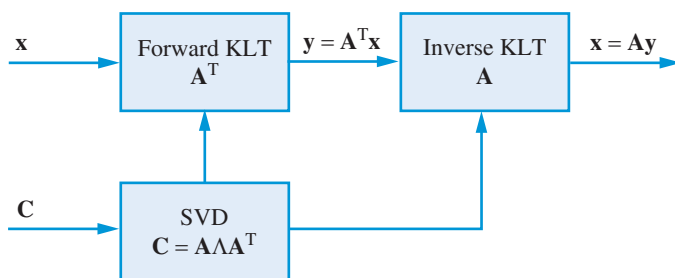
Figure 14.24 Computation of the forward and inverse Karhunen–Loève transform using the SVD.

The equations $\mathbf{C}\boldsymbol{a}_k = \lambda_k \boldsymbol{a}_k$, $k = 1, 2, \ldots, p$, can be grouped together into a single equation as

$$\mathbf{CA} = \boldsymbol{A}\boldsymbol{\Lambda}. \tag{14.202}$$

Multiplying from the right by $\boldsymbol{A}^{-1}$ and using the orthogonality property yields

$$\mathbf{C} = \boldsymbol{A}\boldsymbol{\Lambda}\boldsymbol{A}^{\mathrm{T}}, \tag{14.203}$$

which is known as the *spectral decomposition* of **G**. This eigenvalue–eigenvector decomposition can be obtained using the MATLAB functions `eig` or `svd` as follows:

$$\texttt{[A,Lambda]=eig(c),} \tag{14.204}$$

$$\texttt{[U,Lambda,A]=svd(c).} \tag{14.205}$$

We suggest using function `svd` because it sorts the eigenvalues in decreasing order of magnitude. Given A, the forward KLT (14.179) and inverse KLT (14.183) can be easily computed using matrix-by-vector multiplications. This procedure is summarized in Figure 14.24.

KLT in practice So far, we have assumed that **m** and **G** are known. However, in practice both quantities have to be estimated by the sample mean and covariance

$$\hat{\mathbf{m}} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}[n], \quad \hat{\mathbf{C}} = \frac{1}{N} \sum_{n=1}^{N} (\boldsymbol{x}[n] - \hat{\mathbf{m}})(\boldsymbol{x}[n] - \hat{\mathbf{m}})^{\mathrm{T}}, \tag{14.206}$$

where $\boldsymbol{x}[1], \ldots, \boldsymbol{x}[N]$ are observations from the random vector $\boldsymbol{x}$. Although this gives rise to an "estimated" KLT, we will simply refer to it as KLT; the difference will be obvious from the context. Each observation is transformed using the formula $\boldsymbol{y}[n] = \boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}[n]$, where A is obtained from (14.205).

If we create a matrix X, whose rows are the observation vectors $\boldsymbol{x}[1], \ldots, \boldsymbol{x}[N]$, we can compute the KLT using the MATLAB script shown in Figure 14.25. The error is computed directly, and by using the eigenvalues, to demonstrate the validity of (14.198). This code can be easily modified to fit the needs of different applications.

```
% Script klt.m
[N,p]=size(X);
C=cov(X,1);
[U,L,A]=svd(C);
lambda=diag(L);
Mx=repmat(mean(X),N,1);
Y=(X-Mx)*A;
Xhat=Y(:,1:m)*A(:,1:m)'+Mx;
E=X(:)-Xhat(:);
MSE1=sum(E.^2)/N
MSE2=sum(lambda(m+1:p))
```

Figure 14.25 A MATLAB script for computation of the KLT, which makes efficient use of matrix operations.

Example 14.6   Geometric interpretation

The observations $x[n]$, $1 \leq n \leq N$ of a $p \times 1$ random vector x form a swarm of points in a p-dimensional space. Although the KLT can be applied to observations from any distribution, it is easier to explain and visualize its operation if the swarm of points is ellipsoidal. If the components $x_1, \ldots, x_p$ of x are correlated, the ellipsoidal swarm of points is *not* oriented parallel to any of the axes represented by $x_1, \ldots, x_p$. This is illustrated in Figure 14.26 for $p = 2$; without loss of generality, we assume zero-centered random variables.

The variables x_1 and x_2 in Figure 14.26 exhibit significant positive correlation because they are clustered around the line $x_1 = x_2$. We next rotate the coordinate system with axes x_1 and x_2 to obtain a coordinate system with axes y_1 and y_2. This is an invertible transformation because given y_1 and y_2 we can perform the inverse rotation to obtain the original variables x_1 and x_2.
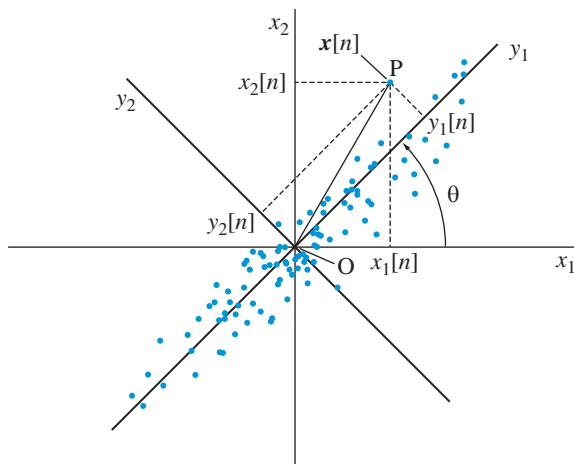
Figure 14.26 Geometric interpretation of the Karhunen–Loève transform for $p = 2$.

The rotation of the coordinate system has two effects upon the new coordinates y_1 and y_2. First, we note that the swarm of points is lined up not with the line $y_1 = y_2$ but with the y_1 axis. Since y_2 is likely to be small independent of the value of y_1, the coefficients y_1 and y_2 are "less correlated" than the variables x_1 and x_2. Second, rotating the coordinate system rearranged the variances. Although the original random variables had approximately the same variance, that is, $\text{var}(x_1) \simeq \text{var}(x_2)$, more of the variance is now in the first coefficient, that is, $\text{var}(y_1) > \text{var}(y_2)$. From Figure 14.26 we can easily see that $OP_n^2 = x_1^2[n] + x_2^2[n] = y_1^2[n] + y_2^2[n]$ (Pythagorean theorem). If we recall that $\text{var}(x_1) \simeq (1/N) \sum_{k=1}^{N} x_1^2[n]$, etc. we conclude that $\text{var}(x_1) + \text{var}(x_2) = \text{var}(y_1) + \text{var}(y_2)$, which is exactly equation (14.200) for $p = 2$. Therefore, the KLT preserves the total variance of the input vector; however, it packs the maximum variance (energy) in the first KLT coefficient.

Each data point in the original (natural) coordinates is represented by two numbers $x_i[n]$ and $x_2[n]$. Next, suppose that we wish to represent each data point with a single coefficient $y_1[n]$, which is the first coordinate in the new (rotated) system. Replacing the two variables $x_1[n]$ and $x_2[n]$ by a single coefficient $y_1[n]$ leads to a reduction of dimensionality from two to one. Equivalently the amount of storage required to store the data is reduced by a factor of two. The quality of the approximation is measured by the distance $y_2[n]$ of the original points from their orthogonal projections $y_1[n]$ on the y_1 axis. Since $\sum_n OP_n^2$ is fixed for a given data set, irrespective of the coordinate system, finding the angle θ that minimizes the sum of squared perpendicular distances of the points from the y_1 axis, $\sum_n y_2^2[n]$, is equivalent to specifying the axis y_1 such that the projections of the points on it have maximum variance. The quantity $(1/N)\sum_{n=1}^{N} y_1^2[n]$ is the sample variance of the projections because O is the centroid of the data points. Further discussion, with MATLAB illustrations, is provided in Tutorial Problem 22. ∎

To summarize the essential definitions and properties of KLT, the kth coefficient is the linear combination $y_k = a_k^T x$, which has the greatest variance for all a_k satisfying $a_k^T a_k = 1$ and $a_k^T a_i = 0$ $(k < j)$. The vector a_k is given by the eigenvector corresponding to the kth largest eigenvalue of $\mathbf{C} = \text{cov}(x)$, and $\text{var}(y_k) = \lambda_k$. The coefficients $y_1, \ldots, y_p$ represent x in a new coordinate system spanned by the eigenvectors $a_1, \ldots, a_p$. The coordinate transformation $y = A^T x$ is a rotation of the original axes to align with the eigenvectors of $\mathbf{C}$. Finally, the KLT is optimum in the sense of completely decorrelating the input signal vector (mutually uncorrelated coefficients) and maximizing the amount of variance (energy) "packed" into the lowest-order coefficients.

Learning summary

- An estimator is a formula used to "guess" the value of some unknown parameter of a probability distribution; the number obtained for specific data is called an estimate. Since an estimate changes from data to data, *our objective should be to find estimators that will give good estimates "on the average."*

- An estimator is unbiased if its expected value equals the true value of the parameter of interest. The variance of an unbiased estimator is a measure of its accuracy. An estimator is consistent if both its variance and bias converge to zero as the number of observations becomes very large.

- The basic tool in estimating the PSD is the periodogram:

$$I(\omega) = \frac{1}{N} \left| \sum_{n=0}^{N-1} x[n] e^{-j\omega n} \right|^2 = \sum_{\ell=-(N-1)}^{N-1} \hat{r}[\ell] e^{-j\omega \ell}.$$

Since the periodogram and the sample ACRS form a Fourier transform pair they convey the same information. However, each form makes some of that information easier to interpret.

- The periodogram is not a good estimator of PSD because its values are asymptotically (that is, for large N) uncorrelated random variables with means and standard deviations equal to the corresponding values of the PSD.

- Although the periodogram itself does *not* provide a consistent estimator of the PSD, if we assume that $S(\omega)$ is a smooth function of ω, we can average over adjacent values (Blackman–Tukey PSD estimator) or average values of multiple periodograms from the same process (Bartlett–Welch PSD estimator) to obtain a much less variable estimate of $S(\omega)$.

- To determine the presence or absence of a signal of known form (shape), we should use a filter that maximizes the output signal-to-noise ratio (matched filter). The impulse response of an FIR matched filter is determined by the correlation matrix of the noise and the waveform of the signal of interest.

- To determine the form (shape) of a signal that is known to be present, we should use a filter that minimizes the mean square error between the actual output and the desired output (Wiener filter). The impulse response of an FIR Wiener filter is specified by the correlation matrix of the input signal and the cross-correlation vector between input signal and desired response.

- The Karhunen–Loève transform is a $p \times p$ orthogonal transform determined by the eigenvectors of the input signal covariance matrix. The transform coefficients are uncorrelated random variables sorted in order of decreasing variance. The KLT maximizes the amount of variance packed into the lowest-order coefficients.

TERMS AND CONCEPTS

Affine estimator A linear estimator which includes an extra constant term, that is, $\hat{y} = a_1 x_1 + \cdots + a_p x_p + b$.

ACRS The autocorrelation sequence of a stationary stochastic process; it measures the correlation between two samples $r_x[\ell] = E(x[n + \ell]x[n])$.

ACVS The autocovariance sequence of a stationary stochastic process; it measures the covariance between two samples $c_x[\ell] = E\{(x[n + \ell] - \mu)(x[n] - \mu)\}$.

Bias of estimator The average deviation of an estimator from the true value.

Blackman–Tukey PSD estimator A PSD estimator obtained by smoothing a single periodogram by windowing the sample ACRS.

Data A collection of observations from a random variable.

Ergodicity The condition that allows us to equate ensemble averages with infinite length time averages obtained from a single "representative" realization of a stochastic process.

Estimate The value produced by an estimator for a given set of data.

Estimator The formula used to determine an estimate from the data.

Karhunen–Loève transform A finite orthogonal transform that represents a random vector by a set of uncorrelated coefficients arranged in terms of decreasing variance.

Linear estimator The estimate is a linear combination of the observations, that is, $\hat{y} = a_1 x_1 + \cdots + a_p x_p$.

Linear predictor The linear minimum mse estimator of the present sample $x[n]$ of a random process given its p past values $x[n - 1], \ldots, x[n - p]$.

Matched filter A filter designed to maximize the output SNR when its input is the sum of a finite length signal with known shape and a stationary random process.

Normal equations The set of linear equations that specifies the coefficients of an FIR Wiener filter or linear predictor.

Orthogonality principle A linear estimator minimizes the mse if the estimation error is uncorrelated to all variables used to form the estimate.

Periodogram A PSD estimator obtained by evaluating the squared DTFT of an N-point data set at N equispaced frequencies; it is equal to the DTFT of the sample ACRS of the data.

PSD The power spectral density of a stationary random process is the Fourier transform of the ACRS; it shows the distribution of power as a function of frequency.

Sampling distribution The probability distribution of an estimator.

Variance of estimator The average squared deviation of an estimator from its mean value.

Welch PSD estimator A PSD estimator obtained by averaging multiple periodograms; each periodogram corresponds to a possibly overlapping segment of the original data set.

Wiener filter A filter designed to minimize the mse between a desired output stochastic process and an estimate formed by a linear combination of samples from another related process.

Matlab functions and scripts

Name	Description	Page
acrs*	Computes the sample ACRS	838
acrsfft*	Computes the sample ACRS using the FFT	841
corrcoef*	Computes the sample correlation coefficient	834
cov	Computes the sample covariance	834
eig	Computes the eigenvalues and eigenvectors	881
klt*	Computes the KLT	881
mean	Computes the sample mean	832
psdbt*	Computes the BT PSD estimate	853
psdmodper*	Computes the modified periodogram	847
psdper*	Computes the periodogram	840
psdwelch*	Computes Welch's PSD estimate	857
pwelch	Computes Welch's PSD estimate	857
svd	Computes the singular value decomposition	881
var	Computes the sample variance	833

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A simple introduction to the theory and practice of statistics, at the same level as in this book, is given in Ross (2004) and Hogg and Tanis (2005).
2. The standard reference for a theoretical treatment of PSD estimation is Priestley (1981). However, an excellent introduction to the concepts and practical application of spectral analysis is provided by Jenkins and Watts (1968) and Percival and Walden (1993). Both books provide practical advice and show how to work with practical data sets. A concise theoretical introduction to spectral analysis, including parametric techniques (based on both ARMA and harmonic models), is given by Stoica and Moses (2005).
3. A lucid theoretical introduction to the basic principles of linear estimation and optimum filtering is provided by Gray and Davisson (2004). The topics of optimum filtering and adaptive filtering are thoroughly discussed in Haykin (2002) and Manolakis et al. (2005). The classical paper by Bode and Shannon (1950), although it is formulated in continuous-time, provides an excellent introduction to the fundamental concepts and ideas of optimum filtering.
4. A theoretical derivation and performance analysis of FIR matched filters, in the context of detection theory, are given in Kay (1998). An insightful introduction to the design, properties, and applications of matched filters is provided by Turin (1960). The continuous-time results can be translated in discrete-time in a straightforward manner.
5. Linear prediction is an area with rich theoretical background and many practical applications, ranging from speech processing to geophysical exploration. More detailed treatments of linear prediction and other parametric signal modeling techniques are given in Stoica and Moses (2005) and Manolakis et al. (2005). The applications of linear prediction to speech processing are discussed in Rabiner and Schafer (2010) and Makhoul (1975).
6. Thorough treatments of the KLT and its applications are given in Johnson and Wichern (2007) (statistics), Fukunaga (1990) (pattern recognition), Jayant and Noll (1984) (data compression), and Jain (1989) (image processing).

Review questions

1. Explain briefly what is the main difference between probability theory and statistics.
2. Explain the difference between an estimator and an estimate.
3. Which term is more accurate and why: "good estimate" or "good estimator"?
4. Describe briefly your understanding of a sampling distribution of an estimator.
5. Explain the meaning of bias and variance of an estimator.
6. Sketch a sampling distribution which is not meaningfully described by its mean and variance.
7. Explain how averaging increases the quality of noisy measurements.
8. Explain how the correlation of observations affects the quality of the sample mean estimator.
9. Explain briefly the concept of ergodicity and its significance in spectral estimation.
10. Can a nonstationary process be ergodic? Why?
11. Explain a strategy to decide how many values of the ACVS can be estimated with sufficient accuracy from N observations.
12. Explain why the values of $\hat{\gamma}[\ell]$ for ℓ close to N do not provide good estimates of the ACVS.
13. Describe briefly why the periodogram is a bad estimator of PSD.
14. Explain why averaging is the only way to improve the quality of the periodogram as a PSD estimator.
15. Describe briefly the Blackman–Tukey and Welch methods of PSD estimation.
16. Explain why reducing the variance of a PSD estimator increases its bias.
17. Describe the method of window closing.
18. What is the difference between a linear estimator and an affine estimator?
19. For what reason do we use linear estimators that minimize the mse criterion?
20. Explain why maximization of SNR is a meaningful criterion for signal detection applications.
21. Explain the origin of the term "matched filter." More specifically, describe what is matched to what and under what assumptions.
22. Explain why a matched filter operates on a block of the input signal at a time.
23. Describe briefly what we need to design an FIR Wiener filter.
24. Describe briefly the concept and properties of the KLT.
25. Explain the operation of the KLT by using the equal probability contours of a two-dimensional normal distribution.

Problems

Tutorial problems

1. The quality of the unbiased estimator $\hat{m}$ for uncorrelated observations is measured by its variance $\text{var}(\hat{m}) = \sigma^2/N$, where σ^2 is the variance of each observation and N the number of observations.

   (a) Generate and plot 1000 samples from the random variable $x \sim N(0, \sigma^2)$ for $\sigma = 2$. Estimate the pdf of the obtained data and compare to the true pdf by plotting them on the same diagram.

   (b) Generate K data sets $\{x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)}\}$ from $x_i^{(k)} \sim N(0, \sigma^2)$, where $\sigma = 2$, $N = 40$, and $K = 1000$.

   (c) Compute the mean $\hat{m}^{(k)}$, $k = 1, \ldots, K$ of each data set and plot the true mean m and the K sample means on the same plot.

   (d) Compute the mean and variance of the sample mean and compare to the theoretically expected values.

   (e) Estimate the pdf of the sample means and compare to the true pdf by plotting them on the same diagram. How is the pdf of the sample mean related to the pdf of the random variable generating the data? Do the simulations agree with theory?

   (f) Repeat (a)–(e) for $\sigma = \sqrt{2}$ and $N = 20$ and explain the results obtained.

2. Let x_k be a sequence of random variables, such that $x_k \sim \text{IID}(0, \sigma^2)$. Let S_N^2 be the *unbiased* sample variance estimator which is given by $S_N^2 = \frac{1}{N-1} \sum_{k=1}^N x_k^2$.

   (a) Show that

$$\text{var}(S_N^2) = \frac{1}{N}\left(m_4 - \frac{N-3}{N-1}\sigma^4\right),$$

   where $m_4 = E(x_k^4)$ is the fourth central moment.

   (b) Show that if $m_4 < \infty$ then the sample variance estimator is a consistent estimator.

   (c) Show that the covariance

$$\text{cov}(x_i - \hat{m}, x_j - \hat{m}) = -\frac{1}{N-1}, \quad \text{for } i \neq j.$$

3. Two random variables, x and y, have a joint density $f(x, y)$ given below:

$$f(x, y) = \begin{cases} 1/4, & 0 \le x \le 2,\ 0 \le y \le 1 \\ 1/4, & -1 \le x \le 0,\ -2 \le y \le 0 \\ 0. & \text{otherwise} \end{cases}$$

   (a) Determine the means m_x and m_y.

   (b) Determine the standard deviations σ_x and σ_y.

   (c) Determine the correlation r_{xy} and the correlation coefficient ρ_{xy} between x and y.

   (d) Verify your answers using MATLAB. To do this, generate N two-dimensional random numbers according to the joint density given above to obtain jointly distributed random numbers x and y. Use the MATLAB functions `mean`, `std`, `corr`, and `corrcoef`. Experiment using $N = 10^4$, 10^5, and 10^6.

4. Consider the following three data sets:

$$x_1[n] \sim \text{WGN}(0, 1), \quad 1 \le n \le 1000$$

$$x_2[n] = x_1[n] + 0.01n$$

$$x_3[n] \sim \begin{cases} \text{WGN}(0, 1), & 1 \le n \le 500 \\ \text{WGN}(1, 1), & 501 \le n \le 1000 \end{cases}$$

$$x_4[n] \sim \begin{cases} \text{WGN}(0, 1), & 1 \le n \le 500 \\ \text{WGN}(0, 2). & 501 \le n \le 1000 \end{cases}$$

(a) Estimate the mean and variance of each data set before plotting the data.

(b) Now plot the data sets and comment on the results after studying the plots.

5. Consider the following AR(1) process:

$$x[n] = a\,x[n-1] + w[n], \quad -1 < a < 1,$$

where $w[n] \sim \text{WN}(0, \sigma_w^2)$.

(a) Determine the mean m_x, variance σ_x^2, and the correlation coefficient $\rho_x[\ell]$.

(b) Let $a = 0.9$. Using the `randn` function with mean 0 and variance $\sigma_w^2 = 1$, generate $N = 100$ samples of the AR(1) process and estimate the mean using (14.19). Repeat the experiment 10 000 times and plot the histogram of the estimated mean.

(c) Repeat the above part (b) for $a = 0.1$.

(d) Comment on the results in parts (b) and (c) above.

6. Consider the following AR(1) process:

$$x[n] = a\,x[n-1] + w[n], \quad -1 < a < 1,$$

where $w[n] \sim \text{WN}(0, \sigma_w^2)$.

(a) Let $a = 0.9$. Using the `randn` function with mean 0 and variance $\sigma_w^2 = 1$, generate $N = 100$ samples of the AR(1) process and estimate the variance using (14.29). Repeat the experiment 10 000 times and plot the histogram of the estimated mean.

(b) Repeat the above part (a) for $a = 0.1$.

(c) Comment on the results in parts (a) and (b) above.

7. (a) Explain the steps and the meaning of the algorithm used in the function `acrsfft`.

(b) Write a MATLAB program to compare the computational complexity of functions `acrs` and `acrsfft` as a function of N and L.

8. (Daniell Method) Suppose that the data length N is large enough so that $I(\omega)$ is essentially an unbiased estimator of $S(\omega)$ and that $I(2\pi k_1/N)$ and $I(2\pi k_2/N)$ are pairwise uncorrelated for $k_1 \ne k_2$. If $S(\omega)$ is slowly varying in the neighborhood of a frequency, say ω_k, then we have $S(\omega_{k-Q}) \simeq \cdots \simeq S(\omega_k) \simeq \cdots \simeq S(\omega_{k+Q})$ for some integer $Q > 0$. Thus, the random variables $I(\omega_{k-Q}), \ldots, I(\omega_k), \ldots, I(\omega_{k+Q})$ are a set of $M = 2Q + 1$ unbiased and uncorrelated estimators of the same quantity, namely, $S_D(\omega_k)$. We can thus average them to produce an improved estimator

$$\hat{S}_D(\omega_k) = \frac{1}{2Q+1} \sum_{m=-Q}^{Q} I(\omega_{k-m}).$$

(a) Show that the estimator $\hat{S}_D(\omega_k)$ is unbiased, that is,

$$E[\hat{S}_D(\omega_k)] \simeq S(\omega_k).$$

(b) Show that the variance of the estimator is given by

$$\text{var}[\hat{S}_D(\omega_k)] = \frac{\text{var}[I(\omega_k)]}{2Q+1} \simeq \frac{S^2(\omega_k)}{2Q+1}.$$

(c) Determine the variance reduction factor for this method.

(d) Generate $N = 2048$ samples of the AR(2) process

$$x[n] = 0.75x[n-1] - 0.5x[n-2] + v[n],$$

where $v[n] \sim \text{WGN}(0, 1)$. Compute $I(\omega_k)$ and $\hat{S}_D(\omega_k)$ for $Q = 4, 8, 16,$ and 32. Comment on the resulting bias–variance trade-off.

9. Consider the harmonic process model given by

$$x[n] = \sum_{k=1}^{3} A_k \cos(\omega_k n + \phi_k) + v[n],$$

with $A_1 = A_2 = 1$, $A_3 = 0.2$, $\omega_1 = 0.4\pi$, $\omega_2 = 0.5\pi$, $\omega_3 = 0.7\pi$, and $\sigma_v^2 = 1$. The phases ϕ_1, ϕ_2, and ϕ_3 are IID random variables uniformly distributed over $[-\pi, \pi]$. In this problem we will investigate the window closing aspect of the Blackman–Tukey PSD estimate.

(a) Generate a realization of $N = 5000$ samples and compute the Blackman–Tukey PSD estimate using a lag window of lengths $L = 20, 30, \ldots$, i.e., in steps of 10 until the sinusoids are resolved and there is no significant change in the estimates. Plot the PSD for the resulting value of L.

(b) Repeat part (a) when $\omega_1 = 0.45\pi$.

(c) Comment on your results in parts (a) and (b).

10. Consider the AR(4) process described in Example 14.2:

$$x[n] = 2.7607x[n-1] - 3.8106x[n-2] + 2.6535x[n-3] - 0.9238x[n-4] + v[n],$$

where $v[n] \sim \text{WGN}(0, 1)$. Generate 1024 samples of $x[n]$.

(a) Compute a periodogram of $x[n]$ using a 1024-point FFT and plot it over $0 \leq \omega \leq \pi$.

(b) Compute the modified periodogram, based on a Bartlett data window, of $x[n]$ using a 1024-point FFT and plot it over $0 \leq \omega \leq \pi$.

(c) Comment on the bias aspect of your results in parts (a) and (b).

11. Consider the following harmonic process:

$$x[n] = \sum_{k=1}^{4} A_k \sin(\omega_k n + \phi_k) + v[n],$$

where $A_1 = 1$, $A_2 = 0.5$, $A_3 = 0.5$, $A_4 = 0.25$, $\omega_1 = 0.1\pi$, $\omega_2 = 0.6\pi$, $\omega_3 = 0.65\pi$, and $\omega_4 = 0.8\pi$. The phases $\{\phi_i\}_{i=1}^4$ are IID random variables uniformly distributed over $[-\pi, \pi]$. Generate 50 realizations of $x[n]$ for $0 \le n \le 255$.

(a) Compute the Welch estimate, using 75 percent overlap, Hamming window, and $L = 16$, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(b) Compute the Welch estimate, using 50 percent overlap, Hamming window, and $L = 16$, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(c) Compute the Welch estimate, using 50 percent overlap, Hann window, and $L = 16$, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(d) Provide a qualitative comparison between the above sets of plots.

12. Consider a stochastic process generated by the following systems:

$$v[n] = av[n-1] + w[n], \ 0 < a < 1$$
$$x[n] = x[n-1] + v[n],$$

where $w[n] \sim \text{WGN}(0, \sigma_w^2)$. Is the process $x[n]$ stationary? Why? Hint: Compute the mean and variance of $x[n]$.

(a) Generate and plot $N = 200$ samples of $x[n]$. How does the nonstationarity affect the shape of the signal waveform?

(b) Can you determine a PSD $S(\omega)$ for the process $x[n]$? Is the PSD finite for all values of ω? Why? Plot $S(\omega)$ for all finite values.

(c) Generate $N = 1000$ samples of $x[n]$, compute the Welch estimate $\hat{S}_x(\omega)$, and compare it to the theoretical PSD.

(d) Compute the Welch PSD estimate $\hat{S}_y(\omega)$ of the first difference signal $y[n] = x[n] - x[n-1]$. Then compute the PSD $\tilde{S}_y(\omega) = \hat{S}_y(\omega)/|1 - \exp(-j\omega)|^2$ for $\omega \ne 0$, and compare it to the theoretical PSD.

(e) Justify the procedure in the last step and compare the estimates $\hat{S}_x(\omega)$ and $\tilde{S}_y(\omega)$.

13. In this problem we want to constrain the linear estimator to be a constant, that is, $\hat{y} = b$. Determine the optimum value of b that minimizes $\text{E}[(y - \hat{y})^2]$.

14. In this problem we discuss in detail the matched filtering problem illustrated in Figure 14.18. The input to the matched filter is given by $x[n] = s_i[n] + v[n]$, where $v[n] \sim \text{WGN}(0, 1)$ and $s_i[n] = 0$ outside the interval $0 \le n \le p - 1$.

(a) Determine the impulse response of the matched filter and the output SNR. Explain why, in the absence of noise, the output is the ACRS of the desired signal.

(b) Suppose that $p = 10$ and $s_i[n] = \cos(2\pi n/10)$. Generate $N = 200$ samples of the noisy signal $x[n]$ and process it through the matched filter designed for $s_i[n]$. Plot the desired, input, and filtered signals and determine when the matched filter output is output.

(c) Repeat part (b) for the signal $s_i[n] = (1/\sqrt{10})\cos(2\pi n/10)$, $p = 100$.

(d) Which signal can be detected more easily by visual inspection of the matched filter output? Is this justified by comparing the output SNR in each case?

15. Consider an AR(1) signal $y[n] = 0.95y[n-1] + w[n]$ where $w[n] \sim \text{WN}(0,4)$. The signal $y[n]$ is contaminated by $v[n] \sim \text{WGN}(0,1)$ to obtain a noisy signal $x[n]$. We want to design a Wiener filter to estimate $y[n]$ from $x[n]$.
    (a) Determine the theoretical PSD $S_y(\omega)$ of the signal $y[n]$.
    (b) Using the PSD $s_v(\omega)$ of $v[n]$ and (14.120), design a Wiener filter function $H_o(\omega)$.
    (c) By sampling $H_o(\omega)$ at sufficiently dense points, obtain an FIR version of the Wiener filter $h_0[n]$.
    (d) Generate 1000 samples of the noisy signal $x[n]$ and process through the Wiener filter designed in (c) above. Plot both the noisy and filtered signals and comment on your results.

16. Using Parseval's theorem prove (14.121) and explain its meaning.

17. Consider a rigid rotation of axes counterclockwise, by an angle θ, as shown in Figure 14.26.
    (a) Using the distances and angles marked therein show that

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \text{ or } x = Qy.$$

    (b) Show that the transformation from the original to the rotated coordinates is given by

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ or } y = Q^T x.$$

    (c) Show that $Q^T Q = I$ or $Q^T = Q^{-1}$, that is, the rotation matrix Q is orthogonal.

18. Let a be a 3×1 vector and C be a 3×3 matrix given by

$$a = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T \text{ and } C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}.$$

    (a) Using (14.194) and direct calculations, show that

$$\frac{\partial a^T C a}{\partial a} = 2Ca.$$

    (b) Using (14.194) and direct calculations, show that

$$\frac{\partial a^T a}{\partial a} = 2a.$$

19. Let $x_1, x_2, \ldots, x_p$ be random variables and let $y_1, y_2, \ldots, y_p$ be their KLT coefficients. Show that

$$\sum_{k=1}^{p} \text{var}(x_k) = \sum_{k=1}^{p} \text{var}(c_k),$$

which is equivalent to the Parseval's relation for the DFT.

20. In this problem we discuss how to generate sample random vectors x from the normal distribution $N(m, C)$ using the KLT. Let A be the eigenvector matrix and Λ be the eigenvalue matrix of C, i.e. $CA = A\Lambda$.
    (a) If $z \sim N(0, I)$, then show that the linear transformation $y = \Lambda^{1/2}z$ results in $y \sim N(0, \Lambda)$.
    (b) Using the inverse KLT $x = Ay + m$, show that x has the desired distribution $N(m, C)$.
    (c) Write a MATLAB function
    ```
    X = Normal_ND(N,Mu,C)
    ```
    which generates an $N \times p$ matrix X whose rows are sample vectors from the pdf $N(m, C)$.
    (d) Suggest different approaches to check whether the generated data have the expected properties and illustrate their application with MATLAB.

21. Consider the speech signal depicted in Figure 14.22(a) and available in file `speech22.mat` on the web site.
    (a) Process the speech signal that produces the plot in Figure 14.22(b).
    (b) Compute and plot the periodogram estimate of the speech spectrum.
    (c) Obtain parameters of the all-pole model using $p = 6$ for the speech signal. Determine and plot the spectrum using this model.
    (d) Repeat part (c) using $p = 12$ and compare the results.

22. In this problem we elaborate on the geometrical interpretations of Karhunen–Loève transform discussed in Example 14.6 using $N = 100$ observations $x[n]$ from a zero mean bivariate normal distribution generated using the algorithm described in Tutorial Problem 20.
    (a) Generate a data set with $\sigma_1 = \sigma_2 = 1$, and $\rho = 0.4$ and plot their scatter diagram.
    (b) Compute the Karhunen–Loève transform $y[n]$ using MATLAB script `klt.m` and plot their scatter diagram.
    (c) Is the form of the scatter plots obtained what is theoretically and intuitively expected?
    (d) What is the approximation of mse if we drop the second coefficient?
    (e) Repeat (a)–(d) for $\rho = 0.9$. Which data set can be described more accurately if we retain only the first Karhunen–Loève transform coefficient? Justify your answer.

Basic problems

23. Consider the variance estimator given in (14.12).
    (a) Show that the mean of $\hat{\sigma}^2$ is given by (14.13).
    (b) If mean m of the random variable X is known and if we define the sample variance as $\hat{\sigma}^2 = \frac{1}{N} \sum_{k=1}^{N} (x_k - m)^2$, show that the mean of $\hat{\sigma}^2$ is given by σ^2/N.

24. Generate $K = 1000$ data sets of length $N = 50$ from a uniform distribution $x_i^{(k)} \sim U(0, 1)$.
    (a) Compute the mean and variance of the sample mean and compare to the theoretically expected values.
    (b) Plot the empirical pdf of the sample mean and explain its shape.

25. Show that $\sum_{k=1}^{N}(x_k - \hat{m})^2$ can be written as

$$\sum_{k=1}^{N}(x_k - \hat{m})^2 = \left[\sum_{k=1}^{N}(x_k - m)^2\right] - N(\hat{m} - m)^2,$$

where m is the mean of the random variables x_k.

26. Two random variables, x and y are jointly Gaussian with $m_x = 1$, $m_y = 2$, $\sigma_x^2 = 4$, $\sigma_y^2 = 1$, and $\rho = 0.8$. Generate $10\,000$ two-dimensional random numbers for this distribution (see Tutorial Problem 20) and compute the estimate $\hat{\sigma}_{xy}$. Repeat the simulation over 1000 data sets.

  (a) Determine the sample mean of the estimate $\hat{\sigma}_{xy}$.

  (b) Determine the sample variance of the estimate $\hat{\sigma}_{xy}$.

  (c) Plot a histogram of the estimate $\hat{\sigma}_{xy}$.

  (d) Comment on the statistical properties of the estimate $\hat{\sigma}_{xy}$.

27. In this problem we introduce Student's t-distribution and discuss some of its basic properties. Given two independent random variables $z \sim N(0, 1)$ and $v \sim \chi_v^2$, the t-distribution with v degrees of freedom and its pdf are defined by

$$x = \frac{z}{\sqrt{v/v}} \sim t_v,$$

$$f(x) = \frac{\Gamma[(v + 1)/2]}{\sqrt{v\pi}\,\Gamma(v/2)}\left(1 + \frac{x^2}{v}\right)^{-\frac{(v+1)}{2}},$$

respectively. The mean and variance of x are given by $E(x) = 0$ and $\mathrm{var}(x) = v/(v-2)$ for $v > 2$. For $v = 1$ the t-distribution is known as the Cauchy distribution.

  (a) Plot the pdf of t_v for $v = 1, 2, 5, 40$ and the pdf of z on the same plot. Compare the pdf of t_{40} to the pdf of z and explain the results.

  (b) Generate $N = 1000$ observations from the random variables in (a) and compare the empirical pdf to the theoretical pdf.

28. Repeat Problem 1 for observations derived from the random variable $x_i \sim t_v$ for (i) $v = 100$, and (ii) $v = 1$, and explain the results obtained.

29. Let $x[n]$ be a stationary white Gaussian noise process with zero-mean and unit variance. The theoretical pdf of $x[n]$ is $S_x(\omega) = \sigma_x^2 = 1$. We will study the periodogram estimates. Generate 50 different N-point records of $x[n]$ and compute the periodogram estimate of each record for $\omega_k = 2\pi k/1024$, $k = 0, 1, \ldots, 512$ by taking a 1024-point FFT.

  (a) For $N = 32$, compute 50 periodogram estimates. Plot periodogram overlays and the average of these overlays in two separate subplots.

  (b) For $N = 128$, compute 50 periodogram estimates. Plot periodogram overlays and the average of these overlays in two separate subplots.

  (c) For $N = 256$, compute 50 periodogram estimates. Plot periodogram overlays and the average of these overlays in two separate subplots.

  (d) Comment on the periodogram properties from the plots in the above three parts.

30. The Parzen window is given by (14.69).

(a) Show that its DTFT is given by (14.70).

(b) Using MATLAB, compute and plot the data window $w_P[\ell]$ and its frequency-domain response $W_P(\omega)$ for $L = 5$, 10, and 20.

(c) From the frequency-domain plots in part (b) experimentally determine the 3 dB mainlobe width $\Delta\omega$ as a function of L.

31. Let $\hat{\theta}_1$ and $\hat{\theta}_2$ be independent unbiased estimators of a parameter θ having variances σ_1^2 and σ_2^2, respectively. We now consider an estimator of the form $\hat{\theta} = a_1\hat{\theta}_1 + a_2\hat{\theta}_2$. Determine a_1 and a_2 so that $\hat{\theta}$ is unbiased and has minimum variance.

32. The measurements x_k for a constant parameter θ are corrupted by independent additive noise v_k as follows:

$$x_k = \theta + v_k, \quad \mathrm{E}(v_k) = 0, \quad \mathrm{var}(v_k) = \sigma_k^2.$$

Determine the coefficients $a_1, \ldots, a_N$ so that the linear estimator

$$\hat{\theta} = a_1 x_1 + a_2 x_2 + \cdots + a_N x_N$$

is unbiased and has minimum variance. Hint: Use the method of Lagrange multipliers.

33. Let X and Y be random variables with means m_x and m_y, variances σ_x^2 and σ_y^2, and covariance σ_{xy}.

(a) Let $\hat{X} = aY + b$ be an affine estimator of X given Y. Determine constants a and b that minimize the variance of $X - \hat{X}$. Did you encounter any difficulty?

(b) Now let $\hat{X} = aY + b$ be an affine estimator of X given Y with an additional assumption that $\mathrm{E}(\hat{X}) = am_y + b = \mathrm{E}(X) = m_x$. Determine constants a and b that minimize the variance of $X - \hat{X}$. Does your result coincide with the minimum mse?

34. Repeat Tutorial Problem 14 for $s_i[n] = 1/3, p = 9$ and $s_i[n] = 1/10, p = 100$.

35. Consider the two 10-point signals $x_0[n]$ and $x_1[n]$ given below:

$$x_0[n] = \{1, 1, 1, -1, -1, 0, 0, 0, 0, 0\},$$
$$x_1[n] = \{1, 1, 1, 1, -1, 0, 0, 0, 0, 0\}.$$

These signals are sent over a communication channel which adds white noise to the signals. Using a correlation-detector approach, we want to detect signals in white noise. Let $h_0[n]$ denote the matched filter for $x_0[n]$ and let $h_1[n]$ denote the matched filter for $x_1[n]$.

(a) Determine and plot the responses of $h_0[n]$ to $x_0[n]$ and $x_1[n]$. Repeat the same for $h_1[n]$. Compare these outputs at $n = 10$.

(b) You should notice that the output of the above matched filters at $n = 10$ can be computed as a correlation of the input and the impulse response. Implement such a structure and determine its output for each case in part (a) above.

(c) How would you modify the signal $x_0[n]$ so that the outputs of $h_0[n]$ to $x_1[n]$ and $h_1[n]$ to $x_0[n]$ are zero?

36. Consider the AR(4) process given by

$$x[n] = 2.7607x[n-1] - 3.8106x[n-2] + 2.6535x[n-3] - 0.9238x[n-4] + v[n],$$

where $v[n] \sim \mathrm{WGN}(0, 1)$ and generate a data set with 2048 observations of $x[n]$.

(a) Compute a Welch PSD estimate of $v[n]$ at 512 frequency values over $[0, \pi]$.
(b) Compute a Welch cross-PSD estimate between $v[n]$ and $x[n]$ at 512 frequency values over $[0, \pi]$.
(c) Now estimate and plot the frequency response $H(e^{j\omega})$ using the above two results.

37. Let $s_1[n]$ and $s_2[n]$ be short-length pulses of unit energy as given below:

$$s_1[n] = 1/3, \ 0 \le n \le 8 \quad \text{and} \quad s_2[n] = 0.1, \ 0 \le n \le 99$$

and zero everywhere. These pulses are observed in noise, i.e.

$$x_i[n] = s_i[n] + v[n], \ i = 1, 2$$

where $v[n] \sim$ WGN$(0, 1)$.
(a) Generate 200 samples of the noisy $x_1[n]$ signal and process it through the matched filter designed for $s_1[n]$. Plot the original, noisy, and filtered signals and determine when the matched filter output is maximum.
(b) Generate 200 samples of the noisy $x_2[n]$ signal and process it through the matched filter designed for $s_2[n]$. Plot the original, noisy, and filtered signals and determine when the matched filter output is maximum.
(c) Discuss your results in (a) and (b) above in terms of filtered waveforms and maximization of the output SNR.

38. Consider the backward linear predictor given in (14.158).
(a) Show that it can be obtained by (14.158a).
(b) Show that the resulting mse is given by (14.158b).
(c) Show that the backward linear-predictor mse is equal to the forward linear-predictor error.

39. Starting with the mse J_m in (14.134c) and then using the Levinson–Durbin recursion prove that J_m can be recursively computed using (14.149).

Assessment problems

40. A Gaussian voltage random variable x has a mean value of zero and variance equal to 9. The voltage x is applied to a square-law full wave diode detector with a transfer characteristic $y = 5x^2$.
(a) Determine mean and variance of the output voltage random variable y.
(b) Verify your answers using MATLAB. To do this generate N Gaussian random numbers (with appropriate mean and variance) to obtain x and then apply the square-law full wave diode detector operation on these numbers to obtain y. Experiment using $N = 10^4$, 10^5, and 10^6.

41. The quality of the sample variance estimator $\hat{\sigma}^2$ for uncorrelated observations is measured by its mean and variance.
(a) Generate K data sets $\{x_1^{(k)}, x_2^{(k)}, \ldots, x_N^{(k)}\}$ from $x_i^{(k)} \sim N(0, \sigma^2)$, where $\sigma = 2$, $N = 40$, and $K = 1000$.
(b) Compute the sample variance $\hat{\sigma}^2_{(k)}$, $k = 1, \ldots, K$ of each data set and plot the true variance σ^2 and the K sample variances on the same plot.

(c) Compute the mean and variance of the sample variance and compare to the theoretically expected values.

(d) Repeat (a)–(c) for $\sigma = \sqrt{2}$ and $N = 20$ and explain the results obtained.

42. Consider the harmonic process signal model

$$x[n] = \cos(0.44\pi n + \phi_1) + \cos(0.46\pi + \phi_2) + v[n], \quad 0 \le n \le 256$$

where ϕ_1 and ϕ_2 are IID random variables uniformly distributed over $[-\pi, \pi]$ and $v[n] \sim \text{WGN}(0, 2)$.

(a) Estimate the PSD using the periodogram and plot the spectrum.

(b) Estimate the PSD using the modified periodogram with Bartlett window and plot the spectrum.

(c) Estimate the PSD using the Blackman–Tukey method with Parzen window and $L = 32$ and plot the spectrum.

(d) Which method performs best in terms of signal resolution?

43. Consider the following harmonic process:

$$x[n] = \sum_{k=1}^{4} A_k \sin(\omega_k n + \phi_k) + v[n],$$

where $A_1 = 1$, $A_2 = 0.5$, $A_3 = 0.5$, $A_4 = 0.25$, $\omega_1 = 0.1\pi$, $\omega_2 = 0.6\pi$, $\omega_3 = 0.65\pi$, and $\omega_4 = 0.8\pi$. The phases $\{\phi_i\}_{i=1}^{4}$ are IID random variables uniformly distributed over $[-\pi, \pi]$. Generate 50 realizations of $x[n]$ for $0 \le n \le 256$.

(a) Compute the Blackman–Tukey estimates for $L = 32$, 64, and 128, using the Bartlett lag window. Plot your results, using overlay and averaged estimates. Comment on your plots.

(b) Repeat part (a), using the Parzen window.

(c) Provide a qualitative comparison between the above two sets of plots.

44. Consider the random process given in Problem 43.

(a) Compute the Bartlett estimate, using $L = 16$, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(b) Compute the Welch estimate, using 50 percent overlap, Hamming window, and $L = 16$, 32, and 64. Plot your results, using overlay and averaged estimates. Comment on your plots.

(c) Provide a qualitative comparison between the above two sets of plots.

45. In this problem we investigate the minimum mse estimate of random variable X from the observation of another related random variable Y. The mse is defined as $\text{MSE} = E[(X - \hat{X})^2]$.

(a) If X and Y are arbitrary (nonGaussian) random variables the minimum mse estimate of X given the observation of Y is the conditional mean, that is, $\hat{X} = E[X|Y]$.

(b) If X and Y are jointly Gaussian with nonzero means, then show that the minimum mse estimate is an affine function of Y.

(c) If X and Y are jointly Gaussian with zero means, then show that the minimum mse estimate is a linear function of Y.

46. Since $\hat{S}_W(\omega)$ is a PSD estimate, one can obtain a biased estimate $\hat{r}_x[\ell]$, $|\ell| < L$, of the ACRS of $x[n]$ from Welch's method as

$$\hat{r}_x[\ell] \triangleq \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{S}_W(\omega) e^{j\omega\ell} d\omega.$$

Let $\tilde{r}_x[\ell]$ be obtained by taking the N_{FFT}-point IDFT of the N_{FFT}-samples $\hat{S}_W[k]$ of the continuous spectrum $\hat{S}_W(\omega)$ around the unit circle.
(a) Show that $\tilde{r}_x[\ell]$ is an aliased version of the ACRS estimate $\hat{r}_x[\ell]$.
(b) If the length of the overlapping segments in Welch's method is L, how should N_{FFT} be chosen to avoid aliasing in $\tilde{r}_x[\ell]$.

47. Consider an AR(1) signal $x[n] = 0.9x[n-1] + w[n]$ where $w[n] \sim \text{WN}(0,1)$. It is smeared by a lowpass filter $H(z) = 1/1 - 0.9z^{-1}$ to obtain $v[n]$.
(a) Generate 1000 samples of $x[n]$ and $v[n]$. Using Welch's method, obtain their PSD and plot them in one figure.
(b) Develop an inverse filter to obtain a restoration $\hat{x}_1[n]$ of $x[n]$ from $v[n]$ and plot the original and restored signal.
(c) The smeared signal $v[n]$ is further contaminated by a WGN $z[n]$ of variance 0.01 to obtain a noisy signal $y[n]$. Using the inverse filter in (b), process $y[n]$ to obtain restored signal $\hat{x}_2[n]$. Plot the original signal $x[n]$ and restored signal $\hat{x}_2[n]$. Comment on your results.
(d) Using the Wiener filter theory, develop an optimum filter to extract $v[n]$ from $y[n]$. Filter $y[n]$ through the optimal filter and then through the inverse filter to obtain $\hat{x}_3[n]$. Plot the original signal $x[n]$ and restored signal $\hat{x}_3[n]$. Comment on your results.

48. The estimate of the autocorrelation matrix of a 4×1 zero mean random vector x is given by $r_{ij} = 0.95^{|i-j|}$, $0 \le i, j \le 3$.
(a) Determine the KLT y of x.
(b) Compare the basis vectors of the KLT with the basis vectors of the 4-point DFT and DCT.
(c) Compare the performance of KLT, DFT, and DCT by plotting the basis restriction error J_m as a function of m, where

$$J_m \triangleq \left(\sum_{k=m}^{3} \sigma_k^2 \right) / \left(\sum_{k=0}^{3} \sigma_k^2 \right),$$

for $m = 0, 1, 2, 3$ and $\sigma_k^2 = \text{E}(|c_k|^2) = \lambda_k$.

49. In radar signal processing, matched filters are used extensively that take advantage of correlation functions. Let $x[n]$ be a signal transmitted by radar. It is reflected by a target and subsequently is received by the radar receiver (after A/D conversion) as a scaled and delayed signal $y[n] = \alpha x[n-k]$, where the delay k is proportional to the distance to the target.
(a) Show that the cross-correlation $r_{yx}[\ell]$ reaches its maximum at $\ell = k$, that is, $r_{yx}[k] = \max_{\ell} r_{yx}[\ell]$.
(b) Propose a matched filter implementation to determine the delay k.

50. A filter with system function

$$H(z) = \frac{1 - 0.1z^{-1} - 0.72z^{-2}}{1 - 0.9z^{-1} + 0.81z^{-2}}$$

is excited by the process $x[n] \sim \text{WGN}(0, 1)$.
   (a) Determine whether the output process $y[n]$ is wide-sense stationary.
   (b) Find the theoretical values of the mean, variance, and ACRS of the output process.
   (c) Excite the filter with $N_0 = 5000$ samples of the input process and compute the output sequence.
   (d) Use the last $N = 4000$ samples (why?) to estimate the mean, variance, and ACRS of the output process and compare them to the theoretically expected values.
   (e) Repeat (c) and (d) for the input process $x[n] \sim U(0, 1)$. Does the distribution of the input have a strong effect upon the results?
   (f) Compute and plot the histograms of the two output sequences and explain the results obtained.

51. The system $H(z)$ in Problem 50 is excited by $x[n] \sim \text{WGN}(0, 1)$.
   (a) Compute and plot the true PSD $S_y(\omega) = \sigma^2 |H(e^{j\omega})|^2$.
   (b) Compute and plot $r_y[\ell]$, for $-100 \le \ell \le 100$.
   (c) Compute $E[I_y(\omega)]$ from (14.53) assuming a rectangular window of length N. Plot the mean periodogram for $N = 32, 64, 128$, and 256 and compare with the true PSD.
   (d) Compute and plot the bias $B(\omega) = S_y(\omega) - E[I_y(\omega)]$ for $N = 32, 64, 128$, and 256 and justify that the bias decreases with increasing N. Why is the bias larger at the peaks of the PSD?

52. Show the following relationship:

$$c_{k+1,m+1} = c_{km} + x[N_1 - k]x[N_1 - m] - x[N_2 + 1 - k]x[N_2 + 1 - m].$$

   (a) What is the form in the windowing method?
   (b) Use the simplified form derived in (a) to speed-up the computation of C in the no-windowing method. What is the computational saving? Write a MATLAB function that uses the results of (a).

53. Using the Levinson–Durbin algorithm prove the recursion (14.160) for the backward prediction error e_m^b.

54. Prove that the minimization of (14.166) leads to (14.168) and (14.169).

Review problems

55. In this problem we analyze the spectral characteristics of quantization noise in a signal $x[n]$ generated by the following AR(4) process:

$$x[n] = 2.7607x[n-1] - 3.8106x[n-2] + 2.6535x[n-3] - 0.9238x[n-4] + v[n],$$

where $v[n] \sim \text{WGN}(0, 1)$. Consider a segment $x[n]$ consisting of $N = 2048$ samples from this process. The signal $x[n]$ is quantized to B bits using a quantizer to obtain

$x_q[n]$ and the quantization error $e[n] = x_q[n] - x[n]$ is computed. We want to examine the PSD of $e[n]$ by computing its ACRS. The procedure is as follows:

- Estimate the mean of $e[n]$ and subtract it from $e[n]$.
- Estimate the ACRS $\hat{r}_e[\ell]$ for $0 \le \ell \le 512$ and normalize to obtain $\hat{\rho}_e[\ell] = \hat{r}_e[\ell]/\hat{r}_e[0]$.
- Window the normalized ACRS using a Bartlett window of size M and then compute the PSD estimate by taking the DFT.

(a) For a $B = 8$ bit quantizer use an $M = 512$ length window to estimate the normalized ACRS and the PSD of the quantization noise. Plot both the ACRS and the PSD and comment on the results.

(b) Repeat the above part (a) using $B = 8$ bits but $M = 50$ and comment on the resulting plots.

(c) Repeat parts (a) and (b) using $B = 3$ bits and comment on the resulting plots.

56. The method of Bartlett–Welch can be easily modified to compute estimates of the cross-PSD $S_{xy}(\omega)$ between processes $x[n]$ and $y[n]$. The key formula, which is a generalization of (14.86) and (14.87), is given by

$$\hat{S}_{xy}(\omega) = \frac{1}{ML} \sum_{m=1}^{M} \left\{ \left[\sum_{n=0}^{L-1} w[n]x_m[n]e^{-j\omega n} \right] \left[\sum_{n=0}^{L-1} w[n]y_m[n]e^{-j\omega n} \right]^* \right\},$$

where $w[n]$ is a data window of length L and $x_m[n]$ and $y_m[n]$ are 50% overlapped segments of $x[n]$ and $y[n]$, respectively, and M is the number of segments. Note that, in general, the cross-PSD is a complex function.

(a) Write a MATLAB function

```
Sxy = cpsd(x,y,Nfft,window(L))
```

that computes the cross-PSD estimates at `Nfft` frequencies around the unit circle.

(b) Let the input $x[n]$ be 1024 samples of a WGN$(0, 1)$ process, which is applied as an input to a filter given by

$$H(z) = \left(1 - 1.2728_z^{-1} + 0.81_z^{-2}\right)^{-1}$$

to obtain $y[n]$. Using your `cpsd` function compute and plot the magnitude of the cross-PSD. Use $L = 32$, Hann data window, and `Nfft=256`. Comment on your results.

(c) Repeat part (b) above with $L = 64$ and comment on your results.

57. This problem uses the signal file f16.mat containing samples of cockpit noise. We want to analyze this signal in terms of its ACRS and its spectral characteristics.

(a) Compute and plot the ACRS estimate of the noise process.

(b) Fit an $AR(p)$ model to the date for $p = 2$ and $p = 4$ and estimate model parameters.

(c) Compute and plot the PSD estimate using the AR(2) and AR(4) models.

(d) Compute and plot the periodogram PSD estimate of the noise process.

(e) Compute the Bartlett PSD estimate using $L = 32$, 64, and 128. Plot your result of the averaged estimate.

(f) Compute the Blackman–Tukey PSD estimate using $L = 32$, 64, and 128 using the Bartlett lag window. Plot your result of the averaged estimate.

(g) Compute the Welch PSD estimate using 50% overlap, Hamming window, and $L = 32$, 64, and 128. Plot your result of the averaged estimate.

(h) Compare the plots in the above four parts and comment on your observation.

58. Consider the 256×256 image, Building, to answer the following parts. It is available on the website. Let $u[m, n]$ denote this image.

(a) Determine the 2-D DCT of the image $u[m, n]$. Retain the 8192 largest (magnitude-wide) elements of this transform and set all other elements to zero. Determine the reconstructed image $\hat{u}[m, n]$. Compute the signal-to-noise ratio (SNR) of the reconstructed image as

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum\limits_{m=0}^{N-1} \sum\limits_{n=0}^{N-1} u^2[m, n]}{\sum\limits_{m=0}^{N-1} \sum\limits_{n=0}^{N-1} e^2[m, n]} \right),$$

where $e[m, n] = u[m, n] - \hat{u}[m, n]$. Visually compare the original image $u[m, n]$ and its reconstruction $\hat{u}[m, n]$.

(b) Partition the 256×256 image into sixty-four 32×32 sub-images. These sixty-four sub-images can be viewed as sample functions of a 32×32 random field. Approximate each 32×32 block by 128 terms in the KLT so that the mse is minimum. Calculate this mse using the reconstructed image.

(c) Discuss the visual comparison between the above transforms for the Building image.

15 Finite wordlength effects

In theory, all signal samples, filter coefficients, twiddle factors, other quantities, and the results of any computations, can assume any value, that is, they can be represented with infinite accuracy. However, in practice, any number must be represented in a digital computer or other digital hardware using a finite number of binary digits (bits), that is, with finite accuracy. In most applications, where we use personal computers or workstations with floating point arithmetic processing units, numerical precision is not an issue. However, in analog-to-digital converters, digital-to-analog converters, and digital signal processors that use fixed-point number representations, use of finite wordlength may introduce unacceptable errors. Finite wordlength effects are caused by nonlinear operations and are very complicated, if not impossible, to understand and analyze. Thus, the most effective approach to analyze finite wordlength effects is to simulate a specific filter and evaluate its performance. Another approach is to use statistical techniques to derive approximate results which can be used to make educated decisions in the design of A/D converters, D/A converters, and digital filters. In this chapter we discuss several topics related to the effects of finite wordlength in digital signal processing systems.

Study objectives

After studying this chapter you should be able to:

- Understand the implications of binary fixed-point and floating-point representation of numbers for signal representation and DSP arithmetic operations.
- Understand how to use a statistical quantization model to analyze the operation of A/D and D/A converters incorporating oversampling and noise shaping.
- Understand the effects of finite precision arithmetic on the operation of digital filters and FFT algorithms and use model-based predictions to make educated decisions during system design processes.

Number representation

· ·

We are all familiar with the decimal (or base-10) number system, where each number is represented as a string of digits (0 through 9) with a decimal point; the value of each digit depends on its position relative to the decimal point. Thus, the value of a decimal number is determined as follows:

$$124.325_{(10)} = 1 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 3 \times 10^{-1} + 2 \times 10^{-2} + 5 \times 10^{-3}.$$

In computer systems or digital circuits, numbers are represented by a string of binary digits or bits that take the value 0 or 1 (see Section 1.1.2). If we denote by $\triangle$ the binary point, the decimal value of the binary number $10_\triangle 101_{(2)}$ is given by

$$10_\triangle 101_{(2)} = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 2.625_{(10)}.$$

The location of the decimal or binary point, which is assumed fixed, determines the value of each bit. Therefore this representation of numbers is called *fixed-point format*. The bits of a binary number are organized into groups of 8 bits called bytes or groups containing 16 or 32 or 64 bits called words.

15.1.1   Binary fixed-point number representation

An arbitrary real number x can be represented in binary format with infinite precision. The binary representation and its decimal value are given by

$$x = \pm(\cdots b_{-2}b_{-1}b_0{}_\triangle b_1b_2b_3b_4\cdots)_{(2)} = \pm\left(\sum_{i=-\infty}^{\infty} b_i\, 2^{-i}\right)_{(10)}. \qquad (15.1)$$

In analog systems, we represent directly the value of x by a voltage; the major limitation in this representation is additive noise. In digital systems we use a finite number of bits and we represent the value of each bit b_k by a voltage (see Section 1.1.2); the truncation of the infinite series (15.1) leads to loss of numerical precision.

Sign and magnitude format Suppose now that we use a binary word with $(B+1)$ bits, say

$$\hat{x}_B = b_0b_1b_2\cdots b_B. \qquad (15.2)$$

To handle negative numbers we use the leftmost bit to indicate the algebraic sign of the number. If the sign bit b_0 is zero, the number is positive, otherwise it is negative; the remaining bits represent the absolute value of the number. This representation is known as *sign and magnitude format*. The leftmost bit, b_1, is called the *most significant bit (MSB)*, and the rightmost bit, b_B, is called the *least significant bit (LSB)*. The binary point does not exist physically in the computer. Simply, the logic circuits in the computer are designed

so that the computations result in numbers that correspond to the assumed binary point location.

A very important observation is that with $(B + 1)$ bits we can represent 2^{B+1} different numbers. If we assume that the binary point is on the right of b_B, we can use (15.2) to represent all integers from $-(2^B - 1)$ to $(2^B - 1)$, including zero. These are altogether $2^{B+1} - 1$ numbers, because zero has two representations, namely $+0$ and -0. However, we do not have to use these $2^{B+1} - 1$ numbers to represent integers; they can represent any set of $2^{B+1} - 1$ equally spaced numbers. For example, if we move the binary point after the Eth bit, that is, we use the format

$$\hat{x} = b_0 \overbrace{b_1 b_2 \cdots b_E}^{E \text{ bits}} {}_\triangle \overbrace{b_{E+1} \cdots b_B}^{(B-E) \text{ bits}}, \tag{15.3}$$

we can represent numbers from $-(1 - 2^{-B})2^E$ to $(1 - 2^{-B})2^E$; the distance between successive numbers is constant at $2^E 2^{-B}$. We say that the binary representation (15.3) has *range* $\mathcal{R}$ and *resolution* Δ given by

$$\mathcal{R} = \{-(1 - 2^{-B})2^E \le \hat{x} \le (1 - 2^{-B})2^E\}, \tag{15.4a}$$

$$\Delta = 2^E 2^{-B} = 2^{-(B-E)}. \tag{15.4b}$$

We note that there is a trade-off between range and resolution, which is controlled by the location of the binary point. For example, we can represent any number within the range (15.4a) by properly scaling a fraction $|\hat{x}_F| \le 1$ as follows:

$$\hat{x}_F \triangleq b_0 {}_\triangle b_1 b_2 \cdots b_B \quad \Rightarrow \quad \hat{x} = 2^E \hat{x}_F. \tag{15.5}$$

Mixed numbers and integer numbers are difficult to multiply and the number of bits cannot be reduced by rounding or truncation. Thus, in fixed-point digital signal processors we assume that we are dealing only with fractions, that is, all numbers are less than one in magnitude.

Two's-complement format   Fixed-point DSP processors use the two's complement to represent signed numbers because it has a single representation for zero and it allows us to perform addition and subtraction using the same hardware. A *two's complement representation* of a binary fractional number with one sign bit and B fractional bits is given by

$$\hat{x}_B = b_0 {}_\triangle b_1 \cdots b_B \triangleq -b_0 + \sum_{i=1}^{B} b_i 2^{-i}. \tag{15.6}$$

The three-bit two's complement numbers and their corresponding decimal values from (15.6) are given by:

$$\begin{aligned} b_0 b_1 b_2 : \quad & 011 \quad 010 \quad 001 \quad 000 \quad 111 \quad 110 \quad 101 \quad 100 \\ \hat{x}_B : \quad & 3/4 \quad 1/2 \quad 1/4 \quad 0 \quad -1/4 \quad -1/2 \quad -3/4 \quad -1 \end{aligned} \tag{15.7}$$

To find the representation of a negative number, we subtract its absolute value from 2, and then convert into binary; since the absolute value is less than one, the first bit will always be one. For example, the two's complement of -0.625 is obtained by $2-0.625 = 1.375$ whose binary representation is 1011. To obtain the binary representation of a decimal fraction, we repeatedly multiply by 2 and remove the integer part until we reach a fraction part zero. Thus, for $x = 0.375$ we have

$$
\left.
\begin{aligned}
2 \times 0.375 &= 0.75 &\to 0 \\
2 \times 0.75 &= 1.50 &\to 1 \\
2 \times 0.50 &= 1.00 &\to 1
\end{aligned}
\right\} \Rightarrow 011_{(2)}.
\tag{15.8}
$$

The representation of $x = 1.375$ is obtained by attaching 1 before the leftmost bit.

The largest two's complement number $011\ldots1$ has value $(1 - 2^B)$ and the smallest number $100\ldots0$ has value -1. Furthermore, zero is represented by $00\ldots0$ and there is one more negative number (-1) than there are positive numbers. Thus, the range of fractions in two's complement representation is

$$
-1 \le \hat{x}_{\mathrm{B}} \le 1 - 2^{-B}.
\tag{15.9}
$$

Numbers outside this range can be represented using the scaling formula

$$
-X_{\mathrm{m}} \le \hat{x} = X_{\mathrm{m}}\hat{x}_{\mathrm{B}} \le X_{\mathrm{m}}(1 - 2^{-B}), \quad X_{\mathrm{m}} \triangleq 2^E.
\tag{15.10}
$$

The resolution of the scaled numbers $\hat{x}$ is scaled by a factor X_m, that is,

$$
\Delta = X_{\mathrm{m}} 2^{-B}.
\tag{15.11}
$$

15.1.2 Quantization process

The infinite-precision analog signal value must be converted to a finite precision binary number through a process called quantization (see Section 6.5.1). The quantization of a number to $(B+1)$ bits can be done using either the operation of *rounding* or *truncation*; the corresponding nonlinear input–output characteristic functions are illustrated in Figure 15.1 (and also see Figure 6.24). The quantization error is given by

$$
e = Q[x] - x,
\tag{15.12}
$$

where $Q(\cdot)$ is a quantizer function. The smallest difference between numbers is determined by the least significant bit. Thus, the resolution of the quantizers (or *quantization step*) is given by (15.11) or

$$
\Delta = X_{\mathrm{m}} 2^{-B}.
$$

The range of quantization error is $-\Delta/2 < e \le \Delta/2$ for rounding and $-\Delta \le e < 0$ for truncation, as long as $|x| < X_{\mathrm{m}}$. If a number x is larger than X_{m} the scaled number
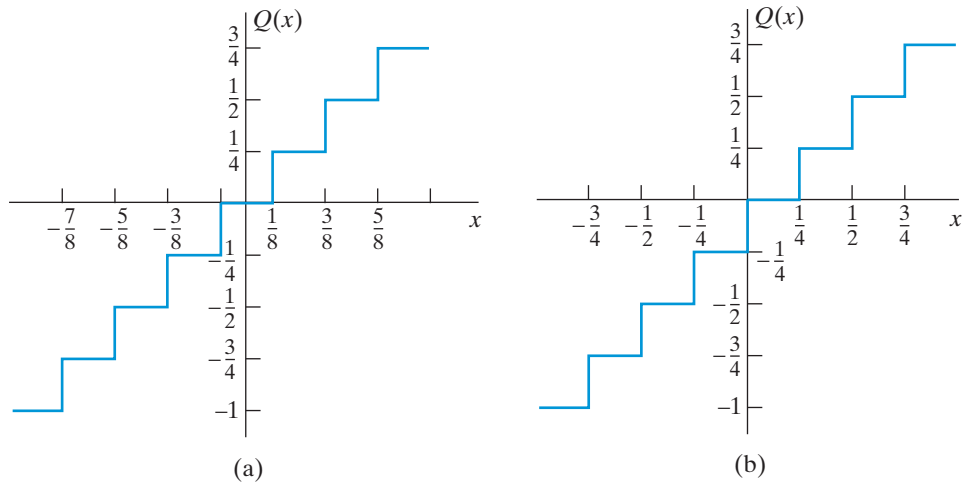
Figure 15.1 Input–output characteristics of a two's complement quantizer using (a) rounding, and (b) truncation, for $B = 2$ bits.

x/X_m is outside the range (15.4a) of the quantizer, a condition known as *overflow*. Overflows arise when a signal exceeds the range of the A/D converter or when the result of an arithmetic operation requires more than $(B + 1)$ bits for its representation. For example, consider the following addition and multiplication operations in binary and decimal formats:

$$
\begin{array}{cccc}
\ 0.1101 & \ 0.8125 & \ 0.1101 & \ 0.8125 \\
+\ \ 0.1001 & +\ 0.5625 & \times\ 0.1001 & \times\ 0.5625 \\
\hline
01.0110 & 1.3750 & 0.01110101 & 0.45703125
\end{array}
\tag{15.13}
$$

We note that adding two $(B+1)$ bit numbers results in a $(B+2)$ bit number; the additional bit appears *before* the binary point. When we multiply two $(B+1)$ bit fractions the product, which is still a fraction within the range (15.4a), has length of $(2B + 1)$ bits; the extra B bits appear *after* the binary point.

The product of two fractions is always within the range of the quantizer but it may fall in the interval between two quantization steps. Thus, reducing the number of bits from $(2B + 1)$ to $(B + 1)$ by rounding or truncation results in an approximation error bounded by the quantization step.

Clearly when we add or subtract two arbitrary numbers the result may be larger than 1, which is an overflow and will cause an incorrect result. For example,

$$
\begin{array}{llll}
0.5625 & +\ 0.6875 & = 1.250 & \\
010010 & +\ 0100110 & = 101000 & \rightarrow\ -0.75,
\end{array}
\tag{15.14}
$$
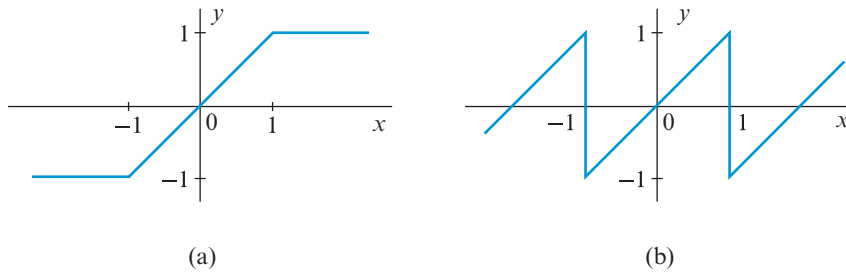
Figure 15.2 Overflow characteristics for (a) saturation, and (b) overflow.

which of course is incorrect. However, if the final result of a long series of additions is within range, we get the correct answer even if some intermediate results are outside the range. This important property is illustrated in the following example:

$$
\begin{array}{rll}
0.87500 & 011100 & \\
+\quad 0.40625 & 001101 & \\
\hline
1.28125 & 101001 & \rightarrow -0.71875 \text{ (incorrect)} \\
+\quad -0.34375 & 110101 & \\
\hline
0.93750 & 011110 & \rightarrow \quad 0.99750 \text{ (correct final result).} \qquad (15.15)
\end{array}
$$

We note that the error caused by overflow is in principle unlimited. One way to handle overflow in fixed-point DSP applications is to use saturation arithmetic, which "clips" the result at a maximum value using the nonlinear characteristic shown in Figure 15.2(a). Another useful approach is to use the "sawtooth" overflow characteristic shown in Figure 15.2(b). We note that the saturation approach voids the property illustrated by (15.15).

The most effective technique for preventing overflow is proper scaling of numbers to control their dynamic range and use of double length accumulators to store intermediate results. This demands greater coding and debugging efforts to optimally combine scaling factors, double accumulators, and saturation arithmetic.

MATLAB functions for fixed-point number representation The basic computational engine in MATLAB does not use fixed-point arithmetic but instead uses a 64-bit floating-point arithmetic (see Section 15.1.3) with internal 80-bit format for greater accuracy and provides results in decimal numbers. However, it provides several useful functions for simulating fixed-point arithmetic using binary numbers. The `dec2bin` function converts a *positive decimal integer* into a binary bit representation which is a character code but not a number. Similarly, the `bin2dec` converts a binary representation back to the corresponding positive decimal integer.

Using the scaling operation (15.10), and the quantization operation (rounding or truncation), we can use MATLAB's native functions to convert any decimal number into an equivalent fixed-point binary number for a given wordlength in bits. Figure 15.3 shows the function `[beq,E,B] = dec2beqR(d,L)` that converts decimal numbers in the array `d` into binary equivalent decimal numbers `beq` given `L` bits, using the rounding operation. The

```
function [beq,E,B] = dec2beqR(d,L)
% Binary equivalent decimal (beq) of decimal numbers
% in d for L=1+E+B bits using the Rounding operation:
%   [beq,E,B] = dec2beqR(d,L)
%    d can be scalar or vector or matrix.
%    L must be >= the number of integer bits required.
dm = abs(d);
E = max(max(0,fix(log2(dm(:)+eps)+1))); % Integer bits
B = L-1-E; % Fractional bits
beq = round(dm./(2^E).*(2^(L-1))); % Rounding to L bits
beq = sign(d).*beq*(2^(-B)); % 1+E+B bit representation
```

Figure 15.3 MATLAB function to convert decimal numbers into equivalent binary numbers using the rounding operation.

```
function [beq,E,B] = dec2beqT(d,L)
% Binary equivalent decimal (beq) of decimal numbers
% in d for L=1+E+B bits using the Truncation operation:
%   [beq,E,B] = dec2beqT(d,L)
%    d can be scalar or vector or matrix.
%    L must be >= the number of integer bits required.
dm = abs(d);
E = max(max(0,fix(log2(dm(:)+eps)+1))); % Integer bits
B = L-1-E; % Fractional bits
beq = fix(dm./(2^E).*(2^(L-1))); % Rounding to L bits
beq = sign(d).*beq*(2^(-B)); % 1+E+B bit representation
```

Figure 15.4 MATLAB function to convert decimal numbers into equivalent binary numbers using the truncation operation.

function uses one bit for sign designation, determines the minimum number of integer bits B required for all numbers in d, and then assigns the remaining $L-1-E$ bits to the fractional part in B. A similar function, using the truncation operation, is shown in Figure 15.4. As an example, consider $x=[-1.8184, 2.426]$. Then its 6-bit representation, using rounding, is

```
>> [xq,E,B] = dec2beqR(x,6)
xq =
   -1.8750    2.3750
E =
     2
B =
     3
```

which requires 2 integer bits and hence 3 bits are allocated to the fractional part.

15.1.3 Floating-point representation

One weakness of fixed-point arithmetic is a continuous need for scaling, which amounts to keeping track of E or the location of the binary point. An alternative to fixed-point representation that largely overcomes this drawback is the *floating-point representation*. This form corresponds to scientific notation, the first part of the word being used to store the fraction, and the rest to store the exponent.

MATLAB and most technical computing environments use floating-point arithmetic based on the ANSI/IEEE Standard 754-1985; see Goldberg (1991) and Press *et al.* (2007) for details. In this representation floating point numbers are normalized so that we can express
them as

$$\hat{x} = \pm(1+f) \times 2^E, \tag{15.16}$$

where f, $0 \leq f < 1$, is the fraction or mantissa and E is the exponent. For example, double precision floating-point numbers are stored in 64 bit words, with 1 bit for the sign, 52 bits for the mantissa, and 11 bits for the exponent. We emphasize that the floating-point representation does *not* allow for more numbers to be represented than the 2^{B+1} states of the $(B+1)$ bit word. The numbers are, however, spread out nonuniformly so that the resolution is proportional to the absolute value. This makes sense because in numerical analysis, it is the relative error size that matters, see Hamming (1973).

The fundamental difference between fixed-point and floating-point implementations is what information is stored about the number and who (the programmer or the processor) is responsible for its tracking and manipulation. In fixed-point processors, the programmer must take care of all these details; in floating-point processors, all these details are managed by the hardware. As a result, floating-point processors are more expensive than their fixed-point counterparts, but easier to program.

15.2 Statistical analysis of quantization error

Analog-to-digital (A/D) and digital-to-analog (D/A) converters provide the necessary interface between the analog and digital worlds. The quality of the A/D conversion is often the critical limiting factor in overall system performance; therefore, it is important to understand and predict the performance of A/D converters.

As we discussed in Chapter 6, A/D conversion involves four operations: pre-filtering, sampling, quantizing, and coding. Since we use an antialiasing filter, the performance of the A/D converter is predominantly limited by quantization error. An analysis of quantization error for sinusoidal signals has been presented in Section 6.5. In this section, we develop a statistical model for the quantization error and we demonstrate how to select the parameters of practical A/D converters based on model-based performance predictions.

A quantizer may be viewed as a staircase approximation to the linear identity operation $y = x$, as illustrated in Figure 15.1. To provide a general mathematical definition of the
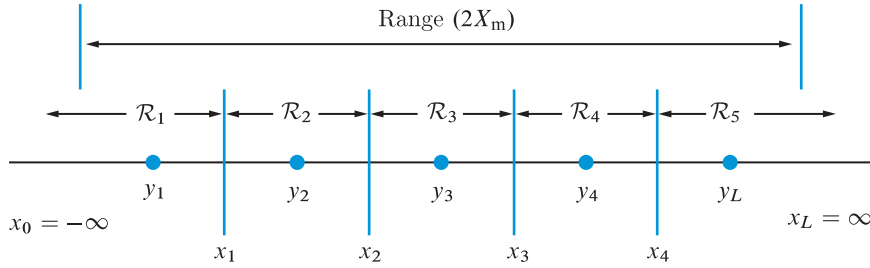
Figure 15.5 Quantization principles.

quantization process, we consider the notation in Figure 15.5. An L-point quantizer is defined by specifying a set of $L+1$ *decision levels* $x_0, x_1, \ldots, x_L$ and a set of L *quantization levels* $y_1, y_2, \ldots, y_L$. When the value x of the input sample lies in the ith *quantization interval*, namely

$$\mathcal{R}_i \triangleq \{x_{i-1} < x < x_i\}, \tag{15.17}$$

the quantizer produces the output value y_i. Since all values in $\mathcal{R}_i$ are represented by a single value y_i, quantization is a nonreversible process that results in loss of information. The end decision levels are usually set to $x_0 = -\infty$ and $x_L = \infty$ to accommodate signals with unbounded amplitudes; all other decision and quantization levels are finite. If $L = 2^{B+1}$, assigning a unique $(B + 1)$-bit word to each quantization level, results in a $(B + 1)$-bit quantizer. The quantization error is given by

$$e = y - x = Q(x) - x. \tag{15.18}$$

When the input sample lies within the interval $x_1 < x < x_{L-1}$, the error is called *granular noise* and is bounded in magnitude. When the input is outside this interval, the error is known as *overload distortion* and the amplitude is unbounded. The inherent nonlinearity of quantizers makes their analysis extremely difficult. However, a tractable and useful analysis is possible if we treat the quantization error as a random process and we use statistical techniques.

　To develop a statistical model for the quantization error we assume that the input sequence is a wide-sense stationary process with zero mean, standard deviation σ_x^2, and probability density function $f_X(x)$. The variance of quantization error is

$$\sigma_e^2 = \int_{-\infty}^{\infty} [x - Q(x)]^2 f_X(x) \, dx. \tag{15.19}$$

If we break the interval of integration into the separate intervals $\mathcal{R}_i$ and we note that $Q(x) = y_i$ when x is in $\mathcal{R}_i$, we can write (15.19) in the form

$$\sigma_e^2 = \sum_{i=1}^{L} \int_{x_{i-1}}^{x_i} (x - y_i)^2 f_X(x) \, dx. \tag{15.20}$$

15.2 Statistical analysis of quantization error

If the function $f_X(x)$ is relatively smooth and L is large, each *quantization step*

$$\Delta_i = x_i - x_{i-1} \tag{15.21}$$

can be made quite small to ensure that $f_X(x) \approx f_X(y_i)$ when x is in $\mathcal{R}_i$. Furthermore, if there is no overload we can assume that $f_X(x) \approx 0$ for x in the overload intervals. Under these conditions, we obtain

$$\sigma_e^2 = \frac{1}{12} \sum_{i=2}^{L-1} f_X(y_i) \Delta_i^3. \tag{15.22}$$

If we consider a uniform quantizer with constant step size $\Delta = \Delta_i$ and we recall that $\sum f_X(y_i)\Delta \approx \int f_X(y)dy = 1$, the variance expression (15.22) is simplified to

$$\sigma_e^2 = \frac{\Delta^2}{12}. \tag{15.23}$$

Thus, the power of the (granular) quantization noise of a uniform quantizer with step size Δ grows as the square of the step size. This result may be obtained directly if we assume that there is no overload noise and that the granular noise is uniformly distributed over the interval $-\Delta/2 \le e < \Delta/2$ (see Problem 24).

From the preceding analysis we conclude that the quantization process can be modeled as the addition of a random noise component $e[n] = Q(x[n]) - x[n]$ to the input sample $x[n]$ as shown in Figure 15.6. Many of the original results and insights into the nature of quantization error were presented in a classical paper by Bennet (1948), who showed that the quantization error can be modeled as a stationary white noise process with mean zero and variance given by (15.23) if:

1. The quantizer does not overload, that is, we have only granular noise.
2. The quantizer has a large number of quantization levels or equivalently a large number of bits.
3. The step size or resolution of the quantizer, that is, the distance between successive quantization levels, is very small.
4. The probability density function of the input signal has smooth shape.
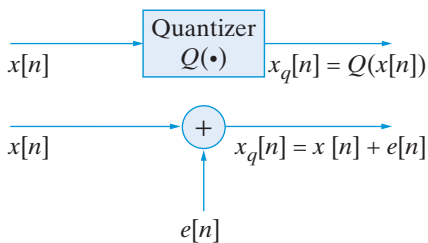
Figure 15.6 Additive quantization noise model.

Under these conditions the quantization error sequence $e[n]$ is a wide-sense stationary process with ACRS and PSD function given by

$$r_e[\ell] = \frac{\Delta^2}{12}\delta[\ell] \xleftrightarrow{\text{DTFT}} S_e(e^{j\omega}) = \frac{\Delta^2}{12}. \quad |\omega| \leq \pi \tag{15.24}$$

Furthermore, we assume that the sequences $e[n]$ and $x[n]$ are uncorrelated, that is,

$$\mathrm{E}(x[n]e[n-k]) = 0. \quad -\infty < k < \infty \tag{15.25}$$

These properties are not completely accurate because the quantization error is a deterministic function of the input signal. However, the additive quantization noise model shown in Figure 15.6 and described by (15.24) and (15.25) provides useful performance predictions for many practical systems. Properties (15.24) and (15.25) provide a good model for quantization noise if successive input samples are only moderately correlated, the number of quantization levels is large, and the quantization levels are in the middle of quantization intervals. These properties are satisfied by most practical signals like speech and music; however, we can easily find signals, for example, a step function, that violate these assumptions. Rigorous discussions of quantization noise models are given by Gersho (1978), Sripad and Snyder (1977), Gray (1990), and Marco and Neuhoff (2005). The properties of quantization noise are illustrated in the following example.

Example 15.1   Probability distribution of quantization noise

To investigate the statistical properties of quantization noise we consider the quantization of sinusoidal signals using a $(B + 1)$ bit rounding quantizer with $X_m = 1$. Since periodic sinusoids will generate periodic quantization error sequences, we avoid periodicity by choosing sinusoids with nonrational frequencies. For example, Figure 15.7(a) shows the unquantized sequence $x[n] = 0.99\sin(2\pi f_0 n)$ with $f_0 = 1/(2\pi 11)$; the amplitude is chosen slightly less than 1 to avoid overload. Figures 15.7(b) and (c) show the quantized signal $x_q[n] = Q\{x[n]\}$ and the quantization error $e[n] = x_q[n] - x[n]$ obtained with a 3 bit quantizer. We note that there are $L = 2^3 = 8$ quantization levels and the error signal is clearly correlated with the unquantized signal. If we quantize $x[n]$ using 8 bits, the quantization error shown in Figure 15.7(d), fluctuates randomly and appears to be uncorrelated with the unquantized signal.

The theoretical model of quantization noise assumes that its amplitude values follow the uniform probability density function $f_E(e)$ shown in Figure 15.8(a). To check whether the quantization error is uniformly distributed we quantize 50 000 samples of the sequence

$$x[n] = 0.99\{\sin(n/11) + \sin(n/31) + \cos(n/67)\}/3 \tag{15.26}$$

using an 8 bit quantizer. Then, we compute the histogram of the quantization error, which is shown in Figure 15.8(b). We note that there is a reasonable agreement between the
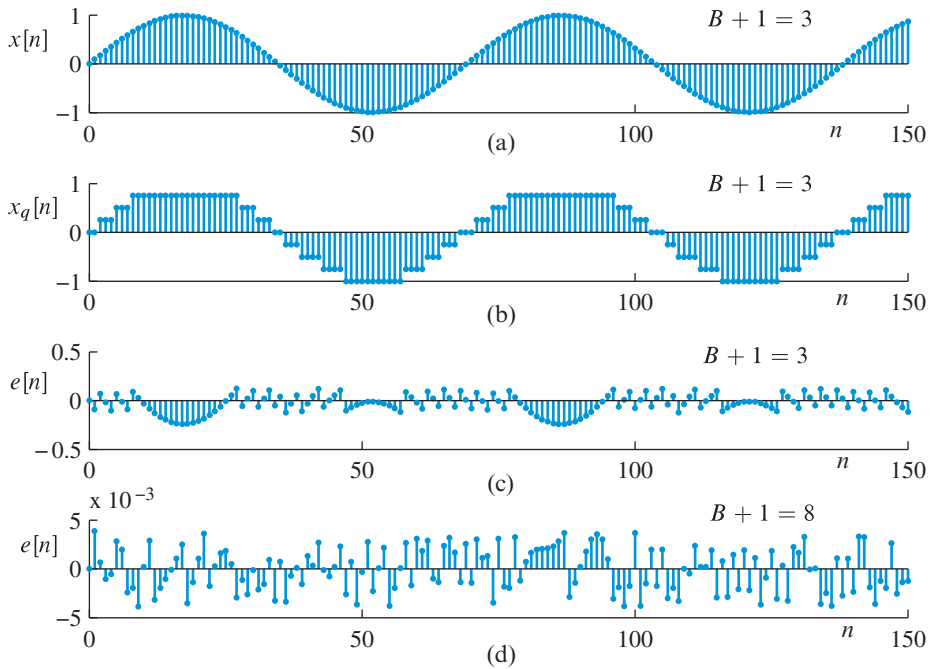
Figure 15.7 Quantization process for a sinusoidal signal: (a) unquantized signal $x[n]$, (b) quantized signal $x_q[n]$ (3 bits), (c) quantization error $e[n]$ (3 bits), and (d) quantization error $e[n]$ (8 bits).
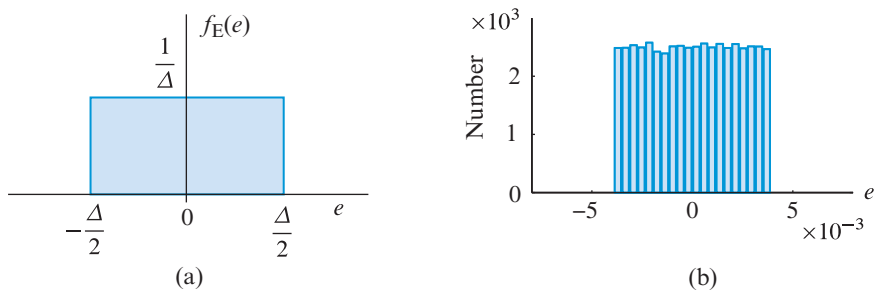
Figure 15.8 (a) Probability density function of quantization error for a rounding quantizer. (b) Histogram of quantization noise of $x[n]$ in (15.26) for $B + 1 = 8$ bits.

theoretical model and the experimental results; this agreement improves as we increase the number of bits. Additional investigations with different types of signal and different numbers of bits are provided in the problems. ∎

A uniform linear quantizer is specified by the number of levels $L = 2^{B+1}$, and either the step size Δ or the overload level X_m, where $X_m = x_L = -x_0$. To avoid significant overload distortion, the overload level is chosen to be a suitable multiple of the loading

factor $\eta = (X_m/\sigma_x)$. The step size of a uniform quantizer is

$$\Delta = \frac{2X_m}{2^{B+1}} = \frac{X_m}{2^B}. \tag{15.27}$$

The standard performance metric, the signal-to-quantization noise ratio (SQNR), is given by

$$SQNR = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right) = 10\log_{10}\left(\frac{12 \cdot 2^{2B}\sigma_x^2}{X_m^2}\right)$$

$$= 6.02(B+1) + 4.771 - 20\log_{10}\left(\frac{X_m}{\sigma_x}\right), \tag{15.28}$$

or more compactly, using $\eta = X_m/\sigma_x$, by

$$SQNR = 6.02B + 10.8 - 20\log_{10}(\eta). \tag{15.29}$$

We note that the loading factor η modifies the constant term in (15.29) but does not alter the rate of increase of SQNR with the number of bits, which is 6 dB per added bit. The SQNR depends both on the number of bits, $(B+1)$, and the loading factor $\eta = X_m/\sigma_x$. If we assume that the amplitude of the input signal follows a Gaussian distribution, choosing $\eta = 4$ ensures that only 0.064 percent of the samples will result in overload noise. Substitution into (15.29) yields

$$SQNR \approx 6B - 1.25 \text{ dB}. \tag{15.30}$$

An almost identical formula for sinusoidal signals was derived in Section 6.5 using a different approach. Due to hardware limitations the SQNR achieved by an A/D converter is usually less than the one predicted by (15.30). The *effective number of bits (ENOB)* for a practical A/D converter is determined by quantizing a sinusoidal signal, measuring the actual SQNR, and solving (15.29) for the ENOB. More information about the design and performance evaluation of A/D and D/A converters can by found in Kester (2005).

Example 15.2   Estimation of quantization noise PSD and SQNR

To confirm the model for the PSD of quantization error and the expression for the SQNR we quantize the signal in (15.26) using a uniform quantizer with different numbers of bits. Then, we compute the PSD of quantization error using Welch's method of averaged periodograms (see Tutorial Problem 3 for details). The resulting spectra are plotted in Figure 15.9 for $B + 1 = 2, 4, \ldots, 16$ bits. We note that when the number of bits is greater than 4, the PSD is quite flat over the entire frequency range, as postulated by the model. Furthermore, we note that the model predicts reasonably well the SQNR. Indeed, the PSD level is close to the value predicted by (15.30) and consecutive curves are offset from one
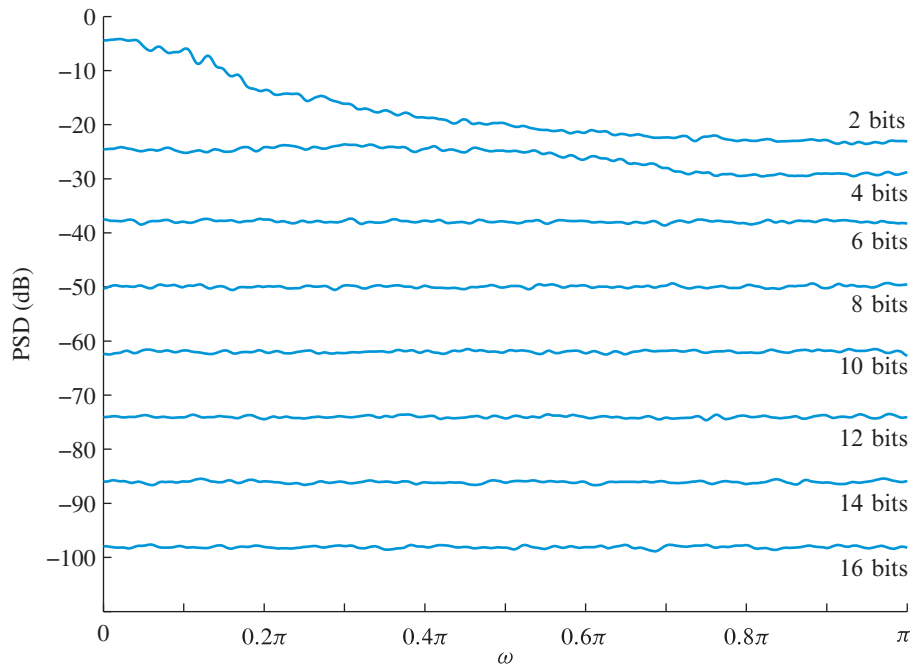
Figure 15.9 Power spectrum densities of quantization noise for different numbers of bits. The curves confirm that quantization noise has a white spectrum and that the SQNR increases 6 dB for each extra bit.

another by 12 dB because the number of bits changes by 2 from one curve to another. Similar results hold for other types of signal. ∎

Matching the dynamic range of the input signal to the full-scale range of the A/D converter, that is, choosing the proper loading factor, is necessary to achieve the performance predicted by (15.30). This is illustrated in the following example.

Example 15.3   Effect of overload quantization noise

In the derivation of the fundamental relations (15.24) and (15.25) we assumed that the overload distortion was negligible. For this assumption to hold, the signal amplitude should "match" the full-scale $2X_\mathrm{m}$ of the converter. To explain this concept we assume that the input signal follows a normal distribution $f_X(x) = N(0, \sigma_x^2)$ with mean zero and variance σ_x^2. Due to the even symmetry of the normal distribution, the granular noise and the overload distortion are given by

$$\sigma_\mathrm{g}^2 = 2 \int_0^{X_\mathrm{m}} e^2(x) f_X(x) \mathrm{d}x, \tag{15.31a}$$

$$\sigma_\mathrm{o}^2 = 2 \int_{X_\mathrm{m}}^{\infty} e^2(x) f_X(x) \mathrm{d}x, \tag{15.31b}$$

where $\eta = X_m/\sigma_x$ is the loading factor. The granular noise integral can be evaluated as follows

$$\sigma_g^2 = 2 \sum_k f(x_k) \int_{-\Delta/2}^{\Delta/2} x^2 dx$$

$$= 2 \frac{\Delta^2}{12} \sum_k f(x_k) \Delta \approx \frac{\Delta^2}{12} 2 \int_0^{\eta\sigma_x} f(x) dx. \qquad (15.32)$$

Since for $|x| > X_m$, the error is $e = x - X_m$, the overload distortion is given by

$$\sigma_o^2 = 2 \int_{\eta\sigma_x}^{\infty} (x - \eta\sigma_x)^2 f(x) dx. \qquad (15.33)$$

The integrals in (15.32) and (15.33) can be expressed in terms of the function

$$\Phi(\eta) = \frac{1}{\sqrt{2\pi}} \int_\eta^{\infty} e^{-u^2/2} du, \qquad (15.34)$$

which can be evaluated using MATLAB function `erf`. Since $\Delta = \eta\sigma_x/2^B$, we can show that (see Gray and Zeoli (1971) and Tutorial Problem 7)

$$\sigma_g^2 = 2\sigma_x^2 \frac{\eta^2 2^{-2B}}{12} [1/2 - \Phi(\eta)], \qquad (15.35a)$$

$$\sigma_o^2 = 2\sigma_x^2 \left[(\eta^2 + 1)\Phi(\eta) - \eta \frac{1}{\sqrt{2\pi}} e^{-\eta^2/2} \right]. \qquad (15.35b)$$

Figure 15.10 shows the SQNR $= \sigma_x^2/(\sigma_g^2 + \sigma_o^2)$ as a function of the loading factor $\eta = X_m/\sigma_x$ for different numbers of bits. Careful inspection of the curves in Figure 15.10 leads to several conclusions. First, we note that for each number of bits there is a maximum SQNR attained when the signal amplitude distribution matches the full range of the converter. For this optimum loading factor, η_{opt}, the signal uses effectively all quantization levels. When $\eta > \eta_{opt}$ the quantization levels are underutilized, which leads to increased granular noise. In this case we observe a linear decrease of SQNR as the loading factor increases logarithmically; the straight lines are offset from one another by 12 dB because the number of bits changes by 2 between consecutive curves. When $\eta < \eta_{opt}$ the SQNR drops rapidly because the quantizer operates in the overload region, which causes severe clipping of the input signal. In practice, matching the input signal to the full range of the A/D converter is accomplished using an automatic gain control system; see Kester (2005). ∎

15.2.1   Input A/D quantization noise through discrete-time systems

One source of error in the output of a discrete-time system is the propagation of the input signal's quantization noise through the system to the output. This error is in addition to
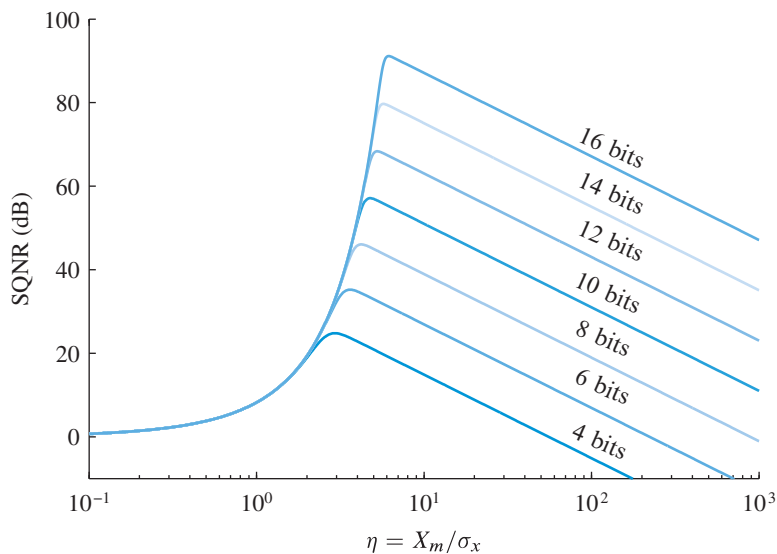
Figure 15.10 Signal-to-quantization noise ratio (SQNR) as a function of loading factor X_m/σ_x for several values of the number of bits B.
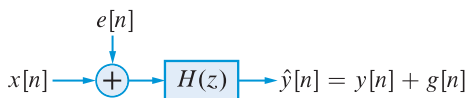
Figure 15.11 Model for analysis of input quantization noise through discrete-time system

the errors caused by the filter coefficient quantization (studied in Section 15.4) and the arithmetic quantization errors within the filter structure (studied in Section 15.5). Using the additive quantization noise model in Figure 15.6, the statistical model given by (15.24) and (15.25), and assuming infinite-precision arithmetic in the filter we can obtain fairly accurate bounds on the output error. We will also assume that the input signal is properly scaled to avoid saturation and overloading and that the quantization process uses the rounding operation.

Consider the system model given in Figure 15.11, which is excited by the quantized input $x_q[n] = x[n] + e[n]$ with the corresponding output $\hat{y}[n] \triangleq y[n] + g[n]$ where $y[n]$ is the output due to $x[n]$ and $g[n]$ is due to $e[n]$. From the model (15.24) $e[n]$ is a zero-mean random process with variance $\sigma_e^2 = \Delta^2/12$ where $\Delta = 2^{-B}$. Hence the sequence $g[n]$ is also a random process and, from (13.96), also has zero mean. The variance of $g[n]$, from (13.116), is given by

$$\sigma_g^2 = \frac{\sigma_e^2}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega, \tag{15.36}$$

since $S_{ee}(\omega) = \sigma_e^2$. The normalized output variance or the *variance-gain* from the input to the output is given by the ratio

$$\text{VG} \triangleq \frac{\sigma_g^2}{\sigma_e^2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega = \sum_{n=-\infty}^{\infty} \left| h[n] \right|^2, \tag{15.37}$$

where we have used Parseval's relation (4.94) in the last equality. For a real and stable filter we can further express VG as (see Tutorial Problem 9)

$$\text{VG} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega = \frac{1}{j2\pi} \oint_{\text{UC}} H(z)H(z^{-1})z^{-1}dz$$

$$= \mathcal{Z}^{-1}\left[H(z)H(z^{-1}) \right]\Big|_{n=0}, \tag{15.38}$$

which can be computed using residues of $H(z)H(z^{-1})$.

If the discrete-time system is an FIR filter with impulse response $h[n]$, $0 \le n \le M$, then the variance gain from (15.37) is given by

$$\text{VG} = \sum_{n=0}^{M} \left| h[n] \right|^2. \tag{15.39}$$

If the discrete-time system is a causal and stable IIR filter with system function

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}, \tag{15.40}$$

and the impulse response $h[n]$, then assuming $M = N$ and simple poles, we can express $H(z)$ as

$$H(z) = R_0 + \sum_{k=1}^{N} \frac{R_k}{z - p_k}, \tag{15.41}$$

where R_0 is a constant and R_k is the residue at the p_k pole. Then the variance-gain is given by (see Tutorial Problem 9)

$$\text{VG} = R_0^2 + \sum_{k=1}^{N} \sum_{k=1}^{N} \frac{R_k R_\ell^*}{1 - p_k p_\ell^*}, \tag{15.42}$$

while the approximate formula is given by

$$\text{VG} \simeq \sum_{k=0}^{K} \left| h[n] \right|^2. \quad K \gg 1 \tag{15.43}$$

Example 15.4   A/D quantization noise through IIR filter

Consider a first-order IIR filter given by

$$H(z) = \frac{1}{1 - 0.9z^{-1}} \quad \text{or} \quad h[n] = (0.9)^n u[n].$$

Then the variance-gain, from (15.42), is given by

$$\text{VG} = \frac{1}{1 - 0.9^2} = 5.26$$

If the pole is moved to 0.99 then the variance-gain would be more than 50. Therefore, a proper scaling is necessary to make VG close to unity to avoid making the output signal excessively noisy due to input signal quantization. ■

15.3   Oversampling A/D and D/A conversion

Oversampling, that is, sampling an analog signal at a sampling rate deliberately far above its Nyquist rate makes possible (a) the use of simpler antialiasing filters, and (b) the design of simpler and more accurate quantizers. The basic idea is to obtain data with increased resolution by averaging lower resolution oversampled data. For example, consider a one-bit sequence $x[n]$, where each sample takes a value of 0 or 1, obtained by oversampling an analog signal 128 times. The averaged sequence $y[n] = x[n] + x[n-1] + \cdots + x[n-127]$ has values in the range from 0 to 127. Decimation by a factor of 128 yields a Nyquist rate sampled sequence with a seven bit resolution. The reverse operations can be used to simplify D/A conversion.

15.3.1   Oversampled A/D conversion with direct quantization

Figure 15.12 shows a general system for A/D conversion based on oversampling the analog input signal and quantizing one sample at a time using a uniform quantizer. For analysis
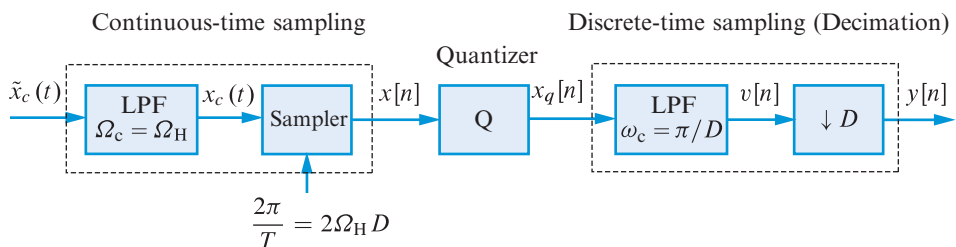
Figure 15.12 System for oversampling A/D conversion using a uniform quantizer.

purposes, we assume that the analog signal $\tilde{x}_c(t)$ is a realization of a zero-mean wide-sense stationary process. To avoid aliasing, we use an antialiasing filter $H_{aa}(j\Omega)$ with cutoff frequency $\Omega_c = \Omega_H$. Thus, the input $x_c(t)$ to the A/D is a zero-mean wide-sense stationary process with bandwidth Ω_H. Since the bandwidth of $x_c(t)$ is Ω_H, the Nyquist rate is $2\Omega_H$. If we use a sampling rate $\Omega_s = 2\pi/T \geq 2\Omega_H$, the *oversampling ratio (OSR)* is defined by

$$\text{OSR} \triangleq \frac{\Omega_s}{2\Omega_H} = \frac{2\pi/T}{2\Omega_H} \triangleq D, \tag{15.44}$$

where D is usually chosen to be a power of 2 for practical convenience.

Antialiasing filter design in oversampling To see how oversampling relaxes the requirements for the antialiasing filter, we assume that there is no quantizer in Figure 15.12. In practice the signal of interest is corrupted by broadband noise and other interferences as illustrated in Figure 15.13(a). The antialiasing filter $H_{aa}(j\Omega)$ is designed to have a
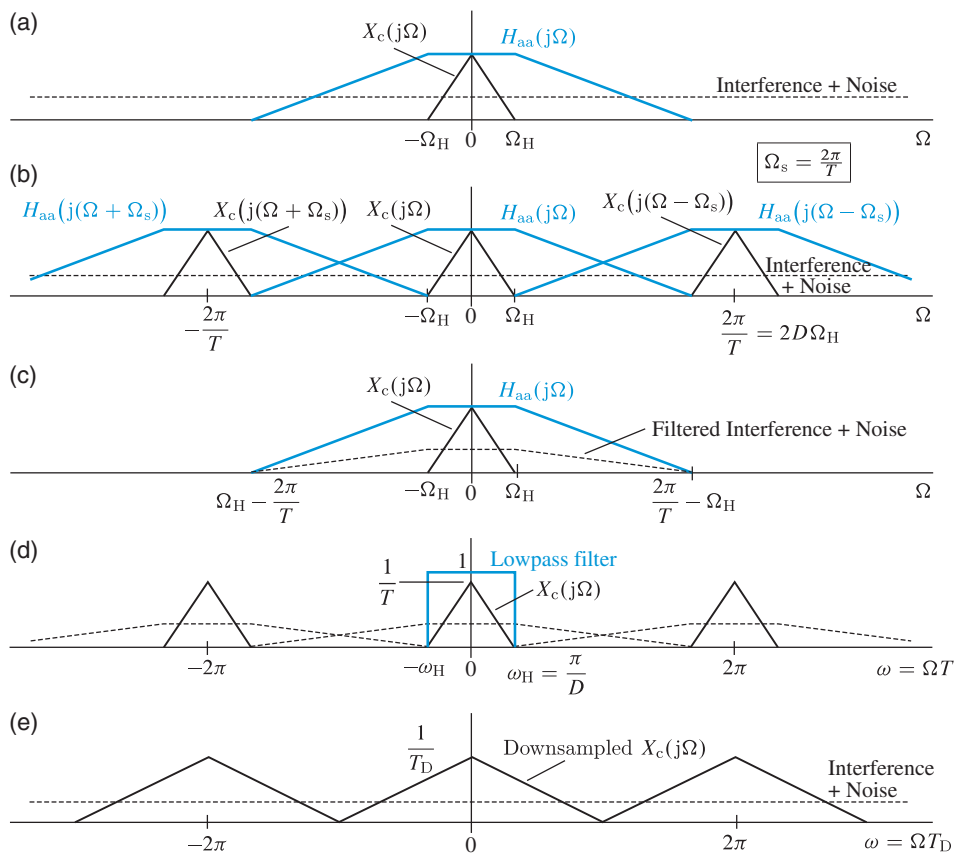
Figure 15.13 Use of oversampling and decimation to simplify implementation of antialiasing filters.

passband that covers the signal spectrum. To determine the transition band we carefully inspect Figure 15.13(b), which shows the images of $X_c(j\Omega)$ and $H_{aa}(j\Omega)$ at multiples of the sampling frequency $2\pi/T = 2D\Omega_H$. We note that the maximum stopband frequency of the antialiasing filter required to prevent aliasing by the noise is equal to $2\pi/T - \Omega_H = (2D-1)\Omega_H$; therefore, the transition band is given by

$$\Omega_{\text{stop}} - \Omega_{\text{pass}} = (2D-1)\Omega_H - \Omega_H = 2(D-1)\Omega_H. \tag{15.45}$$

This choice ensures that the aliased noise components do not affect the useful signal band. The magnitude response of this filter and the spectra of output analog signals are shown in Figure 15.13(c). This unwanted noise is subsequently filtered out with a sharp-cutoff low-pass digital filter with unity gain and cutoff frequency $\omega_c = \pi/D$. The filtered sequence is downsampled by a factor D to obtain a sequence sampled at the Nyquist rate. The spectra of the filtered and downsampled sequences are shown in Figures 15.13(d) and (e). The importance of this approach is that we can replace a complex and expensive analog antialiasing filter with a much simpler analog filter and a digital decimator. We illustrate these ideas with a simple example.

Example 15.5   Antialiasing filter design

In humans the audible range of audio (music) signal frequencies is usually said to be from 20 Hz to 20 kHz, although different applications limits this range. For example, FM radio transmits signals only up to 15 kHz. Let us assume that the bandwidth of the audio signal is 20 kHz. We want to convert this signal to digital form for use in audio CD recording which operates at the sampling rate of 44.1 kHz. Clearly, the sampling principle is satisfied since $20 < 44.1/2$.

We need an antialiasing filter prior to sampling which should have passband up to 20 kHz and a phase response that is linear phase. From Chapter 11 we know that the Butterworth approximation provides almost linear-phase response. From Figure 15.13 the stopband of the filter should begin at $44.1 - 20 = 24.1$ kHz. Let us assume the ripple parameters as $A_p = 0.01$ and $A_s = 80$ dB which are quite stringent. Then the order of the antialiasing Butterworth filter is given by:

```
>> Fs = 44100; Fpass = 20000; Fstop = Fs-Fpass;
>> Ap = 0.01; As = 80;
>> N = buttord(2*pi*Fpass,2*pi*Fstop,Ap,As,'s')
N =
    66.
```

Thus the order of the required filter is 66 which is very high. If we now oversample the signal by a factor of 4, then the required order is given by:

```
>> Fs = 4*44100; Fpass = 20000; Fstop = Fs-Fpass;
>> N = buttord(2*pi*Fpass,2*pi*Fstop,Ap,As,'s')
N =
    6,
```

which is a considerable reduction in order that leads to a simpler analog filter design. An oversampling by a factor of 8 gives a filter order of 5, which is a slight improvement but double the number of samples. ∎

Oversampling A/D converter resolution We next show that oversampling, besides making it possible to use a simpler antialiasing filter, can be used to increase the resolution of an A/D converter. To this end, suppose that the sequence $x[n] = x_c(nT)$ is quantized by a uniform quantizer with step size Δ. As we stated in Section 15.2, the signal at the output of the quantizer can be modeled as

$$x_q[n] = x[n] + e[n], \tag{15.46}$$

where the quantization noise $e[n]$ is a zero-mean wide-sense stationary white noise process with variance

$$\sigma_e^2 = \frac{\Delta^2}{12}. \tag{15.47}$$

Since $e[n]$ is a zero-mean process, the variance σ_e^2 and the average power $E\{e^2[n]\}$ are the same; thus, in the sequel, we use both terms interchangeably. The average power of quantization noise is uniformly spread over the entire Nyquist range $-\pi/T < \Omega < \pi/T$ or $-\pi < \omega < \pi$. Therefore, the power spectral density of the quantization noise process is given by

$$r_e[\ell] = \sigma_e^2 \delta[\ell] \xleftrightarrow{\text{DTFT}} S_e(e^{j\omega}) = \sigma_e^2. \quad |\omega| < \pi \tag{15.48}$$

We note that, as we have seen in Section 15.2, the power spectral density of quantization noise is given by (15.48) irrespective of the sampling rate used to obtain $x[n]$.

The output of the lowpass filter after the quantizer in Figure 15.12 can by expressed as

$$v[n] = h_{lp}[n] * x[n] + h_{lp}[n] * e[n] \triangleq x_o[n] + e_o[n]. \tag{15.49}$$

Since the model (15.46) assumes that $x[n]$ and $e[n]$ are uncorrelated, we have

$$E\{v^2[n]\} = E\{x_o^2[n]\} + E\{e_o^2[n]\}. \tag{15.50}$$

Figure 15.14(a) shows the power spectral densities of the signal and quantization noise at the output of the quantizer. We first note that $E\{x_o^2[n]\} = E\{x^2[n]\}$, because $x[n]$ is bandlimited to π/D. The power of output quantization noise is

$$E\{e_o^2[n]\} = \frac{1}{2\pi} \int_{-\pi/D}^{\pi/D} \sigma_e^2 d\omega = \frac{\sigma_e^2}{D}. \tag{15.51}$$

To obtain the SQNR at the output of the downsampler we use the following results. First, decimation by a factor D does not create any aliasing because the sequence $v[n]$
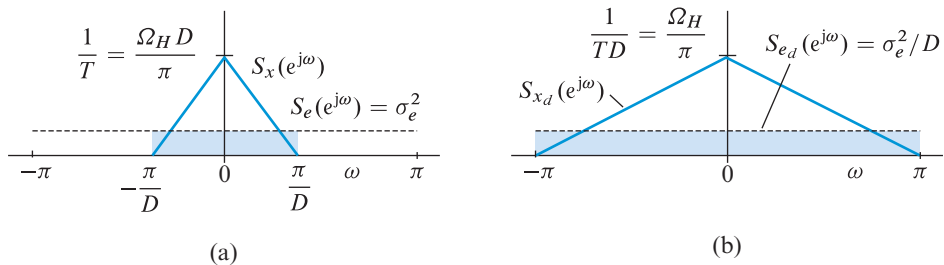
Figure 15.14 Power spectral densities of signal and quantization noise at (a) the input of the lowpass filter, and (b) the output of the downsampler.

is bandlimited to π/D. Second, decimation of a wide-sense stationary process does not change its average power. To this end, we first note that

$$y[n] = v[nD] = x_o[nD] + e_o[nD] \triangleq x_D[n] + e_D[n]. \qquad (15.52)$$

Since $x_o[n]$ and $e_o[n]$ are wide-sense stationary we have $E\{x_D^2[n]\} = E\{x_o^2[nD]\}$ and $E\{e_D^2[n]\} = E\{e_o^2[nD]\}$ for all n and D. As shown in Figure 15.14(b), downsampling stretches the input spectra by a factor of D and scales their amplitude by $1/D$, which leads to the same conclusion (see Problem 27). Therefore, the SQNR at the output of the downsampler is given by

$$\text{SQNR}_D \triangleq \frac{E\{x_D^2[n]\}}{E\{e_D^2[n]\}} = \frac{\sigma_x^2}{\sigma_e^2/D} = \text{SQNR}_{\text{NR}} \cdot D, \qquad (15.53)$$

where SQNR_{NR} is the SQNR for a Nyquist rate A/D converter. If $D = 2^r$, the SQNR improvement in dB is given by

$$\text{SQNR}_D = \text{SQNR}_{\text{NR}} + 3.01r \text{ (dB)}. \qquad (15.54)$$

Thus, each time we double the sampling rate we gain 3 dB in SQNR. Since one extra bit improves the performance of a linear quantizer by 6 dB, doubling the sampling rate is equivalent to adding 1/2 bit or an improvement of 1/2 bit/octave. For example, if we oversample by a factor of $D = 4$, we achieve an enhancement of one bit, that is, we need one fewer bit to achieve the same accuracy. However, to achieve an enhancement of 10 bits requires an oversampling factor of about one million ($D = 2^{20}$); however, this is not a desirable trade-off between sampling rate and resolution.

15.3.2 Oversampled A/D conversion with noise shaping

From the preceding development it should be apparent that (a) oversampling decreases the quantization noise power in the signal band by spreading a fixed quantization noise power over a bandwidth much larger than the signal band, and (b) the 1/2 bit/octave

Finite wordlength effects
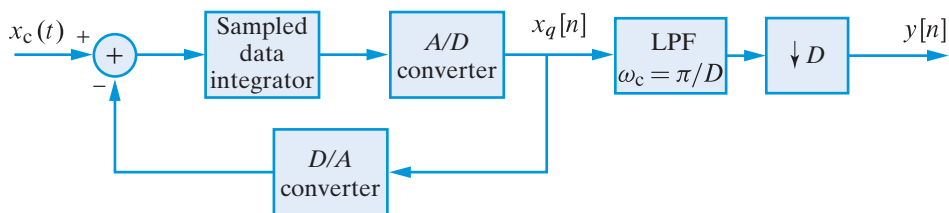
Figure 15.15 System for oversampled A/D conversion with noise shaping.
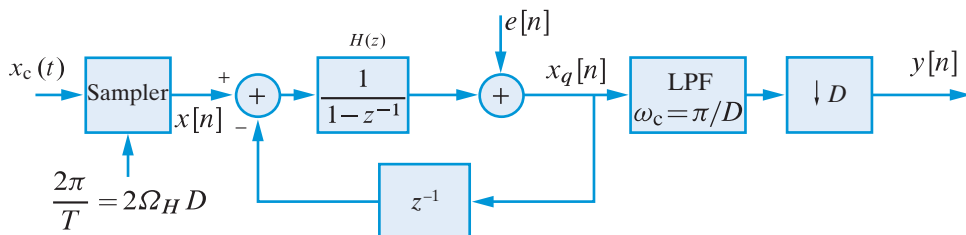
Figure 15.16 Discrete-time signal model of first-order sigma-delta A/D converter.

improvement stems from the flatness of the quantization noise spectrum introduced by the quantizer. Thus, it seems reasonable to expect additional performance improvement if we could push more quantization noise power outside the signal band. This is the basic principle underlying the operation of noise shaping quantizers discussed in the this section.

Figure 15.15 illustrates a general A/D conversion architecture with quantization noise shaping. The system consists of an analog integrator, which is typically implemented using sampled data switched capacitor technology, an internal A/D converter or quantizer, and a D/A converter used in a feedback loop.

The simplest A/D converter with noise shaping is obtained by using a first-order analog integrator. If we use an additive noise model for the quantizer and an accumulator for the analog integrator, we obtain the discrete-time model shown in Figure 15.16. The unit delay in the feedback loop, introduced by the D/A converter, is necessary to make the system realizable.

To determine the transfer function from $x[n]$ to $x_q[n]$, we assume for now that $x[n]$, $e[n]$, and $x_q[n]$ are deterministic sequences with z-transforms $X(z)$, $E(z)$, and $X_q(z)$, respectively. Then, the output $X_q(z)$ of the feedback loop is given by

$$X_q(z) = \frac{1}{1 - z^{-1}}[X(z) - z^{-1}X_q(z)] + E(z). \tag{15.55}$$

Solving this algebraic equation for $X_q(z)$ yields

$$X_q(z) = X(z) + (1 - z^{-1})E(z). \tag{15.56}$$

We note that the quantized sequence $x_q[n]$ is the sum of two components: the original signal $x[n]$ and a "filtered" quantization noise. In general, if we define the signal transfer

function $H_x(z)$ and noise transfer function $H_e(z)$ by

$$H_x(z) \triangleq Z\{h_x[n]\} = 1, \tag{15.57a}$$

$$H_e(z) \triangleq Z\{h_e[n]\} = 1 - z^{-1}, \tag{15.57b}$$

the output of the quantizer, for both deterministic and random signals, is given by

$$x_q[n] = h_x[n] * x[n] + h_e[n] * e[n], \tag{15.58a}$$

$$\triangleq x_f[n] + e_f[n]. \tag{15.58b}$$

Since the sequences $x[n]$ and $e[n]$ are wide-sense stationary, we have

$$S_{x_f}(e^{j\omega}) = S_x(e^{j\omega}), \tag{15.59a}$$

$$S_{e_f}(e^{j\omega}) = |H_e(e^{j\omega})|^2 S_e(e^{j\omega}) = \sigma_e^2 [2\sin(\omega/2)]^2. \tag{15.59b}$$

The power spectral densities in (15.59) are illustrated in Figure 15.17. We note that the power spectral density of quantization noise has been shaped so that more of the noise power is outside the signal band $|\omega| < \pi/D$. This "out-of-band" power is filtered out by the lowpass filter of the decimator. The quantization noise power at the output of the lowpass filter in Figure 15.16 is

$$E\{e_0^2[n]\} = \frac{4\sigma_e^2}{2\pi} \int_{-\pi/D}^{\pi/D} \sin^2(\omega/2)d\omega = \sigma_e^2 \left(\frac{\pi}{D} - \sin\frac{\pi}{D} \right) \approx \sigma_e^2 \frac{\pi^2}{3D^3}, \tag{15.60}$$

where we have used the approximation $\sin\theta \approx \theta - \theta^3/6$, $\theta \ll 1$, which holds for sufficiently large values of D. Since $x[n]$ is bandlimited to π/D, we have $E\{x_0^2[n]\} = E\{x^2[n]\}$. Therefore, following the arguments leading to (15.53), the SQNR at the output of the downsampler is given by

$$\text{SQNR}_D \triangleq \frac{E\{x_D^2[n]\}}{E\{e_D^2[n]\}} = \frac{\sigma_x^2}{\sigma_e^2} \frac{3D^3}{\pi^2} = \text{SQNR}_{NR} \cdot \frac{3D^3}{\pi^2}, \tag{15.61}$$
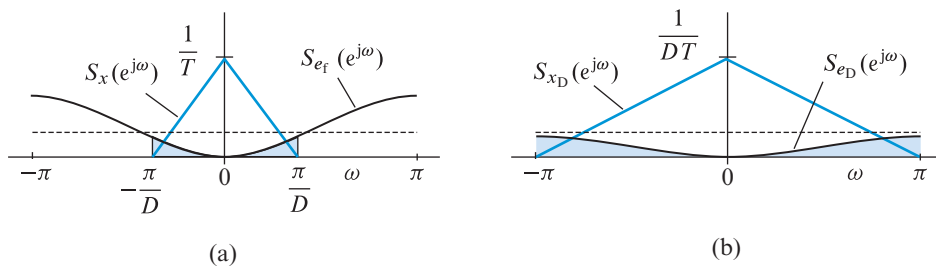
Figure 15.17 Power spectral densities of signal and quantization noise at (a) the input of the lowpass filter, and (b) the output of the downsampler following a first-order noise shaping quantizer.

where SQNR_{NR} is the SQNR for a Nyquist rate A/D converter. If $D = 2^r$, the SQNR improvement due to noise shaping is given by

$$\text{SQNR}_D = \text{SQNR}_{\text{NR}} - 5.17 + 9.03r \text{ (dB).} \tag{15.62}$$

Therefore, for every doubling of the oversampling ratio D, that is, for each increment in r, the SQNR improves by 9 dB, or equivalently, the resolution improves by 1.5 bits/octave. The loss of 5.17 dB, which corresponds to the $\pi^2/3$ multiplicative factor, results from the doubling of total quantization noise power by the noise shaping filter (see Tutorial Problem 10). If we revisit the case considered at the end of Section 15.3.1, we note that to achieve an enhancement of 10 bits with a noise shaping A/D converter requires a much lower oversampling factor $D \approx 2^{10/1.5} \approx 256$.

To achieve additional performance improvement, we need a more selective noise shaping transfer function $H_e(z)$ that pushes even more noise power outside the signal band. A straightforward extension of the first-order system in Figure 15.16 can be obtained by using multiple feedback loops as shown in Figure 15.18 for $p = 2$. The noise-shaping transfer function, which is given by $H_e(z) = (1 - z^{-1})^p$, has magnitude response

$$|H_e(e^{j\omega})| = [2\sin(\omega/2)]^{2p}. \tag{15.63}$$

The SQNR enhancement for the standard second-order system is given by

$$\text{SQNR}_D = \text{SQNR}_{\text{NR}} - 12.90 + 15.05r \text{ (dB),} \tag{15.64}$$

which shows that the resolution increases by 2.5 bits for each doubling of the oversampling ratio (see Problem 12 for details). Another architecture, known as multistage noise shaping (MASH), is obtained by cascading independent first-order stages; see Schreier and Temes (2005) for details. This cascading ensures the stability of the overall system provided that the individual stages are stable.

The fundamental ideas presented in this section can be extended in a variety of ways to create architectures that provide different trade-offs among bandwidth, resolution, circuit complexity, and feedback loop stability. Most practical systems use a one-bit quantizer, which is known as a *sigma-delta* $(\Delta\Sigma)$ *modulator*; see Hauser (1991). The nonlinearity of the quantization process, the feedback loops, and the imperfections of actual circuits, make
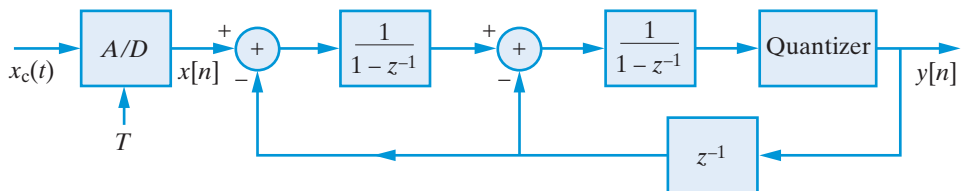
Figure 15.18 Oversampled A/D converter with second-order noise shaping.

the design and performance evaluation of practical noise-shaping converters extremely difficult; thus, we have to resort to sophisticated simulation packages.

15.3.3   Oversampled D/A conversion with noise shaping

The ideas discussed in the preceding section can be used to simplify the design and implementation of D/A converters. The basic system for oversampling D/A conversion is illustrated in Figure 15.19. The input sequence $x[n]$ has been quantized with high resolution (B bits), and is interpolated by a factor of D. The oversampled sequence $x_i[n]$ is then quantized using a one-bit quantizer. If the resulting quantization noise can be pushed outside the useful signal band, it can be subsequently removed by a simple analog filter. The required noise shaping can be done using the system shown in Figure 15.20.

To analyze the performance of this system we replace the quantizer by an additive white noise source; that is, we assume that $x_q[n] = v[n] + e[n]$, where $e[n]$ is the white noise process defined by (15.24) and (15.25). We can easily show that the signal and quantization noise transfer functions are given by (15.57a) and (15.57b), respectively. Thus,
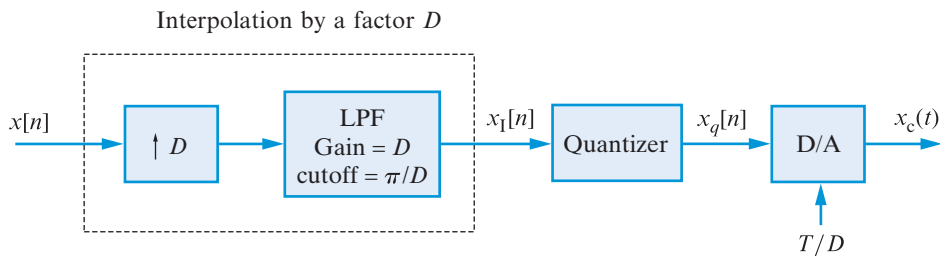
Figure 15.19 Basic system for oversampling D/A conversion.
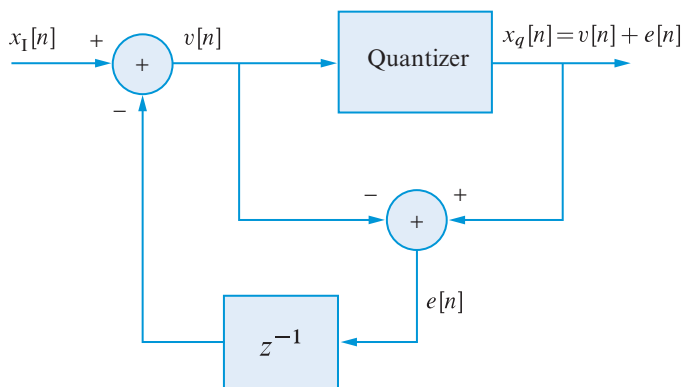
Figure 15.20 First-order noise shaping system for oversampled D/A conversion.
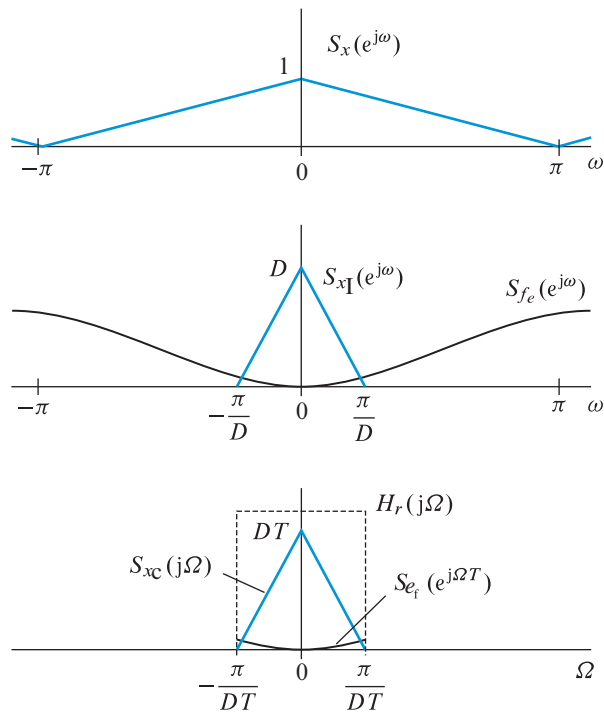
Figure 15.21 Signal and quantization noise power spectral densities in an oversampled D/A converter with first-order noise-shaping.

the quantization noise at the input of the D/A converter has a power spectral density given by

$$S_{e_f}(e^{j\omega}) = \sigma_e^2 [2\sin(\omega/2)]^2. \tag{15.65}$$

Figure 15.21 illustrates the operation of the oversampled D/A converter with noise-shaping in the frequency domain. Using multistage techniques for noise shaping, we can push more quantization noise power outside the useful signal band. This increases the SQNR and allows the use of inexpensive analog filters with larger transition bands (see Tutorial Problem 11).

15.4   Quantization of filter coefficients

Quantization of filter coefficients is a one time operation that changes filter coefficient values. This results in a filter with different characteristics than the original; a different frequency response, pole-zero locations, and/or stability. The new filter is still an LTI system (maybe unstable); simply it is different. If the new filter does not satisfy the design requirements, we have to obtain another that does by increasing the wordlength in the quantizer.

Quantization of IIR filter coefficients

Consider the system function of a direct form IIR filter, which is given by

$$H(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}, \qquad (15.66)$$

where the coefficients a_k and b_k are assumed to be unquantized. Unquantized coefficients are obtained using 32 or 64 bit floating point arithmetic, which for all practical purposes, provides an infinite precision representation. After quantization to fixed-point representation we obtain a new set of coefficients given by

$$\hat{b}_k = b_k + \Delta b_k, \quad k = 0, 1, \ldots, M \qquad (15.67a)$$

$$\hat{a}_k = a_k + \Delta a_k, \quad k = 1, 2, \ldots, N \qquad (15.67b)$$

where Δb_k and Δa_k are the changes or *perturbations* due to quantization. The system function after coefficient quantization becomes

$$\hat{H}(z) = \frac{\sum_{k=0}^{M} \hat{b}_k z^{-k}}{1 + \sum_{k=1}^{N} \hat{a}_k z^{-k}}. \qquad (15.68)$$

The new filter $\hat{H}(z)$ has poles and zeros moved to new locations and hence its frequency response is different from that of $H(z)$. If these changes are very sensitive to the perturbations then the resulting system may not have the required frequency response or it may even be stable. Using perturbation analysis it is possible to investigate analytically the bounds on the movement of poles and zeros, but the general analysis of the change in frequency response is difficult. Using the MATLAB functions developed in Section 15.1.2 it is possible to study changes in the frequency response and usability of the resulting filter $\hat{H}(z)$.

Pole and zero locations We begin with the simple case of a first-order IIR filter $H(z) = 1/(1 + az^{-1})$ with a pole at $z = -a$. Hence the quantization of a directly provides the new location of the pole at $-\hat{a}$. For a B bit fixed-point quantization $\hat{a}$ is uniformly spaced by 2^{-B} on the real axis according to $\hat{a} = \ell 2^{-B}$ for $-2^{B-1} \le \ell \le 2^{B-1} - 1$ using two's-complement format. To obtain satisfactory filter responses, we should provide sufficient bits or value B so that $\hat{a}$ is close to a.

Next, we consider the case of a second-order IIR filter with a complex pole-pair $re^{\pm j\theta}$, given by

$$H(z) = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}; \quad a_1 = -2r\cos(\theta), \ a_2 = r^2, \qquad (15.69)$$

and shown in Figure 15.22(a) using the direct form II structure. The fixed-point quantization of a_1 and a_2 would lead to a grid structure for pole locations that is given by the vertical lines (corresponding to the quantization of $2r\cos\theta$) and concentric circles (corresponding to r^2). This grid structure is shown in Figure 15.23(a) for $B = 3$ fractional bits and in (b) for $B = 5$ fractional bits. Note that to represent a_1 and a_2 with B fractional
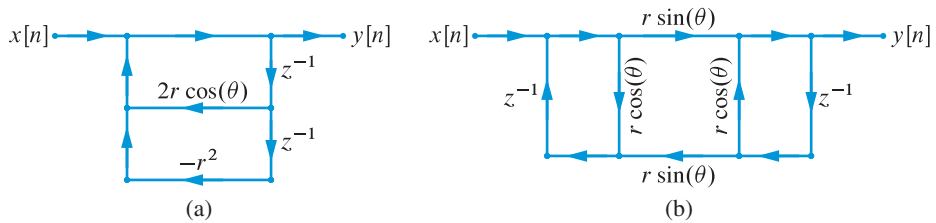
Figure 15.22 Second-order IIR filter implementations: (a) direct form II, and (b) coupled-form.
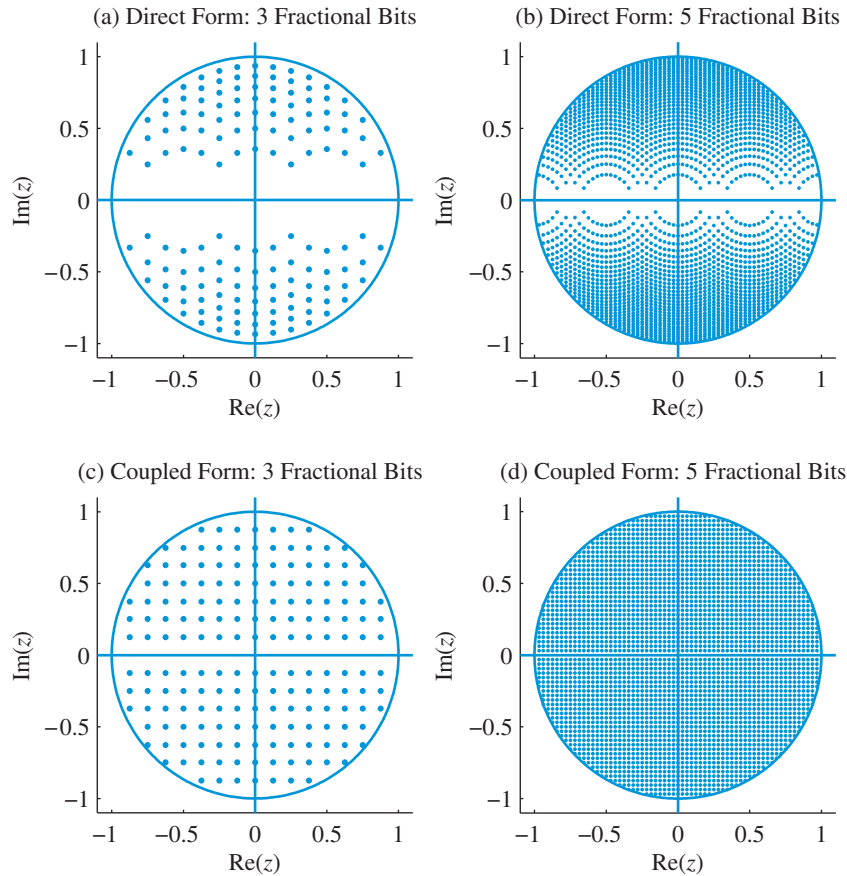
Figure 15.23 Pole distributions in second-order IIR filter implementations: (a) direct form II, $B = 3$, (b) direct form II, $B = 5$, (c) coupled-form, $B = 3$, and (d) coupled-form $B = 5$.

bits, we need additionally one sign bit and one integer bit since $-2 < a_1 < 2$ for a total of $B + 2$ bits. The grid structure is interesting but potentially troubling. The coefficients are sparsely distributed along the real axis but have a good distribution away from it. Thus poles located near real $z = \pm 1$ (lowpass or highpass filters) will have a greater movement and bigger impact on the frequency response.

Another second-order structure due to Gold and Rader (1969), called a coupled-form structure, is shown in Figure 15.22(b). Its poles with infinite precision are still given by $r\mathrm{e}^{\pm\mathrm{j}\theta}$ although its coefficients are either $r\cos(\theta)$ or $r\sin(\theta)$ (see Problem 28). Hence the two realizations have the same denominator. Quantization of these filter coefficients leads to a grid of evenly-spaced horizontal and vertical lines since $r\cos(\theta)$ and $r\sin(\theta)$ are real and imaginary parts of the pole locations. This grid structure is shown in Figure 15.23(c) for $B = 3$ bits and in (d) for $B = 5$ bits. Clearly, the coupled-form structure has less sensitivity near the real axis compared to the direct form, although it requires twice as many multipliers.

For higher-order IIR filters, it is rather tedious to determine the grid structure unless we examine the individual second-order sections. However, using sensitivity analysis it is possible to obtain a bound on the movement of poles as a function of the change Δa_k in (15.67b). It can be shown, see Proakis and Manolakis (2007) and Tutorial Problem 13, that a perturbation Δa_k to the denominator coefficient a_k produces a change in every pole location given by

$$\Delta p_i = -\sum_{k=1}^{N} \frac{p_i^{N-k}}{\prod_{j=1, j \neq i}^{N}(p_i - p_j)} \Delta a_k. \tag{15.70}$$

This relation is known as the *sensitivity* formula. It shows that filters with tightly clustered poles (that is, narrowband filters) are very sensitive to coefficient quantization because the denominator of (15.70) can become very small. The direct form implementations of these filters therefore suffer from increased sensitivity and can become unstable and/or unusable. To minimize sensitivity, we have to maximize $|p_i - p_j|$. Second-order sections with complex-conjugate poles that are usually far apart from each other can reduce sensitivity and improve performance. Therefore, for IIR filters implemented with fixed-point arithmetic, cascade or parallel forms are preferred over direct forms because they are composed of less sensitive first- and second-order sections. These observations are explored in Example 15.6.

A formula analogous to (15.70) can be obtained for the zeros of the system function and is developed in Problem 29. These movements of zeros are also sensitive to the tight clustering of zeros and can change frequency responses but have no effect on the stability. Once again, using a cascade form, the effect of coefficient change Δb_k in (15.67a) on the filter performance can be mitigated.

Example 15.6   Effect of coefficient quantization on pole movements
Consider the bandpass elliptic filter with specifications $\omega_{s_1} = 0.2\pi$, $\omega_{p_1} = 0.3\pi$, $\omega_{p_2} = 0.4\pi$, $\omega_{s_2} = 0.5\pi$, $A_p = 0.1$ dB, and $A_s = 60$ dB designed using the following MATLAB script:

```
>> omegap = [0.3,0.4]; omegas = [0.2,0.5]; Ap = 0.1; As = 60;
>> [N,omegac] = ellipord(omegap,omegas,Ap,As);
>> [b,a] = ellip(N,Ap,As,omegac);
```
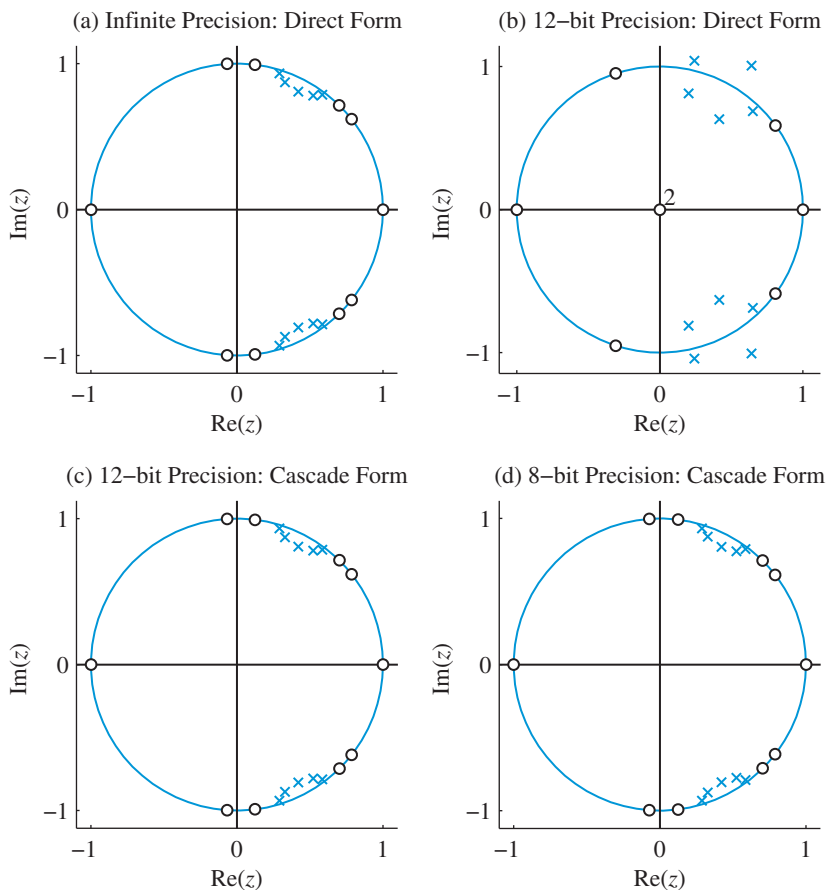
Figure 15.24 Effects of coefficient quantization on pole-zero movement for the bandpass elliptic filter in Example 15.6.

This filter design is assumed to be of infinite precision. Figure 15.24(a) shows a zero-pole plot of the resulting filter. We quantize filter coefficients to 12 bits using

```
>> L1 = 12; [bahat,E1,B1] = d2beqR([b;a],L1);
bhat1 = bahat(1,:); ahat1 = bahat(2,:); B1
B1 =
     6
```

that assigns 6 bits to the fractional part. The resulting zero-pole plot is shown in Figure 15.24(b). The poles and zeros have dispersed to new locations from the original ones and some poles are outside the unit circle making the resulting filter unstable and unusable. Next, we convert the direct form to cascade form and quantize the resulting coefficients using 12 bits and the MATLAB script:

```
>> [sos,G] = tf2sos(b,a);
>> L2 = 12; [soshat,E2,B2] = d2beqR(sos,L2);
```

```
>> [bhat2,ahat2] = sos2tf(soshat,G); B2
B2 =
    10
```

Since cascade form coefficients have smaller values, they need only one integer bit, leaving 10 bits for the fractional part. The resulting zero-pole plot is shown in Figure 15.24(c) which shows no discernible movement of poles or zeros. Finally, we quantize coefficients using 8 bits so the fractional bits are 6 which is the same as those in Figure 15.24(b). The resulting zero-pole plot is shown in Figure 15.24(d), which again shows no discernible movement of poles or zeros. This example clearly demonstrates that cascade (and similarly parallel) forms are preferable to direct forms.   ∎

Frequency response The frequency response of the IIR filter with the quantized coefficients in (15.68) is given by

$$\hat{H}(e^{j\omega}) = \frac{\sum_{k=0}^{M} \hat{b}_k z^{-k}}{1 + \sum_{k=1}^{N} \hat{a}_k z^{-k}}.$$
(15.71)

Although it is possible to obtain bounds on the performance of the magnitude or phase response as a function of change in filter coefficients due to quantization, such analysis is very tedious. Therefore, one resorts to numerical analysis as illustrated in the following example.

Example 15.7   Effect of coefficient quantization on frequency response

Consider the bandpass filter designed in Example 15.6. Figure 15.25(a) shows its magnitude response which satisfies the given requirements in Example 15.6. The coefficients of the filter were quantized to 12 bits with 6 bits assigned to the fractional part. The resulting magnitude response is shown in Figure 15.25(b), which clearly shows distortion of the equiripple behavior and shows that the requirements are not satisfied. Next, the unquantized coefficients were converted to cascade form, then quantized to 12 bits with 10 bits assigned to the fractional part. The resulting magnitude response is shown in Figure 15.25(c), which does not exhibit any distortion from the unquantized response. Finally, the coefficients were quantized to 8 bit so that 6 bits were assigned to the fractional part and the resulting magnitude response is shown in Figure 15.25(d). This response again has no noticeable distortion and the specifications are satisfied.   ∎

15.4.2   Quantization of FIR filter coefficients

Since FIR filters are characterized either by zeros or the numerator polynomial coefficients, they are easier to analyse compared to IIR filters. Let the system function of an FIR filter be

$$H(z) = \sum_{n=0}^{M} h[n]z^{-n},$$
(15.72)
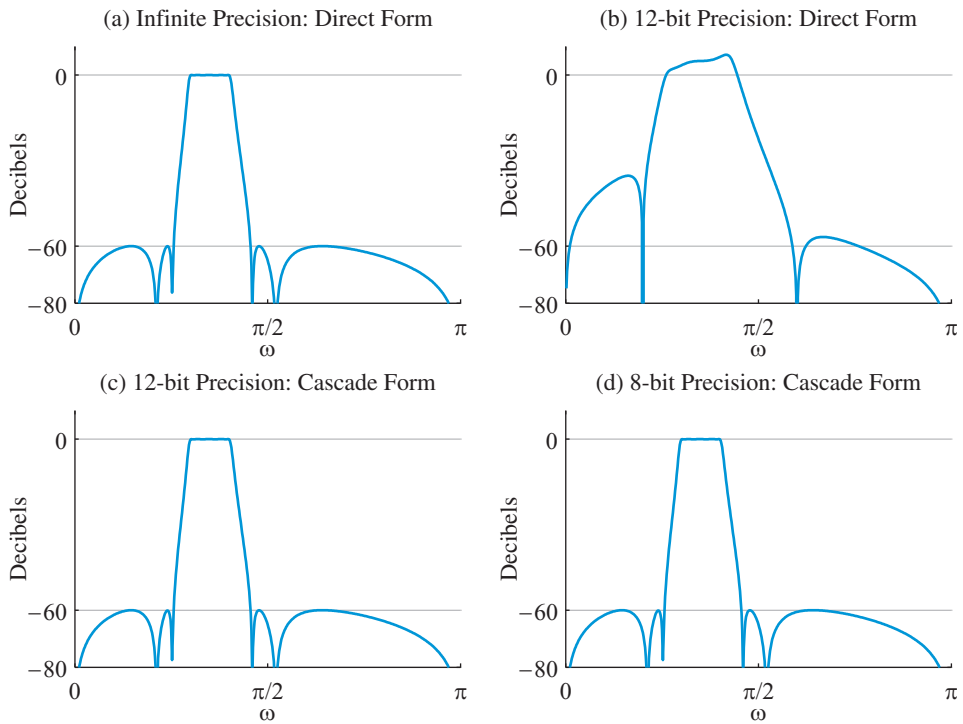
Figure 15.25 Effects of coefficient quantization on magnitude response for the bandpass elliptic filter in Example 15.7.

where $h[n]$ is the impulse response as well as the filter coefficients. Since (15.72) is linear in $h[n]$, the change $\Delta H(z)$ in $H(z)$ due to a change $\Delta h[n]$ in $h[n]$ is given by

$$\Delta H(z) = \sum_{n-0}^{M} \Delta h[n] z^{-n}. \tag{15.73}$$

Hence the magnitude of the change in the frequency response is given by

$$\left| \Delta H(e^{j\omega}) \right| = \left| \sum_{n-0}^{M} \Delta h[n] e^{-j\omega n} \right| \leq \sum_{n=0}^{M} |\Delta h[n]|. \tag{15.74}$$

If each coefficient is quantized to B fraction bits using the rounding operation, then $|\Delta h[n]| \leq \frac{1}{2} 2^{-B}$ (see Section 15.1.2) or

$$\left| \Delta H(e^{j\omega}) \right| \leq (M+1) 2^{B-1}, \tag{15.75}$$

which implies that the bound on the change in the frequency response is affected not only by the number of fractional bits B but also by the length of the filter. Therefore, if the filter

has a large order and a small B then the frequency response will be significantly distorted, which can destroy the equiripple behavior of a filter designed using the Parks–McClellan algorithm. One approach to reducing the size of the bound is to implement smaller length impulse response sections using a cascade form as illustrated in Example 15.8.

The bound in (15.75) is a conservative bound and assumes the worst-case scenario that all errors $|h[n]|$ achieve their maximum. Using the statistical model given by (15.24) and (15.25) and assuming that $|h[n]|$ are independent random variables, a more realistic bound has been derived in Chan and Rabiner (1973).

Example 15.8   Coefficient quantization effect on FIR filter

Consider again the specifications of the bandpass filter given in Example 15.6. We design an equiripple FIR filter using the following MATLAB script:

```
>> b = firpm(69,[0,0.2,0.3,0.4,0.5,1],[0,0,1,1,0,0]);
```

Figure 15.26(a) shows the magnitude response of a length 70 filter that satisfies the given requirements. The coefficients of the filter were quantized to 12 bits with 11 bits assigned
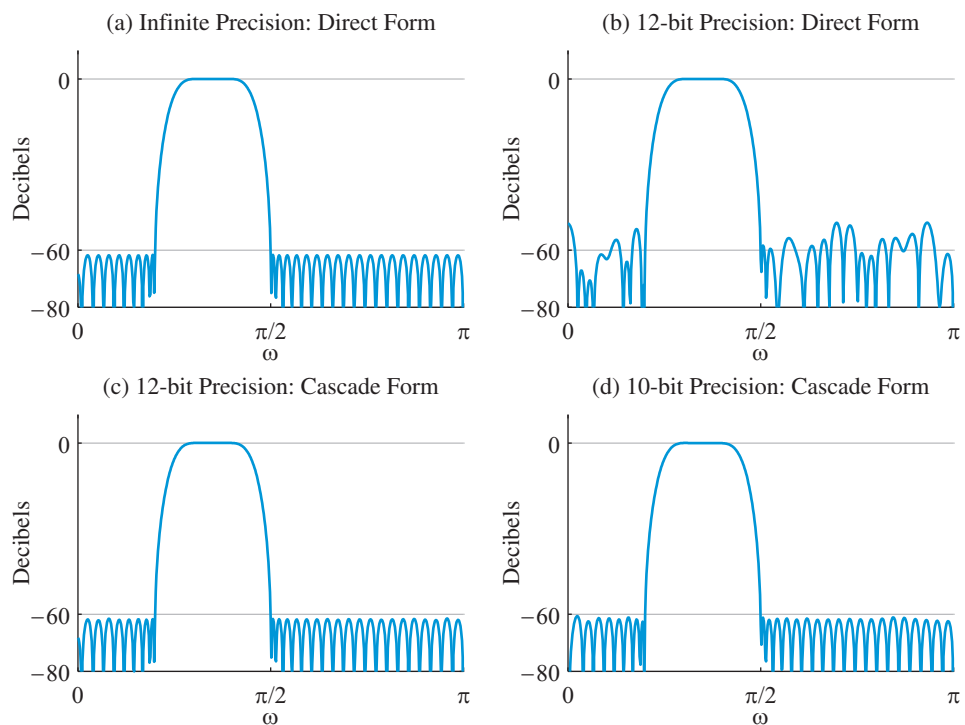
Figure 15.26 Effects of coefficient quantization on magnitude response for the bandpass equiripple FIR filter in Example 15.8.

to the fractional part. The resulting magnitude response is shown in Figure 15.26(b) which clearly shows distortion of the equiripple behavior in the stopband and shows that many peaks have less than 60 dB attenuation. Next, the unquantized coefficients were converted to cascade form, then quantized to 12 bits with 10 bits assigned to the fractional part. The resulting magnitude response is shown in Figure 15.26(c), which does not exhibit any distortion from the unquantized response even with 10 bits. Finally, the coefficients were quantized to 10 bit so that 8 bits were assigned to the fractional part and the resulting magnitude response is shown in Figure 15.25(d). This response again has some noticeable distortion but the specifications are satisfied.   ■

15.5   Effects of finite wordlength on digital filters

In many applications digital filters are implemented on personal computers or workstations built using very accurate and high speed floating-point arithmetic units. In such cases, we can assume that all computations are executed with practically "infinite" accuracy. However, in high-volume applications, where cost should be low, we must use finite wordlength fixed-point arithmetic.

The implementation of digital filters requires addition and multiplication operations. From Section 15.1.2 we recall that the addition of two $(B + 1)$ bit numbers may produce $(B + 2)$ bits, while their product always requires $(2B + 1)$ bits. If the sum is outside the range, addition increases the number of bits before the binary point, and we have an overflow. On the other hand, multiplication always increases the number of bits after the binary point. Removing the extra bits to fit intermediate results to the available $(B + 1)$ bit words makes a digital filter a nonlinear system. Nonlinear systems are extremely difficult or even impossible to analyze theoretically. Thus, the most effective approach to analyzing finite wordlength effects is to simulate a specific filter and evaluate its performance.

The fundamental difference between overflow and quantization errors, which determines their analysis and impact on the performance of a digital filter, is that overflow errors can be unbounded whereas quantization errors are always bounded in size by the least significant bit. There are three types of error caused by finite wordlength arithmetic operations inside a digital filter:

- Round-off noise at the output of a digital filter caused by quantization of intermediate multiplication results. Since round-off noise is unavoidable, the designer's goal is to choose the filter structure and wordlength which ensure acceptable signal-to-noise ratio at the output of the filter.
- Large errors caused by addition overflows; here, the goal of the designer is to prevent overflows by proper scaling of the quantities to be added.
- Zero-input limit cycles, which are sustained oscillations at the output of IIR digital filters even when there is no applied input, caused by either product quantization or addition overflow. The goal of the designer is to find filter structures that do not support limit cycle oscillations.

In this section we introduce a number of results which can be used to guide implementation of digital filters. However, we emphasize that simulation of a particular system is necessary to evaluate its actual performance. To simplify the subsequent analyses we treat each type of error separately.

15.5.1   Effects of round-off noise in direct-form FIR filters

In this section we consider the finite wordlength arithmetic operations (multiplication quantization and addition overflow) and their effects on the output of direct form FIR filters. Since rounding quantizer models are mostly employed, these effects are also known as round-off noise effects. To introduce the basic ideas, we start with the direct form FIR filter shown in Figure 15.27(a). The filter output, evaluated with infinite precision arithmetic, is given by

$$y[n] = \sum_{k=0}^{M} b_k x[n-k]. \tag{15.76}$$

Consider next a fixed-point implementation where each product, $b_k x[n-k]$, is quantized to $(B+1)$ bits with a rounding quantizer. Then, the output is given by

$$\hat{y}[n] = \sum_{k=0}^{M} Q\{b_k x[n-k]\}. \tag{15.77}$$

To analyze the effects of rounding errors, we replace the quantizer after each filter multiplier by an additive noise source, as shown in Figure 15.27(b), and we compute the noise

(a)

(b)
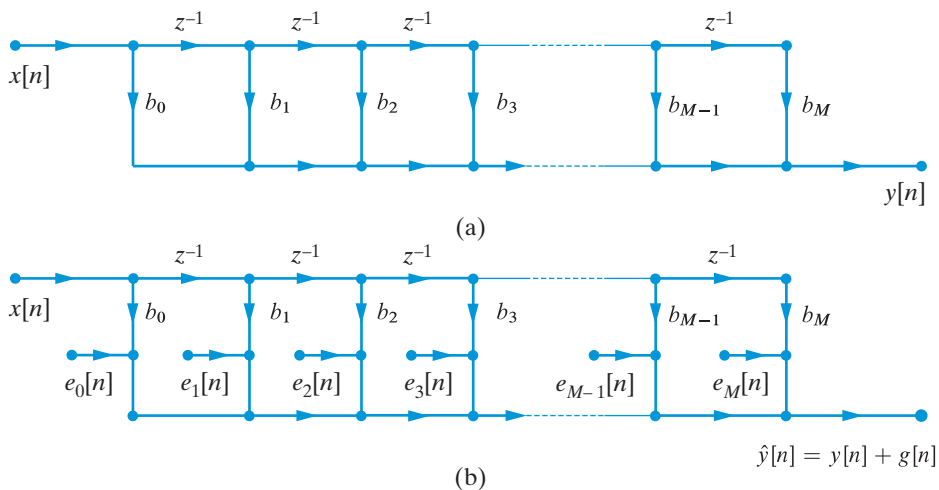
Figure 15.27 (a) Direct form structure for FIR filter infinite-precision implementation, and (b) linear noise model.

power at the output of the filter. The round-off noise sources, $e_k[n]$, satisfy the following properties:

1. Each round-off noise source $e_k[n]$ is a wide-sense stationary white noise process with mean and variance given by

$$\mu_e = 0, \quad \sigma_e^2 = \frac{\Delta^2}{12} = \frac{2^{-2B}}{12}. \tag{15.78}$$

2. Each round-off noise source is uniformly distributed in the quantization interval $-\Delta/2 < e_k[n] < \Delta/2$.
3. Each round-off noise source is uncorrelated with the input to the corresponding quantizer, all other round-off noise sources, and the input signal.

We concentrate on rounding because it is the most often used type of quantization; the results for truncation can be obtained by changing the mean value to $\mu_e = 1/2$.

Since $Q\{b_k x[n-k]\} = b_k x[n-k] + e_k[n]$, using (15.76) and (15.77), we obtain

$$\hat{y}[n] = y[n] + g[n], \tag{15.79}$$

where the total round-off noise at the filter output is given by

$$g[n] \triangleq \sum_{k=0}^{M} e_k[n]. \tag{15.80}$$

Since $g[n]$ is a linear combination of uncorrelated random variables, using (13.96) and (13.116) from Section 13.4, we have

$$E\{g[n]\} = 0, \quad \sigma_g^2 = (M+1)\frac{\Delta^2}{12} = \left(\frac{M+1}{3}\right)2^{-2(B+1)}. \tag{15.81}$$

We note that the round-off noise power at the filter output increases proportionately to the number of coefficients.

If a double-length, that is, a $2(B+1)$ bit, accumulator is available, we can improve performance by accumulating the sum of products with double accuracy and quantizing the final result. This approach uses a single quantizer as follows:

$$\hat{y}[n] = Q\left\{\sum_{k=0}^{M} b_k x[n-k]\right\}. \tag{15.82}$$

Since we use a single quantizer, the output round-off noise variance is given by

$$\sigma_g^2 = \frac{\Delta^2}{12} = \left(\frac{1}{3}\right)2^{-2(B+1)}, \tag{15.83}$$

which is significantly smaller than (15.81). Most DSP processors have accumulators with $2(B + 1)$ or more bits to avoid overflows and achieve more precision by rounding the final accumulated sum of products.

15.5.2   Scaling to avoid overflows in direct-form FIR filters

For a direct form FIR filter implemented by (15.76), the power of output round-off noise is fixed for a given number of bits. The simplest way to increase the SNR at the output is to increase the input signal level. However, if we make the level too high internal computations may lead to overflows. Thus, proper scaling of the input signal to maximize the SNR while we avoid overflow is very important in practical implementation of digital filters.

Since we assume the use of two's complement arithmetic, if the final result $y[n]$ is within the range, it gives the correct value even if there are overflows in the partial sums. Thus, a necessary and sufficient condition to avoid overflow is

$$|y[n]| = \left| \sum_{k=0}^{M} h[k]x[n-k] \right| \leq \sum_{k=0}^{M} |h[k]||x[n-k]| < 1. \tag{15.84}$$

If the input signal is bounded, that is, $|x[n]| \leq X_m$ for all n, we can always avoid overflow by finding a scaling factor S such that

$$S X_m \sum_{k=0}^{M} |h[k]| < 1. \tag{15.85}$$

Practical experience has shown that this condition is unreasonably conservative because it attenuates the input signal too much. In this case, precision is lost because the input signal uses only a small part of the dynamic range.

Another bound, which is more suitable for sinusoidal and narrowband signals (harmonic scaling), is based on the result

$$x[n] = X_m \cos(\omega n) \overset{\mathcal{H}}{\longmapsto} y[n] = X_m |H(e^{j\omega})| \cos[\omega n + \angle H(e^{j\omega})], \tag{15.86}$$

which suggests the following condition for preventing overflow:

$$S X_m < \frac{1}{\max_{\omega} |H(e^{j\omega})|}. \tag{15.87}$$

A third approach is to scale the input signal so that

$$E_y = \sum_{n=-\infty}^{\infty} |y[n]|^2 \leq S^2 \sum_{n=-\infty}^{\infty} |x[n]|^2 = S^2 E_x. \tag{15.88}$$

Using Parseval's theorem and Schwartz's inequality we obtain

$$E_y = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) X(e^{j\omega}) \right|^2 d\omega \le E_x \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega. \tag{15.89}$$

Combining (15.88) with (15.89) yields the following power based scaling factor:

$$S^2 < \frac{1}{\sum_{k=0}^{M} \left| h[n] \right|^2} = \frac{1}{\frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega}. \tag{15.90}$$

To compare the three scaling methods we recall the following inequalities

$$\left\{ \sum_{0}^{M} \left| h[n] \right|^2 \right\}^{1/2} \le \max_{\omega} |H(e^{j\omega})| \le \sum_{0}^{M} |h[n]|, \tag{15.91}$$

which show that (15.85) provides the most conservative scaling. Finally, we note that the scaling factor can be absorbed by the FIR filter coefficients, that is, we can replace each b_k by Sb_k in the filter structure.

15.5.3   Round-off noise and scaling in IIR filters

As we stated in Section 15.4.1, high-order IIR filters are implemented using parallel and cascade connection of second-order sections because they are the more robust to coefficient quantization. Since second-order sections are the building block for higher-order filters we discuss the effects of round-off noise and scaling on direct form II and transposed direct form II structures.

Normal direct form II structures – round-off noise The round-off noise analysis method introduced in Section 15.5.1 can be applied to any filter structure as long as we can determine the corresponding frequency responses between every noise source node and the output node of the filter. Consider the direct form II structure shown in Figure 15.28. As a consequence of the recursion, the products $-a_1 w[n-1]$ and $-a_2 w[n-2]$ increase by B bits at each iteration; therefore, quantization is unavoidable. The quantization of each product can be modeled by an additive round-off noise source after the multiplier (shown by blue arrows). The sources related to a_k and b_k coefficients can be replaced by the single sources $e_a[n] = e_4[n] + e_5[n]$ and $e_b[n] = e_1[n] + e_2[n] + e_3[n]$, respectively, as shown in Figure 15.28. Since the combined sources are uncorrelated, the combined noise sources have means $\mu_a = \mu_b = 0$ and variances given by

$$\sigma_a^2 = 2 \frac{\Delta^2}{12}, \quad \sigma_b^2 = 3 \frac{\Delta^2}{12}. \tag{15.92}$$

Since $e_a[n]$ appears at the input it is filtered by the entire system function $H(z)$. In Section 15.2.1 we showed that the output $g_a[n]$ of an LTI system $H(z)$ with a white noise input $e_a[n]$ has mean value and variance given by

$$\mu_{g_a} = \mu_{e_a} \sum_{n=0}^{\infty} h[n] = \mu_{e_a} H(e^{j0}) = 0, \tag{15.93}$$
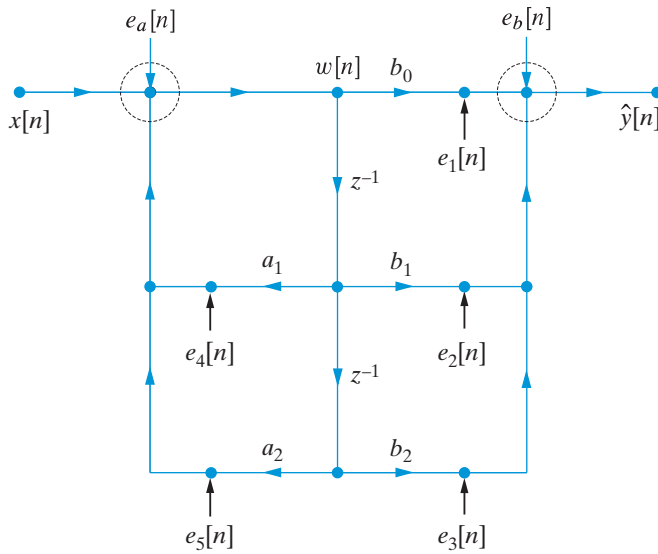
Figure 15.28 Direct form II structure showing individual and combined round-off noise sources and potential overflow nodes.

and

$$\sigma_{g_a}^2 = \sigma_{e_a}^2 \sum_{n=0}^{\infty} |h[n]|^2 = 2 \frac{\Delta^2}{12} \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega})|^2 d\omega. \tag{15.94}$$

The noise source $e_b[n]$ appears at the output; thus, the total round-off variance at the output of the filter is

$$\sigma_g^2 = 2 \frac{\Delta^2}{12} \sum_{n=0}^{\infty} h^2[n] + 3 \frac{\Delta^2}{12}. \tag{15.95}$$

The quantity $\sum_{n=0}^{\infty} h^2[n]$ can be easily evaluated in MATLAB with sufficient accuracy; thus, we can avoid the analytical formulas that require contour integration.

Transposed direct form II structures – round-off noise Consider next the transposed direct form II structure shown in Figure 15.29. The quantization of each product can be modeled by an additive round-off noise source after the multiplier (shown by blue arrows).
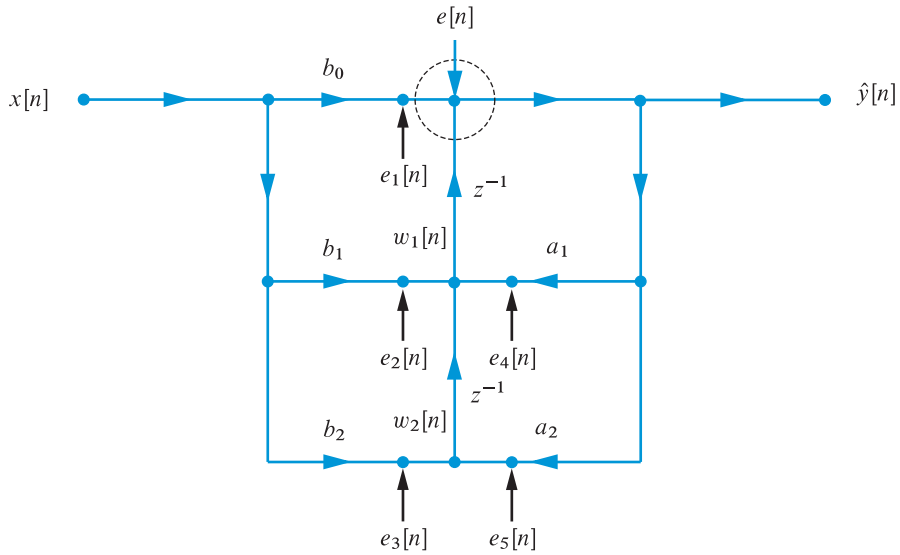
Figure 15.29 Transposed direct form II structure showing individual and combined round-off noise sources and potential overflow nodes.

The five noise sources can be combined to a single equivalent source, placed at the top adder, given by

$$e[n] = e_1[n] + e_2[n-1] + e_3[n-2] + e_4[n-1] + e_5[n-2]. \tag{15.96}$$

Because the individual noise sources are mutually uncorrelated we have

$$\mu_e = 0, \quad \sigma_e^2 = 5\frac{\Delta^2}{12}. \tag{15.97}$$

The system function between the noise source $e[n]$ and the filter output is equal to

$$H_e(z) = \frac{1}{A(z)} = \frac{1}{1 + a_1 z^{-1} + a_2 z^{-2}}. \tag{15.98}$$

The output round-off noise $g[n]$ has mean value zero and variance given by

$$\sigma_g^2 = \sigma_e^2 \sum_{n=-\infty}^{\infty} |h_e[n]|^2 = 5 \frac{\Delta^2}{12} \frac{1}{2\pi} \int_{-\pi}^{\pi} |H_e(e^{j\omega})|^2 d\omega. \tag{15.99}$$

From (15.95) and (15.99) we conclude that each structure is affected differently by round-off noise; however, we cannot say which structure has lower output round-off noise variance without assigning specific values to the coefficients of the system. This approach can be used to analyze the effect of round-off noise for any filter structure. However, the analysis for some structures may be quite complicated. These models are not entirely correct because the noise sources are not exactly white and mutually uncorrelated. However,

there is the advantage that we are not interested in an analysis with a hundred percent accuracy. Since we can vary the wordlength by one bit at a time, which corresponds to a change of SNR by a factor of two, an analysis with accuracy of 30 to 40 percent is sufficient for most applications.

Scaling to avoid overflow We now discuss the scaling procedures required to avoid overflows in the second order direct form II and transposed direct form II structures. To this end, we should examine each node of the flow graph for the possibility of overflow. As we recall from Section 15.1.2, only nodes representing addition can produce overflows. However, in two's complement fixed-point arithmetic, nodes computing partial sums are permitted to overflow as long as the final sum is within range. The direct form II structure is described by the equations

$$w[n] = -a_1 w[n-1] - a_2 w[n-2] + x[n], \tag{15.100a}$$

$$y[n] = b_0 w[n] + b_1 w[n-1] + b_2 w[n-2]. \tag{15.100b}$$

Thus, in the direct form II structure of Figure 15.28 we should properly scale the input, by a factor $S < 1$, to prevent overflows in the adders enclosed by the two dashed circles. Since $S < 1$ the signal power is reduced by a factor S^2. The product $Sx[n]$ introduces an additional round-off noise source; thus, the factor 2 in (15.92) increases by one. Thus, scaling to avoid overflow decreases the SNR at the filter output.

The transposed direct form 2 structure in Figure 15.29 is described by

$$y[n] = b_0 x[n] + w_1[n-1], \tag{15.101a}$$

$$w_1[n] = b_1 x[n] - a_1 y[n] + w_2[n-1], \tag{15.101b}$$

$$w_2[n] = b_2 x[n] - a_2 y[n]. \tag{15.101c}$$

Careful inspection of the flow graph shows that we should avoid overflow in the adder enclosed by the dashed circle; the other two adders provide partial sums and they are allowed to overflow. Thus, if the input is properly scaled to avoid overflow in $y[n]$, all internal computations are insensitive to intermediate overflows; furthermore, the scaling factor S can be absorbed by the b_k coefficients. In this respect, the transposed structure is preferable over the standard direct form II structure.

Example 15.9   Trade-off between scaling and round-off noise
To understand the trade-off between scaling to prevent overflow and expanding dynamic range to reduce round-off noise, consider the first-order IIR system shown in Figure 15.30(a) in which the input $x[n]$ is a stationary white sequence uniformly distributed between -1 and 1. After the multiplier is replaced by the equivalent noise source $e[n]$ at the circled node using the round-off quantizer model, we obtain the signal flow graph shown in Figure 15.30(b). Let $g[n]$ be the response due to $e[n]$ and let $h_e[n]$ be the impulse response between $e[n]$ and $g[n]$. Then from Figure 15.30(b),

$$h_e[n] = h[n] = a^n u[n], \tag{15.102}$$
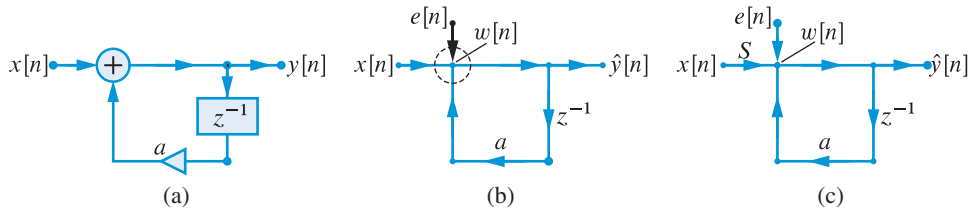
Figure 15.30 First-order IIR system in Example 15.9: (a) block diagram, (b) signal flow graph with round-off noise model, (c) scaled input in the round-off noise model.

where $h[n]$ is the impulse response of the system. Since $e[n]$ is a zero-mean white noise sequence with variance $\sigma_e^2 = 2^{-2B}/12$, $g[n]$ is also a zero-mean sequence with variance

$$\sigma_g^2 = \frac{2^{-2B}}{12} \sum_{n=0}^{\infty} (a^n)^2 = \frac{2^{-2B}}{12(1-a^2)}. \tag{15.103}$$

If the pole of the system at a is close to the unit circle, then the high gain of the system means that we have to prevent possible overflow following the adder marked by the circled node. We use the harmonic scaling in (15.87), which in this case is given by

$$S = \frac{1}{\max_{\omega} |H(e^{j\omega})|} = 1 - a. \tag{15.104}$$

Hence to prevent overflow, input $x[n]$ must satisfy $-(1-a) \le x[n] \le (1-a)$. Figure 15.30(c) shows the scaled input in the model. Now that the system is properly scaled, we can determine the signal to noise ratio. The scaled signal average power at the input is

$$\sigma_x^2 = \frac{(1-a)^2}{3}. \tag{15.105}$$

The output $y[n]$ due to $x[n]$ is also a zero-mean process with average power

$$\sigma_y^2 = \sigma_x^2 \sum_{n-0}^{\infty} |h[n]|^2 = \frac{(1-a)^2}{3(1-a^2)}, \tag{15.106}$$

while the average noise power is given by (15.103). Hence the SNR at the output is given by

$$\text{SNR}_o \triangleq \frac{\sigma_y^2}{\sigma_g^2} = 4(2^{2B})(1-a)^2 = 2^{2(B+1)}(1-a)^2, \tag{15.107}$$

or in dB the SNR is

$$\text{SNR}_o = 6.02(B+1) + 20\log_{10}(1-a) \text{ dB}. \tag{15.108}$$

Since $|a| < 1$, the second term in (15.108) is negative. Thus the output SNR decreases as the pole at a approaches the unit circle. The scaling has reduced the dynamic range of the input thereby lowering the SNR. Furthermore, the noise variance is amplified as a moves closer to unity. Thus the opposing interaction between the scaling and round-off noise reduces the overall SNR. ■

Pairing and ordering in cascade form Transposed direct form II second-order sections can be used as building blocks in parallel or cascade structures to implement higher-order filters. The total round-off noise at the output of a parallel structure is equal to the sum of the round-off noise of individual sections. However, evaluating the total round-off noise at the output of a cascade structure is more complicated because the round-off noise of each section is filtered by the poles of this section and by all subsequent sections. Therefore, pairing of poles to create second-order sections and ordering of the resulting second-order sections in a cascade structure is critical to obtain the best trade-off between scaling and round-off noise.

For most practical applications good pairing and ordering results can be obtained by applying some simple rules of thumb suggested by Jackson (1970b). The first rule is based on the observation that poles, especially the ones close to the unit circle, provide gain and as a result they can cause overflow. On the other hand zeros provide attenuation and can be used to counter the gain of the poles. The first rule is:

Rule 1. Pair the pole which is closest to the unit circle with the zero which is nearest to it. Repeat this process until all poles and zeros have been paired.

This rule is illustrated in Figure 15.31 for two sixth-order IIR filters. The second rule, which deals with the ordering of the resulting second-order sections, is:

Rule 2. Order the second-order sections obtained by using Rule 1 according to the closeness of their poles to the unit circle. Use either decreasing or increasing closeness.
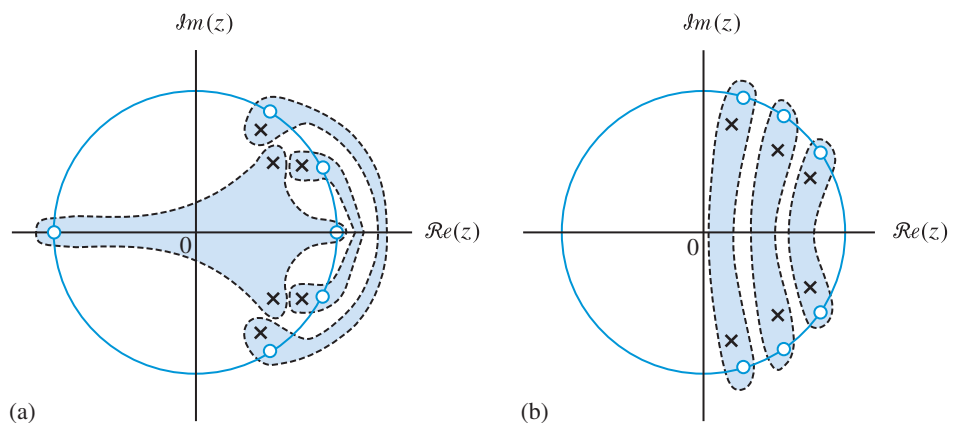
Figure 15.31 Proper pairing of poles and zeros of (a) a sixth-order bandpass filter, and (b) a sixth-order bandstop filter. See Jackson (1970b) for details.

The motivation behind this rule is to move "troubled," as far as overflow and round-off noise are concerned, sections in the beginning or end of the cascade. Clearly, such sections have strong resonances ("high Q" factor), that is, poles very close to the unit circle, and should be moved at the beginning of the system. However, high-Q sections attenuate their input signals a lot because the magnitude response is small everywhere except in the neighborhood of the resonance frequency. Thus, sometimes it is preferable to put these sections at the end of the cascade to avoid excessive reduction of the signal power at the early stages of the system. These observations indicate the multitude of factors influencing the fixed-point implementation of cascade IIR filters. The cascade structure should be preferred over the parallel structure because (a) the total output noise power of the parallel form is about the same as that of the cascade form with the best pairing and ordering, (b) for sections with zeros on the unit circle the cascade structure requires fewer multiplications, and (c) the cascade structure offers more control on the locations of the filter zeros. More in-depth information about finite wordlength effects in IIR digital filters is given by Jackson (1996) and Oppenheim and Schafer (2010).

MATLAB functions for pairing and ordering   MATLAB provides some functions that can be used for pairing and ordering of second-order sections. The important functions useful for our purpose are `tf2sos` and `zp2sos`. We describe in detail the `tf2sos`, which internally uses the `zp2sos` function. Let the system function $H(z)$ be given by

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_M z^{-M}}{a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}}, \quad M \leq N \tag{15.109a}$$

$$= G \prod_{k=1}^{K} \frac{b_{0,k} + b_{1,k} z^{-1} + b_{2,k} z^{-2}}{1 + a_{1,k} z^{-1} + a_{2,k} z^{-2}}. \quad K = \lceil N/2 \rceil \tag{15.109b}$$

The default invocation `[sos,G]=tf2sos(b,a)` converts the rational form (15.109a) into the cascade form (15.109b) with real coefficients. The vectors `b` and `a` contain the numerator and denominator coefficients in (15.109a), respectively. The matrix `sos` contains K rows, each row containing the numerator coefficients followed by the denominator coefficients in (15.109b). The scalar `G` contains the gain $G = 1/a_0$. The function uses the following steps proposed by Jackson (1970b):

- First it computes zeros and poles from `b` and `a`, respectively, using the `roots` function and calls the `zp2sos` function, which in turn combines zeros and poles into complex-conjugate pairs using the `cplxpair` function.
- Forms the second-order section by matching the pole and zero pairs according to Rule-1.
- Groups real poles (zeros) into sections with the real poles (zeros) closest to them in absolute value.
- Orders the sections according to Rule 2 using the decreasing closeness, that is, the pair closest to the unit circle is last in the cascade. This default ordering can be changed by the third input argument `order` using the `down` flag. The default `order` flag is `up`.
- Finally, if the fourth input argument `scale` is present, then scales the gain and the numerator second-order section coefficients according to value `inf` (infinity norm

scaling using (15.87)) or `two` (2-norm scaling using (15.90)). The default scaling value is `none`.

We illustrate the use of this function in the following example.

Example 15.10   Pairing, ordering, and scaling

Consider a 6th-order IIR filter given by

$$H(z) = \frac{3 - 8.642z^{-1} + 8.64z^{-2} - 8.64z^{-4} + 8.64z^{-5} - 3z^{-6}}{2 - 8.01z^{-1} + 15.26z^{-2} - 17.05z^{-3} + 11.81z^{-4} - 4.79z^{-5} + 0.93z^{-6}},$$

with all zeros on the unit circle given by

$$z_1 = 1, \; z_2 = -1, \; z_3 = e^{j20°}, \; z_4 = e^{-j20°}, \; z_5 = e^{j60°}, \; z_6 = e^{-j60°},$$

and poles at

$$p_1 = 0.9e^{j30°} \qquad p_3 = 0.85e^{j40°} \qquad p_5 = 0.89e^{j50°}$$
$$p_2 = 0.9e^{-j30°} \qquad p_4 = 0.85e^{-j40°} \qquad p_6 = 0.89e^{-j50°}.$$

The default invocation of the `tf2sos` function results in:

```
>> [sos,G] = tf2sos(b,a)
sos =
    1.0000   -0.0000   -1.0000    1.0000   -1.3023    0.7225
    1.0000   -1.0000    1.0000    1.0000   -1.1442    0.7921
    1.0000   -1.8794    1.0000    1.0000   -1.5588    0.8100
G =
    1.5000
```

To investigate the pairings and ordering strategy consider

```
>> [Z1,P1,G1] = sos2zp(sos(1,:),1);
>> Z1mag = abs(Z1.'), Z1ang = angle(Z1.')*180/pi,
Z1mag =
    1.0000    1.0000
Z1ang =
     0    180
>> P1mag = abs(P1.'), P1ang = angle(P1.')*180/pi,
P1mag =
    0.8500    0.8500
P1ang =
    40.0000   -40.0000
```

Thus the first pair is made from the two real zeros z_1, z_2 and poles p_3 and p_4 as shown in Figure 15.31(a). This second-order section is first in the cascade since the poles are furthest from the unit circle, as per the default ordering. A similar calculation shows that the next second-order section is made of zeros z_5, z_6 and poles p_5 and p_6 which are closer to the unit circle. Finally, the last second-order section is made of zeros z_3, z_4 and poles p_1 and p_2 which are closest to the unit circle. The pairings again follow the rules diagram in Figure 15.31(a). If the `order` flag is set to `down`, we obtain:

```
>> [sos,G] = tf2sos(b,a)
sos =
    1.0000   -1.8794    1.0000    1.0000   -1.5588    0.8100
    1.0000   -1.0000    1.0000    1.0000   -1.1442    0.7921
    1.0000   -0.0000   -1.0000    1.0000   -1.3023    0.7225
G =
    1.5000
```

which are the same pairs but in reverse order.

To investigate the scaling operation, we first consider the `inf` scaling with the default order:

```
>>
[sos,G] = tf2sos(b,a,'up','inf')
sos =
    0.1569   -0.0000   -0.1569    1.0000   -1.3023    0.7225
    0.4208   -0.4208    0.4208    1.0000   -1.1442    0.7921
  127.3417 -239.3240  127.3417    1.0000   -1.5588    0.8100
G =
    0.1784
```

which results in fairly large coefficient values in the last section. The `scale` flag `two` along with `order` flag `down` gives a reasonable set of coefficient values for this filter as shown below:

```
>>
[sos,G] = tf2sos(b,a,'down','two')
sos =
    1.1450   -2.1519    1.1450    1.0000   -1.5588    0.8100
    0.5263   -0.5263    0.5263    1.0000   -1.1442    0.7921
    8.3518   -0.0000   -8.3518    1.0000   -1.3023    0.7225
G =
    0.2980
```

Hence for a given filter it is essential to analyse every option to determine the best scaling and ordering strategy.   ■

Limit cycle oscillations

The round-off noise analysis that we developed in the previous sections assumed that the quantization errors were uncorrelated based on the white noise model. This assumption is valid in most situations when we have normal filter operations with many bits in the wordlength involving signals with sufficient amplitude and spectral contents. In situations when signals have very low amplitudes (almost zero input) or have periodic values tuned to the sampling rate, it is possible for the quantization errors to become *correlated* making the white noise model untenable.

When filter input is zero or constant, its output should approach zero or a steady-state constant value. However, the rounding errors prevent this and create oscillations around the steady-state value. Sometimes these oscillations have small amplitudes but sometimes can take values over the entire dynamic range. This behavior is called *limit-cycle behavior* and the appropriate model to analyze it is the basic nonlinear model and not the statistical one. Limit cycles with small amplitudes are termed *granular limit cycles* which cause audible tones in speech signals. These can be minimized using more bits in the quantizer. The limit cycles with large amplitudes are due to overflow characteristics in the quantizer and termed *overflow limit cycles*. These are problematic and should be avoided. It should be noted that limit cycles exists only in IIR systems due to feedback paths but not in FIR systems.

Granular limit cycles To understand this limit cycle, consider the first-order filter given by

$$y[n] = x[n] - 0.75y[n-1], \quad x[n] = 0.875\delta[n]. \tag{15.110}$$

The steady-state response of this filter using infinite-precision is zero. Now assume that we have 3 fractional bits in the quantizer plus a sign bit. The input sample at $n = 0$ has the binary representation $0_\triangle 111$ which is not quantized and since the input is zero for $n > 0$, there is no overflow in filter operation. The filter coefficient $0.75 \equiv 0_\triangle 110$ and thus is not quantized. The output due to the multiplication quantizer is now given by

$$\hat{y}[n] = x[n] - Q(0.75\hat{y}[n]), \quad x[n] = 0.875\delta[n]. \tag{15.111}$$

The recursive computation of $\hat{y}[n]$ in (15.111) gives

$$\hat{y}[n] = \{0.875, -0.625, 0.5, -0.375, 0.25, -0.125, 0.125, -0.125, \ldots\}. \tag{15.112}$$

Thus the resulting quantized output oscillates with values ± 0.125 with the period of 2 samples. Thus in steady-state, the 4-bit implementation has a pole on the unit circle and the nonlinear system (15.111) has effectively become a linear system, see Jackson (1996). For a general first-order system

$$\hat{y}[n] = x[n] + Q(a\hat{y}[n-1]), \tag{15.113}$$

it can be shown, see Jackson (1996), that the output settles in the range

$$\hat{y}[n] \leq \frac{2^{-(B+1)}}{1-|a|}, \quad n \gg 1 \tag{15.114}$$

for a $(B+1)$-bit quantizer. This range is called a *dead band* because when the unquantized output gets into this region, it stays there. The period of the oscillation is either one sample for $a > 0$ or two samples if $a < 0$. For a general second-order system

$$\hat{y}[n] = x[n] + Q(a_1\hat{y}[n-1]) + Q(a_2\hat{y}[n-2]), \tag{15.115}$$

the dead band region is given by

$$\hat{y}[n] \leq \frac{2^{-(B+1)}}{1-|a_2|}, \quad n \gg 1 \tag{15.116}$$

with a_1 determining the frequency of oscillations. Although granular limit cycles are a nuisance, they can be minimized using a large enough wordlength.

Overflow limit cycles This form of limit cycles is produced by the overflow characteristics of the two's-complement form arithmetic even if we ignore multiplication quantization. They are more serious than granular limit cycles because they can cover the entire dynamic range of the quantizer. The overflow characteristics are shown in Figure 15.2(b). Consider the zero-input second-order system (excited by initial conditions) given by

$$\hat{y}[n] = Q_o(1.1\hat{y}[n-1] - 0.9\hat{y}[n-2]), \tag{15.117}$$

where $Q_o(\cdot)$ is the overflow operation in addition. If $\hat{y}[n-1] = 2/3$ and $\hat{y}[n-2] = -2/3$ then from Figure 15.2(b), the output $\hat{y}[n]$ would be $-2/3$ instead of the correct $4/3$. Thus a limit cycle of amplitude $2/3$ is created by the overflow arithmetic with the period of 2 samples. Using different filter coefficients and signal values, limit cycles of any amplitude and periodicity can be obtained.

The overflow limit cycles can be eliminated if we use the saturation characteristics shown in Figure 15.2(a), which can introduce more nonlinear effects in the overall calculations. Therefore, the approach that is used in modern signal processors is to use a larger-length internal accumulator to collect additions with overflow characteristics to preserve intermediate calculations and then saturate the final result if an overflow is detected. This then effectively eliminates overflow limit cycles.

15.6   Finite wordlength effects in FFT algorithms

In this section we provide some basic results about the effects of finite wordlength fixed-point arithmetic on the performance of FFT algorithms. We use the same statistical
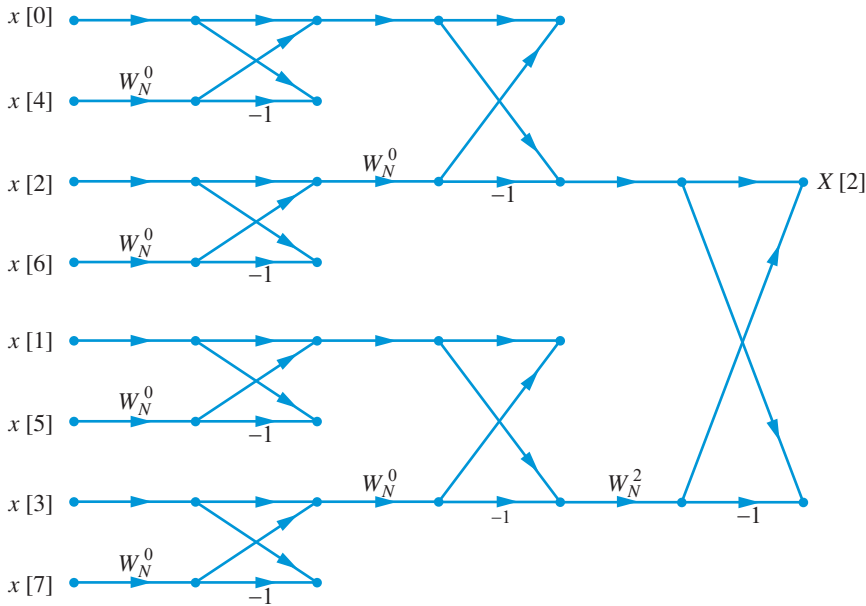
Figure 15.32 Butterflies that affect $X[2]$ in an 8-point radix-2 DIT FFT.

approach, introduced in Section 15.5 for digital filters, to analyze the effects of scaling and round-off error on the radix-2 decimation-in-time FFT algorithm; however, the results obtained are typical of other FFT algorithms as well.

To understand the basic idea we consider the 8-point decimation-in-time FFT flow graph shown in Figure 8.6. We recall that the computation of each DFT coefficient involves a series of butterfly operations. As illustrated in Figure 15.32 for $X[2]$, the computation of the $X[k]$ coefficient for an N-point DFT requires $N - 1 = 7$ butterflies. Each butterfly involves a complex multiplication by the twiddle factor W_N^r. Since each complex multiplication usually requires four real multiplications, there are four round-off noise sources per multiplication. If we make the usual assumptions that (a) all round-off noise sources are uncorrelated with each other and uncorrelated with the input sequences, and (b) the round-off noise sources are white noise processes with zero mean and variance $\sigma_e^2 = 2^{-2B}/12$, the total variance of the roundoff noise at the output $X[k]$ is given by

$$\sigma_g^2 = 4(N - 1)\frac{2^{-2B}}{12} \approx \frac{2^{-2B}N}{3}. \tag{15.118}$$

We now explain how to use scaling to prevent overflow. If $|x[n]| \leq 1$ we can prevent overflow by scaling the input by $1/N$; however, this decreases the output SNR by $1/N^2$ (see Tutorial Problem 18). Fortunately, the SNR can be improved by using a different scaling rule: instead of scaling the input signal by $1/N$, we can scale the input signals at each stage by $1/2$. Since we have $\log_2 N$ stages, each output value, $X[k]$, will be scaled by a factor of $1/N$. However, the distribution of scaling factor introduces one extra noise source at each stage and at the same time reduces the noise variance

by a factor of $(1/2)^2$. As a result, the total variance at the output is given by (see Problem 38)

$$\sigma_g^2 = \frac{4}{3} 2^{-2B} \left(1 - \frac{1}{N} \right) \approx \frac{4}{3} 2^{-2B}, \tag{15.119}$$

for $N \gg 1$. Comparison of (15.118) to (15.119) shows that the distributed scaling reduces the round-off noise power by a factor of N. To determine the SNR we must know the variance of the input signal. If $x[n]$ is a wide-sense stationary white noise process, with amplitudes uniformly distributed between $-1/N$ and $1/N$, the output signal power is given by $\sigma_g^2 = 1/(3N)$ (see Tutorial Problem 19). Therefore, the output SNR_o is

$$\text{SNR}_\text{o} = \frac{2^{2B}}{4N}, \tag{15.120}$$

which shows that the SNR_o increases proportionately to $1/N$, which is equivalent to half a bit per stage. A detailed analysis is provided by Welch (1969) and Oppenheim and Schafer (2010).

The dominant effect of scaling on the output SNR_o suggests that floating-point arithmetic, which provides automatic scaling, should improve the performance of FFT algorithms. Indeed, using single-precision or double-precision floating-point arithmetic provides almost theoretical performance. In depth analysis of floating-point FFT algorithms is provided by Weinstein (1969), Oppenheim and Weinstein (1972), and Schatzman (1996).

Learning summary

- The use of finite precision arithmetic introduces nonlinear effects into digital filters and fast Fourier transform algorithms. These effects include quantization of input signals and filter coefficients and finite precision arithmetic operations in digital filters and FFT algorithms.

- A Nyquist-rate A/D converter uses a linear quantizer which introduces quantization noise with power dependent on the number of bits, but independent of the sampling rate. The SQNR increases by about 6 dB for each extra bit.

- Oversampling spreads the quantization noise power over a wider range of frequencies. Subsequent filtering of the out-of-band noise followed by decimation increases the SQNR, which effectively increases the resolution of the quantizer. Oversampling combined with decimation, interpolation, and noise-shaping simplify the analog electronic parts of A/D and D/A converters by properly "substituting" analog filtering by digital filtering.

- Quantization of filter coefficients is a one time process that changes the filter coefficients, leading to a filter with a different frequency response. The quantized filter can be still used for the intended application as long as it satisfies the design requirements.

- Scaling to avoid overflows, control of round-off noise, and limit cycle oscillations are important considerations in implementation of digital filters with fixed-point arithmetic. In general, there is a trade-off between round-off noise and dynamic range. Round-off noise can be analyzed with adequate accuracy using statistical techniques.

- The signal-to-noise ratio at the output of an FFT algorithm implemented with fixed-point arithmetic depends on the number of input points and the scaling method used to avoid overflow. Floating-point implementations do not require scaling and provide very accurate results if we use accurate values for the twiddle factors.

TERMS AND CONCEPTS

Antialiasing filter   An analog filter needed in eliminating frequencies above the folding frequency (or half the sampling frequency).

Binary floating-point representation   A binary number representation in which the location of the binary point is changed according to the magnitude of the number using exponent bits.

Binary fixed-point representation   A binary number representation in which the location of the binary point is fixed.

Binary number representation   A number representation using binary digits or bits 0 and 1.

Binary point   A symbol similar to the decimal point that separates integer bits from fraction bits.

Finite precision   A number with finite number of bits or digits in its representation.

Finite wordlength effects   Changes in a discrete-time system output due to a fixed number of bits used in the representation of system coefficients and in its arithmetic operations.

Granular limit cycles   Limit cycles of small amplitude created due to nonlinearity in multiplication quantization.

Granular noise   The quantization error created when the given input sample lies within the quantization levels.

Infinite precision   A real number with an infinite number of bits or digits in its representation.

Limit cycle oscillations   Periodic response created at the output of a filter due to correlated quantization errors even though the input has ceased to exist.

Noise shaping converter   An A/D or D/A converter that employs strategies to push undesirable noise spectra out of the desired band.

Number representation   An arrangement of numeric symbols that provides a numeric value.

Overflow limit cycle   Limit cycles of arbitrary (and possibly large) amplitude created due to overflow operation in addition quantization.

Overflow   A situation where a number results from arithmetic operation that is outside the representable range of numbers for a given number of bits (or digits).

Overload noise   The quantization error created when the given input sample lies outside the number range for the given binary representation.

Oversampling A/D converter   An analog to digital converter in which the sampling rate is well above the required Nyquist rate.

Oversampling D/A converter   A digital to analog converter which uses digital interpolation well above the sampling rate to simplify analog interpolation.

Pairing and ordering   A procedure in creating second-order IIR filter sections that minimize various finite wordlength effects in filter calculations.

Quantization interval   The smallest interval between two quantization steps.

Quantization level   A quantized value created in a quantizer.

Quantization process   An operation by which an infinite precision number is converted to a finite number of bits or digits.

Quantization step   The smallest interval between two quantization steps.

Round-off noise analysis Analysis of errors in a filter output due to rounding quantizers used in its arithmetic.

Rounding quantizer A quantization operation in which the finite-precision conversion is done by assigning the given value to the quantization value closest to it.

SQNR – signal to quantization noise ratio The ratio of signal power to noise power. Generally expressed in decibels.

Scaling operation A strategy used in avoiding addition overflow in filter arithmetic.

Sensitivity formula An expression that provides for the amount of movement of a pole or zero given the change in filter coefficients.

Sign-magnitude format A number representation in which negative numbers are represented using a sign bit and the magnitude of the number.

Truncation quantizer A quantization process in which the finite-precision conversion is done by assigning the given value to the quantization value closest to zero.

Two's-complement format A binary number representation in which the negative numbers are represented by subtracting the magnitude from two.

Variance-gain The gain in the power at the output of a discrete-time system over the power at its input. Used in determining the effects of quantization on the input signal.

MATLAB functions and scripts

Name	Description	Page
bin2dec	Retains every Nth sample starting with the first	907
dec2bin	Resamples data at a lower rate after lowpass filtering	907
dec2beqR*	FIR decimation by an integer factor	908
dec2beqT*	Resamples data at a higher rate using lowpass interpolation	908
tf2sos	Polyphase implementation of the decimation filter	946
zp2sos	Polyphase implementation of the interpolation filter	946

*Part of the MATLAB toolbox accompanying the book.

FURTHER READING

1. A more detailed treatment of the topics discussed in this chapter, at the same level as in this book, is given by Oppenheim and Schafer (2010), Proakis and Manolakis (2007), Mitra (2006), and Porat (1997).

2. A lucid introduction to oversampling A/D and D/A conversion, including a historical development, is given by Hauser (1991) and Aziz *et al.* (1996); a complete treatment of the subject is provided by Schreier and Temes (2005).

3. Jackson (1996) provides a clear and concise introduction to finite wordlength effects on digital filters implemented using a variety of structures and fixed-point binary arithmetic. A detailed investigation of these effects using MATLAB is provided by Ingle and Proakis (2007).

4. The implementation of digital filters and FFT algorithms using DSP processors with fixed-point or floating-point arithmetic is discussed by Kuo and Gan (2005), Kuo *et al.* (2006), and Chassaing and Reay (2008). FPGA-based implementations are discussed by Woods *et al.* (2008).

Review questions

1. Finite wordlength effects in a discrete-time system have three main components. What are these components? Describe each of them concisely.
2. What is the binary number system and where do we use it?
3. There are two types of number representation: fixed-point and floating-point. Compare and contrast them.
4. Describe a general binary fixed-point number representation containing integer and fraction parts. Give some examples.
5. Describe the sign-magnitude format for representing positive and negative integers.
6. Describe the two's-complement format for representing positive and negative integers.
7. Describe the sign-magnitude format for representing positive and negative fractions.
8. Describe the two's-complement format for representing positive and negative fractions.
9. What approaches are used in the process of quantization to convert a real number into a finite-precision one?
10. Describe the characteristics of the quantization error in the rounding quantizer.
11. Describe the characteristics of the quantization error in the truncation quantizer.
12. What is an overflow? Why do they occur?
13. It is claimed that in the two's complement format, addition overflows do not matter so long as the final result is correct. Do you agree or disagree? Explain with an example.
14. What are the strategies to deal with overflows in computation? Explain using their characteristics.
15. Describe the components of floating-point representation and explain how it performs automatic tracking and manipulation of a binary point.
16. Explain the quantization principle in a uniform quantizer.
17. Show that quantization is a nonlinear function of its input.
18. What model is used to make a quantization operation a linear one?
19. Describe the conditions used in developing the statistical model of a uniform quantizer.
20. If quantization error is a random process, what kind of process is it? Provide details of its statistical properties.
21. If the number of bits in a quantizer is increased by two, by how much does the SQNR increase if all other factors are held constant?
22. Describe granular and overload quantization noise and their resulting SQNRs.
23. How is variance-gain at the output of a discrete-time system related to the system characteristics?
24. What is an oversampling A/D converter and what purpose does it serve?
25. When the sampling rate is tripled in an oversampling A/D converter, by how much do we gain in the resulting SQNR?
26. Describe the noise-shaping principle involved in sigma-delta A/D converters.

27. Describe the noise-shaping principle involved in oversampled D/A converters.
28. What role does the pole-zero clustering play in the coefficient quantization in a discrete-time system?
29. Due to coefficient quantization effects, which structures are favored in implementation and why?
30. Describe the statistical model that is used in dealing with the multiplication quantization problem. What assumptions are needed?
31. What strategy is used to avoid overflows in filter structures?
32. How do the strategies to deal with round-off noise and addition overflow interact with each other?
33. Explain the rules involved in pairing and ordering of poles and zeros to mitigate finite wordlength effects in digital filters.
34. Describe the two types of limit cycle oscillation and the cause of their existence.
35. How do we avoid granular limit cycle? What strategy is used to eliminate overflow limit cycles?
36. How is scaling strategy used to prevent overflow in FFT calculations and how does it help in increasing the output SNR?

Problems

Tutorial Problems

1. The MATLAB functions `dec2bin` and `bin2dec` convert *nonnegative* decimal numbers into binary representations and vice versa respectively.
   (a) Using `dec2bin` convert the following positive as well as negative integers into their respective sign-magnitude format binary codes:
       (a) 121, (b) -48, (c) 53, (d) -27, and (e) -347.
   (b) Using `bin2dec` convert the following sign-magnitude format binary representations of integers into decimal numbers:
       (a) 1011011, (b) 10101, (c) 01001, (d) 00101, (e) 1100110.
2. Determine the 8-bit sign-magnitude and two's-complement representations of the following decimal numbers:
   (a) 0.12345, (b) -0.54321, (c) 0.90645,
   (d) 0.45388623, (e) -0.237649.
3. Consider the sinusoidal signal $x[n]$ in (15.26). Generate 100 000 samples of $x[n]$.
   (a) Using the `dec2beqR` function and $B + 1 = 8$ bits where B is the number of fraction bits, obtain the quantization error $e[n]$. Compute and plot the histogram of $e[n]$ using 20 bins. Verify Figure 15.8(b).
   (b) Using the `psdwelch` function with window length 250 and FFT size 1024, obtain and plot the PSD of $e[n]$.
   (c) Repeat parts (a) and (b) for $B + 1 = 2, 4, 6, 10, 12, 14,$ and 16 and verify Figure 15.9.

4. Consider the quantization noise in Figure 15.6. If consecutive samples of $e[n]$ are statistically independent, then the probability density function of the sequence $e_1[n] \triangleq (e[n] + e[n-1])/2$ is given by $f_{e_1}(e_1) = f_e(e) * f_e(e)$.

   (a) Assuming that $e[n]$ follows a uniform density given in Figure 15.8(a), determine the probability density function of $e_1[n]$.

   (b) Generate 100 000 samples of the sinusoidal signal $x[n]$ in (15.26). Using the `dec2beqR` function and $B + 1 = 2$ bits where B is the number of fraction bits, obtain the signal $e_1[n]$. Compute and plot the histogram of $e_1[n]$ using 100 bins. Can you conclude from the histogram that the consecutive samples are independent? Explain.

   (c) Repeat (b) using $B + 1 = 3, 4, 5, 6, 7$, and 8. What is the minimum value of B for which you can assume that the consecutive samples are independent?

5. The granular quantization noise model is developed in Section 15.2.

   (a) Develop a MATLAB function `QNmodel` that computes statistical properties of the quantization error using B fractional bit quantization with rounding characteristics. The format of the function should be

   `[H,bins,eavg,evar]=QNmodel(x,B),`

   where `x` contains input samples, `H` contains error histogram counts, `bins` contains histogram bins, `eavg` contains the error mean estimate, and `evar` contains the error variance estimate.

   (b) Consider the signal $x[n] = 0.99 \cos(n/17)$. Generate 100 000 samples of $x[n]$ and quantize it using $B = 4$ bits. Use the `QNmodel` function to plot the histogram of the quantization error and obtain its statistics.

   (c) Comment on your results in (b) above.

6. Determine the variance-gain for the following system when the input quantization noise is propagated through it

$$H(z) = \frac{1 + 3z^{-1} - 4z^{-2}}{1 + 0.6z^{-1} + 0.08z^{-2}}.$$

7. Let $x[n]$ be a random signal in which each sample follows the normal distribution with mean 0 and variance σ_x^2. The signal is quantized using a $B+1$ bit uniform quantizer.

   (a) Using (15.32)–(15.34), show that the granular noise variance σ_g^2 and the overload noise variance σ_o^2 are given by (15.35) where η is the loading factor.

   (b) Plot the noise variances in (a) above as a function of η over a log-space from 10^{-1} to 10^3 for $B + 1 = 4, 6, 8, 10, 12, 14, 16$ and verify Figure 15.10.

8. Let $x[n]$ be a random signal in which each sample follows the normal distribution with mean 0 and variance σ_x^2. The signal is quantized using a $B+1$ bit uniform quantizer.

   (a) Generate 100 000 samples of $x[n]$ with zero mean and unit variance (that is $\eta = X_m/\sigma_x = 1$). Quantize the signal using $B + 1 = 4$ bits and compute the SQNR by estimating granular and overload noise variances.

   (b) Repeat (a) for different σ_x^2 values so that η ranges over a log-space from 10^{-1} to 10^3 to obtain a Monte-Carlo simulation of the curve in Figure 15.10 for $B + 1 = 4$ bits.

   (c) Repeat (a) and (b) to obtain Monte-Carlo curves for $B + 1 = 6, 8, 10, 12, 14$, and 16.

9. The variance-gain VG is defined in (15.37).
   (a) Substituting $H(e^{j\omega}) = H(z)|_{z=1}$ in (15.37), show that VG is given by (15.38).
   (b) Substitute $H(z)$ given by (15.41) in (15.38), perform a partial fraction expansion, and take the inverse z-transform to obtain (15.42).
10. Consider the quantization noise signal $e[n]$ in Figure 15.16 which is filtered by the noise-equivalent impulse response $h_e[n]$ in (15.58) to obtain $e_f[n]$. The PSD of $e_f[n]$ is given in (15.59b). Show that $E\{e_f^2[n]\} = 2E\{e^2[n]\}$.
11. A traditional D/A converter is operating at 30 Mhz and outputs signal with a maximum frequency of 10 Mhz.
    (a) Assuming that we want to attenuate images of the input signal below 60 dB, determine the order of a Butterworth analog anti-imaging filter.
    (b) We now increase the DAC operating frequency to 60 Hz by inserting a "zero" between each original data sample. Now determine the order of the required anti-imaging analog Butterworth filter in this interpolating DAC.
12. Let the noise shaping filter $H_e(z)$ in the oversampled noise-shaping A/D converter be a second-order filter given by

$$H(z) = (1 - z^{-1})^2.$$

Show that the signal-to-quantization noise ratio at the output of the downsampler is given by
$$SQNR_D = SQNR_{NR} - 12.90 + 15.05r \text{ (dB)},$$

where $SQNR_{NR}$ is the SQNR at the Nyquist rate and $D = 2^r$.
13. Consider the denominator of $H(z)$ in (15.66) and expressed as

$$D(z) \triangleq 1 + \sum_{k=1}^{N} a_k z^{-k} = \prod_{j=1}^{N} (1 - p_j z^{-1}), \tag{15.121}$$

where $\{p_j\}$ are the system poles.
   (a) Show that

$$\frac{\partial p_i}{\partial a_k} = \frac{\left.\frac{\partial D(z)}{\partial a_k}\right|_{z=p_i}}{\left.\frac{\partial D(z)}{\partial p_i}\right|_{z=p_i}}. \tag{15.122}$$

   (b) Now using (15.122) prove (15.70).
14. Design a digital lowpass filter using the Chebyshev II prototype to satisfy the requirements: $\omega_p = 0.2\pi$, $\omega_s = 0.3\pi$, $A_p = 1$ dB, and $A_s = 50$ dB.
    (a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.
    (b) Quantize the direct form coefficients to $L = 16$ bits using the dec2beqR function and plot the resulting magnitude response and pole-zero diagram.
    (c) Quantize the direct form coefficients to $L = 12$ bits and plot the resulting magnitude response and pole-zero diagram.
    (d) Quantize the direct form coefficients to $L = 8$ bits and plot the resulting magnitude response and pole-zero diagram.
    (e) Comment on your plots.

15. Consider the signal

$$x[n] = \big[\cos(n/11) + \sin(n/17) + \cos(n/31)\big]/3.$$

It is multiplied by a constant $a = 0.9375$ and then quantized to B fraction bits. Let $e[n] = Q(ax[n]) - ax[n]$. Generate 100 000 samples of $x[n]$ and use the `QNmodel` function introduced in Problem 5 to answer the following parts.

(a) Quantize $ax[n]$ to $B = 4$ bits and plot the histogram of the resulting error sequence.

(b) Quantize $ax[n]$ to $B = 8$ bits and plot the histogram of the resulting error sequence.

(c) Quantize $ax[n]$ to $B = 10$ bits and plot the histogram of the resulting error sequence.

(d) Comment on your histogram plots and on the validity of the multiplication quantization model.

16. Consider a first-order IIR system given by $y[n] = x[n] + 0.375y[n-1]$. Input to the filter is $x[n] = \cos(n/7)$. It is scaled according to the harmonic scaling to avoid overflow and quantized to B fractional bits before filtering. The multiplication $0.375y[n-1]$ is also quantized to B bits. Generate 100 000 samples of $x[n]$ and assume rounding operations.

(a) Determine the output $y[n]$ assuming infinite precision in the multiplier.

(b) Using $B = 4$ bit multiplication quantization in the filter implementation, determine the resulting output $\hat{y}[n]$, the error $g[n] = \hat{y}[n] - y[n]$, and the estimated output SNR. Provide a plot of the error histogram.

(c) Repeat part (b) using $B = 8$ bits.

(d) How does the estimated SNR compare with (15.107) in the above parts?

17. Consider a first-order IIR system $y[n] = 0.5\delta[n] + 0.625y[n-1]$ with zero initial condition. The filter uses a $B + 1 = 4$ bit multiplier and rounding.

(a) Assuming that two's-complement overflow is used in the addition, compute and plot the first 20 samples of $y[n]$. Does the output display any oscillations? If it does then determine their amplitude and frequency.

(b) Assuming that saturation characteristics are used in the addition, compute and plot the first 20 samples of $y[n]$. Does the output display any oscillations? If it does then determine their amplitude and frequency.

18. Consider finite-wordlength effects in a DIT-FFT algorithm.

(a) To avoid overflow in the calculations and to guarantee that $|X(k)| < 1$ for $0 \leq k < N$ show that it is necessary and sufficient that the input complex-valued signal satisfies $|x[n]| < 1/N$ for $0 \leq n < N$.

(b) If the input $|x[n]| < 1$ is scaled by $1/N$, then show that the output SNR decreases by $1/N^2$.

19. If the input $|x[n]| < 1$ is scaled by $1/N$, then show that the output signal power in (15.120) is given by $\sigma_x^2 = 1/(3N)$.

Basic problems

20. Determine the 10-bit sign-magnitude and two's-complement representations of the following decimal numbers:

(a) 0.12345, (b) −0.54321, (c) 0.90645,

(d) 0.45388623, (e) −0.237649.

21. Let $x[n]$ be an IID random process where each sample is uniformly distributed over $[-1, 1]$. Generate 100 000 samples of $x[n]$.
    (a) Using `dec2beqR` function and $B + 1 = 8$ bits where B is the number of fraction bits, obtain the quantization error $e[n]$. Compute and plot the histogram of $e[n]$ using 20 bins.
    (b) Using the `psdwelch` function with window length 250 and FFT size 1024, obtain and plot the PSD of $e[n]$.
    (c) Repeat parts (a) and (b) for $B + 1 = 2, 4, 6, 10, 12, 14$, and 16 and obtain a plot similar to Figure 15.9. Comment on the plot.

22. Consider the second-order system with complex-conjugate poles at $re^{\pm j\theta}$, zeros at $z = \pm 1$, and gain of 1. Assume $0 < r < 1$.
    (a) Using (15.42), obtain a formula for the variance-gain VG in terms of r and θ.
    (b) Determine VG for $r = 0.9$ and $\theta = \pi/4$.
    (c) Repeat (b) for $r = 0.99$ and $\theta = \pi/2$.
    (d) Verify your results in (b) and (c) using (15.43).

23. Let $x[n]$ be a random signal in which each sample follows the uniform distribution over $[-A, A]$. The signal is quantized using a $B + 1$ bit uniform quantizer.
    (a) Generate 10 000 samples of $x[n]$ with zero mean and $A = \sqrt{3}$ (that is $\eta = X_m/\sigma_x = 1$). Quantize the signal using $B + 1 = 4$ bits and compute the SQNR by estimating granular and overload noise variances.
    (b) Repeat (a) for different A values so that η ranges over a log-space from 10^{-1} to 10^3 to obtain a Monte-Carlo simulation of the curve similar to the one in Figure 15.10 for $B + 1 = 4$ bits.
    (c) Repeat (a) and (b) to obtain Monte-Carlo curves for $B + 1 = 6, 8, 10, 12, 14$, and 16.

24. For the additive quantization noise in Figure 15.6 show that the variance of the granular quantization noise $e[n]$ is given by $\sigma_e^2 = \Delta^2/12$ where Δ is the step size in the uniform quantizer when there is no overload noise.

25. Determine the variance-gain for the following system when the input quantization noise is propagated through it:

$$H(z) = \frac{1 - 8z^{-1} + 19z^{-2} - 12z^{-3}}{1 - 0.4z^{-1} - 0.2375z^{-2} - 0.0188z^{-3}}.$$

26. Let $x[n]$ be an IID random sequence with uniform distribution over $[-1, 1]$. Generate 100 000 samples of $x[n]$.
    (a) Using `QNmodel` developed in Problem 5 obtain the quantization histogram and the statistics when the signal is quantized to $B = 3$ bits.
    (b) Repeat (a) using $B = 8$.
    (c) Comment on your results in (a) and (b).

27. Show that the signal power at the output of a downsampler in Figure 15.14 is given by σ_x^2 and that the noise power at the output of a downsampler is given by σ_e^2/D.

28. Show that the poles of the coupled-form system in Figure 15.22 are given by $r\cos(\theta)$ and $r\sin(\theta)$.

29. Consider the numerator of $H(z)$ in (15.66) and expressed as

$$C(z) \triangleq \sum_{k=0}^{M} b_k z^{-k} = \prod_{j=1}^{M} (1 - z_j z^{-1}), \qquad (15.123)$$

where $\{z_j\}$ are the system zeros.
(a) Show that

$$\frac{\partial z_i}{\partial b_k} = \frac{\left.\frac{\partial C(z)}{\partial b_k}\right|_{z=z_i}}{\left.\frac{\partial C(z)}{\partial z_i}\right|_{z=z_i}}. \qquad (15.124)$$

(b) Using (15.124) derive the zero sensitivity formula

$$\Delta z_i = -\sum_{k=1}^{M} \frac{z_i^{M-k}}{\prod_{j=1, j\neq i}^{M}(z_i - z_j)} \Delta b_k. \qquad (15.125)$$

30. A digital filter is given by the system function

$$H(z) = \frac{1 - \sqrt{3}z^{-1}}{1 - z^{-1}/\sqrt{3}}, \quad |z| > 1/\sqrt{3}. \qquad (15.126)$$

(a) Show that the above filter is an allpass filter. Plot its magnitude response over $-\pi \leq \omega \leq \pi$ and verify.
(b) Round the filter coefficients to $B = 8$ fraction bits using the dec2beqR function and then plot the magnitude response of the resulting filter. Is the filter allpass?
(c) Round the filter coefficients to $B = 4$ fraction bits using the dec2beqR function and then plot the magnitude response of the resulting filter. Is the filter still allpass?

31. Design a digital lowpass filter using the Chebyshev II prototype to satisfy the requirements: $\omega_p = 0.25\pi$, $\omega_s = 0.35\pi$, $A_p = 0.5$ dB, and $A_s = 50$ dB.
(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.
(b) Quantize the cascade form coefficients to $L = 16$ bits using the dec2beqR function and plot the resulting magnitude response and pole-zero diagram.
(c) Quantize the cascade form coefficients to $L = 12$ bits and plot the resulting magnitude response and pole-zero diagram.
(d) Quantize the cascade form coefficients to $L = 8$ bits and plot the resulting magnitude response and pole-zero diagram.
(e) Comment on your plots.

32. Design a digital bandstop filter using the elliptic prototype to satisfy the requirements: $\omega_{p_1} = 0.25\pi$, $\omega_{s_1} = 0.3\pi$, $\omega_{s_2} = 0.5\pi$, $\omega_{p_2} = 0.6\pi$, $A_p = 1$ dB, and $A_s = 60$ dB.
(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.
(b) Quantize the direct form coefficients to $L = 16$ bits using the dec2beqR function and plot the resulting magnitude response and pole-zero diagram.
(c) Quantize the direct form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.

(d) Quantize the cascade form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.

(e) Comment on your plots.

33. Design a digital highpass filter using the Butterworth prototype to satisfy the requirements: $\omega_s = 0.65\pi$, $\omega_p = 0.8\pi$, $A_s = 50$ dB, and $A_s = 1$ dB.

(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.

(b) Quantize the cascade form coefficients to $L = 16$ bits using the `dec2beqR` function and plot the resulting magnitude response and pole-zero diagram.

(c) Quantize the cascade form coefficients to $L = 12$ bits and plot the resulting magnitude response and pole-zero diagram.

(d) Quantize the cascade form coefficients to $L = 8$ bits and plot the resulting magnitude response and pole-zero diagram.

(e) Comment on your plots.

34. Design a digital bandpass filter using the Chebyshev I prototype to satisfy the requirements: $\omega_{s_1} = 0.3\pi$, $\omega_{p_1} = 0.4\pi$, $\omega_{p_2} = 0.6\pi$, $\omega_{s_2} = 0.7\pi$, $A_p = 1$ dB, and $A_s = 50$ dB.

(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.

(b) Quantize the direct form coefficients to $L = 16$ bits using the `dec2beqR` function and plot the resulting magnitude response and pole-zero diagram.

(c) Quantize the direct form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.

(d) Quantize the cascade form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.

(e) Comment on your plots.

35. Let $x[n]$ be an IID sequence where each sample is uniformly distributed over $[-1, 1]$. It is multiplied by a constant $a = 0.6667$ and then quantized to B fraction bits. Let $e[n] = Q(ax[n]) - ax[n]$. Generate $100\,000$ samples of $x[n]$ and use the `QNmodel` function introduced in Problem 5 to answer the following parts.

(a) Quantize $ax[n]$ to $B = 3$ bits and plot the histogram of the resulting error sequence.

(b) Quantize $ax[n]$ to $B = 6$ bits and plot the histogram of the resulting error sequence.

(c) Quantize $ax[n]$ to $B = 9$ bits and plot the histogram of the resulting error sequence.

(d) Comment on your histogram plots and on the validity of the multiplication quantization model.

36. Consider an FIR system given by $y[n] = 0.25x[n] + 0.5x[n-1] + 0.25x[n-2]$. It is implemented in a direct form using $B = 8$ bits. Input to the filter is $x[n] = \cos(n/11)$. It is scaled according to (15.90) to avoid overflow and quantized to B fractional bits before filtering. All multiplications are also quantized to B bits. Generate $100\,000$ samples of $x[n]$ and assume rounding operations.

(a) Determine the output $y[n]$ assuming infinite precision in the multipliers.

(b) Using all three B bit multipliers in the filter implementation, determine the resulting output $\hat{y}[n]$, the error $g[n] = \hat{y}[n] - y[n]$, and the estimated output SNR. Provide a plot of the error histogram.

(c) Repeat part (b) but using only one multiplier in the filter implementation.

37. Consider a first-order IIR system $y[n] = 0.75\delta[n] + 0.75y[n-1]$ with zero initial condition. The filter uses a $B + 1 = 5$ bit multiplier and rounding.

(a) Assuming that two's-complement overflow is used in the addition, compute and plot the first 20 samples of $y[n]$. Determine the amplitude and frequency of oscillations, if any.

(b) Assuming that saturation characteristics are used in the addition, compute and plot the first 20 samples of $y[n]$. Determine the amplitude and frequency of oscillations, if any.

38. Using the description about the scaling at each stage of the FFT algorithm given in the text, derive (15.119).

Assessment problems

39. Determine the 8-bit sign-magnitude and two's-complement representations of the following decimal numbers:

(a) 0.54321, (b) −0.12345, (c) 0.54609,

(d) 0.862338, (e) −0.497623.

40. Consider the second-order system with real poles at $\pm r$, zeros at $z = \pm j$, and gain of 1. Assume $0 < r < 1$.

(a) Using (15.42), obtain a formula for the variance-gain VG in terms of r.

(b) Determine VG for $r = 0.9$.

(c) Repeat (b) for $r = 0.99$.

(d) Verify your results in (b) and (c) using (15.43).

41. Let $x[n]$ be an IID random process where each sample is Gaussian distributed with zero mean and variance of $1/16$. Generate 100 000 samples of $x[n]$ and make sure that the samples are between the range $-1 \le x[n] < 1$ by clipping, if necessary.

(a) Using `dec2beqR` function and $B + 1 = 8$ bits where B is the number of fraction bits, obtain the quantization error $e[n]$. Compute and plot the histogram of $e[n]$ using 20 bins.

(b) Using the `psdwelch` function with window length 250 and FFT size 1024, obtain and plot the PSD of $e[n]$.

(c) Repeat parts (a) and (b) for $B + 1 = 2, 4, 6, 10, 12, 14,$ and 16 and obtain a plot similar to Figure 15.9. Comment on the plot.

42. Let $x[n]$ be a sinusoidal signal given by $x[n] = 0.49[\cos(n/11) + \sin(n/31)]$. Generate 100 000 samples of $x[n]$.

(a) Using `QNmodel` developed in Problem 5 obtain the quantization histogram and the statistics when the signal is quantized to $B = 2$ bits.

(b) Repeat (a) using $B = 10$.

(c) Comment on your results in (a) and (b).

43. Determine the variance-gain for the following system when the input quantization noise is propagated through it:

$$H(z) = \frac{1 + 3z^{-1} - 4z^{-2}}{1 + 0.6z^{-1} + 0.08z^{-2}}.$$

44. Let $x[n]$ be a random signal in which each sample follows the Laplacian distribution $f_x(x) = 1/(\sigma\sqrt{2}) \exp(-\sqrt{2}|x|/\sigma)$ where σ is the standard deviation. This signal models speech sources. The signal is quantized using a $B + 1$ bit uniform quantizer.

(a) Generate 10 000 samples of $x[n]$ with zero mean and $\sigma = 1$ (that is $\eta = X_{\text{m}}/\sigma_x = 1$). Quantize the signal using $B + 1 = 4$ bits and compute the SQNR by estimating granular and overload noise variances.

(b) Repeat (a) for different σ values so that η ranges over a log-space from 10^{-1} to 10^3 to obtain a Monte-Carlo simulation of the curve similar to the one in Figure 15.10 for $B + 1 = 4$ bits.

(c) Repeat (a) and (b) to obtain Monte-Carlo curves for $B + 1 = 6, 8, 10, 12, 14$, and 16.

45. Let

$$H(z) = \frac{1}{\left(1 + a_1 z^{-1} + a_2 z^{-2}\right)} = \frac{1}{\left(1 - r\, e^{j\theta} z^{-1}\right)\left(1 - r\, e^{-j\theta} z^{-1}\right)}$$

be a second-order filter implemented in a direct-form structure where $a_1 = -2r\cos\theta$ and $a_2 = r^2$.

(a) Assume that the coefficients a_1 and a_2 are represented in sign-magnitude format with 3 bits. Then there are 7 nonzero values for a_1 and a_2 each in the first quadrant of the z-plane. Using MATLAB, plot the pole distribution of a stable second-order structure in the first quadrant.

(b) Assume that r is represented in sign-magnitude format with 3 bits and that the angle θ is represented using 8 uniform levels between $0°$ and $90°$. Using MATLAB, plot the pole distribution in the first quadrant.

(c) Compare the number of possible pole positions in each case above.

46. Design a digital lowpass filter using the Chebyshev I prototype to satisfy the requirements: $\omega_p = 0.15\pi$, $\omega_s = 0.25\pi$, $A_p = 1$ dB, and $A_s = 50$ dB.

(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \le \omega \le \pi$.

(b) Quantize the cascade form coefficients to $L = 16$ bits using the `dec2beqR` function and plot the resulting magnitude response and pole-zero diagram.

(c) Quantize the cascade form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.

(d) Quantize the cascade form coefficients to $L = 6$ bits and plot the resulting magnitude response and pole-zero diagram.

(e) Comment on your plots.

47. Design a digital bandpass filter using the elliptic prototype to satisfy the requirements: $\omega_{s_1} = 0.25\pi$, $\omega_{p_1} = 0.3\pi$, $\omega_{p_2} = 0.5\pi$, $\omega_{s_2} = 0.6\pi$, $A_p = 0.5$ dB, and $A_s = 60$ dB.

(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \le \omega \le \pi$.

(b) Quantize the direct form coefficients to $L = 16$ bits using the `dec2beqR` function and plot the resulting magnitude response and pole-zero diagram.

(c) Quantize the direct form coefficients to $L = 12$ bits and plot the resulting magnitude response and pole-zero diagram.

(d) Quantize the cascade form coefficients to $L = 12$ bits and plot the resulting magnitude response and pole-zero diagram.

(e) Comment on your plots.

48. Design a digital highpass filter using the Chebyshev II prototype to satisfy the requirements: $\omega_s = 0.7\pi$, $\omega_p = 0.8\pi$, $A_s = 60$ dB, and $A_s = 1$ dB.

(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.
(b) Quantize the direct form coefficients to $L = 16$ bits using the `dec2beqR` function and plot the resulting magnitude response and pole-zero diagram.
(c) Quantize the direct form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.
(d) Quantize the cascade form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.
(e) Comment on your plots.

49. Design a digital bandstop filter using the Butterworth prototype to satisfy the requirements: $\omega_{p_1} = 0.3\pi$, $\omega_{s_1} = 0.4\pi$, $\omega_{s_2} = 0.6\pi$, $\omega_{p_2} = 0.7\pi$, $A_p = 1$ dB, and $A_s = 60$ dB.
(a) Plot the magnitude response and pole-zero diagram of the filter over $0 \leq \omega \leq \pi$.
(b) Quantize the direct form coefficients to $L = 16$ bits using the `dec2beqR` function and plot the resulting magnitude response and pole-zero diagram.
(c) Quantize the direct form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.
(d) Quantize the cascade form coefficients to $L = 10$ bits and plot the resulting magnitude response and pole-zero diagram.
(e) Comment on your plots.

50. Consider the signal

$$x[n] = 0.49\big[\cos(n/13) + \sin(n/29)\big].$$

It is multiplied by a constant $a = 0.3333$ and then quantized to B fraction bits. Let $e[n] = Q(ax[n]) - ax[n]$. Generate 100 000 samples of $x[n]$ and use the `QNmodel` function introduced in Problem 5 to answer the following parts.
(a) Quantize $ax[n]$ to $B = 3$ bits and plot the histogram of the resulting error sequence.
(b) Quantize $ax[n]$ to $B = 6$ bits and plot the histogram of the resulting error sequence.
(c) Quantize $ax[n]$ to $B = 12$ bits and plot the histogram of the resulting error sequence.
(d) Comment on your histogram plots and on the validity of the multiplication quantization model.

51. Consider an FIR system given by $y[n] = 0.1x[n] + 0.2x[n-1] + 0.3x[n-2] + 0.2x[n-3] + 0.1x[n-4]$. It is implemented in a direct form using $B = 8$ bits. Input to the filter is $x[n] = \cos(n/13)$. It is scaled according to the harmonic scaling to avoid overflow and quantized to B fractional bits before filtering. All multiplications are also quantized to B bits. Generate 100 000 samples of $x[n]$ and assume rounding operations.
(a) Determine the output $y[n]$ assuming infinite precision in the multipliers.
(b) Using all five B bit multipliers in the filter implementation, determine the resulting output $\hat{y}[n]$, the error $g[n] = \hat{y}[n] - y[n]$, and the estimated output SNR. Provide a plot of the error histogram.
(c) Repeat part (b) but using only one multiplier in the filter implementation.

52. Consider a first-order IIR system given by $y[n] = x[n] - 0.875y[n-1]$. Input to the filter is an IID random sequence with samples uniformly distributed over $[-1, 1]$. It is scaled according to the harmonic scaling to avoid overflow and quantized to B

fractional bits before filtering. The multiplication $0.375y[n-1]$ is also quantized to B bit. Generate 100 000 samples of $x[n]$ and assume rounding operations.

(a) Determine the output $y[n]$ assuming infinite precision in the multiplier.

(b) Using $B = 5$ bit multiplication quantization in the filter implementation, determine the resulting output $\hat{y}[n]$, the error $g[n] = \hat{y}[n] - y[n]$, and the estimated output SNR. Provide a plot of the error histogram.

(c) Repeat part (b) using $B = 10$ bits.

(d) How does the estimated SNR compare with (15.107) in the above parts?

53. Consider a second-order IIR system $y[n] = 0.75\delta[n] - 0.5y[n-2]$ with zero initial condition. The filter uses a $B + 1 = 4$ bit multiplier and rounding.

(a) Assuming that two's-complement overflow is used in the addition, compute and plot the first 20 samples of $y[n]$. Determine the amplitude and frequency of oscillations, if any.

(b) Assuming that saturation characteristics are used in the addition, compute and plot the first 20 samples of $y[n]$. Determine the amplitude and frequency of oscillations, if any.

Review problems

54. Design a 5th-order elliptic lowpass filter with $\omega_p = 0.2\pi$, $A_p = 1$ dB, and $A_s = 40$ dB. Let the input to the filter be 100 000 samples of $x[n] = 0.99\cos(n/19)$. The filtering operation is to be implemented using $B = 5$ fractional bits with direct form structure.

(a) Assuming an infinite-precision representation and arithmetic, compute the output $y[n]$.

(b) Quantize input values to B bits. Assume that filter representation and multiplication operations have infinite precision. Determine the output $\hat{y}_1[n]$ and the error $g_1[n] = \hat{y}_1[n] - y[n]$. Determine the mean and variance of $g_1[n]$ and plot its histogram.

(c) Quantize filter coefficients to B bits (in addition to integer and sign bits). Assume that input representation and multiplication operations have infinite precision. Determine the output $\hat{y}_2[n]$ and the error $g_2[n] = \hat{y}_2[n] - y[n]$. Determine the mean and variance of $g_2[n]$ and plot its histogram.

(d) Now assume that input values and filter coefficients are quantized to B bits but the filter arithmetic is of infinite precision. Determine the output $\hat{y}_3[n]$ and the error $g_3[n] = \hat{y}_3[n] - y[n]$. Determine the mean and variance of $g_3[n]$ and plot its histogram.

(e) Finally assume that all representations and arithmetic operations are quantized to B bits. Scale the input values so that there are no overflows in the addition using the harmonic scaling. Determine the output $\hat{y}_4[n]$ and the error $g_4[n] = \hat{y}_4[n] - y[n]$. Determine the mean and variance of $g_4[n]$ and plot its histogram.

(f) How do the various errors compare in the above parts?

55. Design a 5th-order elliptic lowpass filter with $\omega_p = 0.2\pi$, $A_p = 1$ dB, and $A_s = 40$ dB. Let the input to the filter be 100 000 samples of uniformly distributed IID random sequence over $[-1, 1]$. The filtering operation is to be implemented using $B = 5$ fractional bits with cascade form structure.

(a) Using the `tf2sos`, default ordering, and norm-2 scaling obtain the cascade form structure of the filter.

(b) Assuming an infinite-precision representation and arithmetic, compute the output $y[n]$.

(c) Quantize input values to B bits. Assume that filter representation and multiplication operations have infinite precision. Determine the output $\hat{y}_1[n]$ and the error $g_1[n] = \hat{y}_1[n] - y[n]$. Determine the mean and variance of $g_1[n]$ and plot its histogram.

(d) Quantize filter coefficients to B bits (in addition to integer and sign bits). Assume that input representation and multiplication operations have infinite precision. Determine the output $\hat{y}_2[n]$ and the error $g_2[n] = \hat{y}_2[n] - y[n]$. Determine the mean and variance of $g_2[n]$ and plot its histogram.

(e) Now assume that input values and filter coefficients are quantized to B bits but the filter arithmetic is of infinite precision. Determine the output $\hat{y}_3[n]$ and the error $g_3[n] = \hat{y}_3[n] - y[n]$. Determine the mean and variance of $g_3[n]$ and plot its histogram.

(f) Finally assume that all representations and arithmetic operations are quantized to B bits. Scale the input values so that there are no overflows in the addition using the harmonic scaling. Determine the output $\hat{y}_4[n]$ and the error $g_4[n] = \hat{y}_4[n] - y[n]$. Determine the mean and variance of $g_4[n]$ and plot its histogram.

(g) How do the various errors compare in the above parts?

REFERENCES

A. Akansu and R. Haddad. *Multiresolution Signal Decomposition*. Academic Press, San Diego, 2nd edition, 2001.

A. Antoniou. *Digital Signal Processing*. McGraw-Hill, New York, 2006.

P. Aziz, H. Sorensen, and J. Van Der Spiegel. An overview of sigma-delta converters. *IEEE Signal Processing Magazine*, 61–84, January 1996.

W. R. Bennet. Spectra of quantized signals. *Bell System Technical Journal*, **27**:446–472, July 1948.

J. Berrut and L. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, **46**(3):501–517, 2004.

R. E. Blahut. *Fast Algorithms for Signal Processing*. Cambridge University Press, New York, 2010.

L. Bluestein. A linear filtering approach to the computation of Discrete Fourier Transform. *IEEE Transactions on Audio and Electroacoustics*, **18**:451–455, 1970.

H. Bode and C. Shannon. A simplified derivation of linear least square smoothing and prediction theory. *Proceedings of the IRE*, **38**:417–425, April 1950.

K. Bollacker. Avoiding a digital dark age. *American Scientist*, **98**:106–110, March-April 2010.

M. Bosi and R. E. Goldberg. *Introduction to Digital Audio Coding and Standards*. Kluwer Academic Publishers, Boston, 2003.

G. Box, G. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting and Control*. Wiley, New York, 2008.

R. N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, New York, NY, 2nd edition, 2000.

E. Brigham. *Fast Fourier Transform and Its Applications*. Prentice Hall, Upper Saddle River, NJ, 1988.

J. W. Brown and R. V. Churchill. *Complex Variables and Applications*. McGraw-Hill, New York, NY, 7th edition, 2004.

C. S. Burrus, Ramesh A Gopinath, and Haitao Guo. *Introduction to Wavelets and Wavelet Transforms: A Primer*. Prentice Hall, Upper Saddle River, NJ, 1998.

C. S. Burrus and T. W. Parks. *DFT/FFT and Convolution Algorithms: Theory and Implementation*. Wiley, New York, 1985.

C. S. Burrus, A. W. Soewito, and R. A. Gopinath. Least squared error FIR filter design with transition bands. *IEEE Transactions on Signal Processing*, **40**(6):1327–1340, June 1992.

S. Butterworth. On the theory of filter amplifiers. *Experimental Wireless & The Wireless Engineer*, **7**:536–541, October 1930.

E. Cauer, W. Mathis, and R. Pauli. Life and work of Wilhelm Cauer (1900–1945). In *Proceedings of the Fourteenth International Symposium of Mathematical Theory of Networks and Systems (MTNS2000)*, Perpignan, June 2000.

W. Cauer. New theory and design of wave filters. *Physics*, **2**(4):242–268, April 1932.

D. S. Chan and L. R. Rabiner. Analysis of quantization errors in the direct form for finite impulse response digital filters. *IEEE Transactions on Audio and Electroacoustics*, **21**(4):354–366, August 1973.

R. Chassaing and D. Reay. *Digital Signal Processing and Applications with the TMS320C6713 and TMS320C6416 DSK*. Wiley, Hoboken, New Jersey, 2008.

E. W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, New York, NY, 1966.

T. A. Claasen and W. F. Mecklenbräuker. On the transposition of linear time-varying discrete-time networks and its application to multirate digital systems. *Philips J. Res.*, **23**:78–102, 1978.

M. Clements and J. Pease. On causal linear phase IIR digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**(4):479–484, April 1989.

W. Cochran, J. Cooley, D. Favin, *et al.* What is the Fast Fourier Transform? *IEEE Transactions on Audio and Electroacoustics*, **15**(2):45–55, June 1967.

A. G. Constantinides. Spectral transformations for digital filters. *Proceedings of the Institution of Electrical Engineers*, **117**(8):1585–1590, August 1970.

J. Cooley. How the FFT gained acceptance. *IEEE Signal Processing Magazine*, **9**(1):10–13, January 1992.

J. Cooley, P. Lewis, and P. Welch. Historical notes on the Fast Fourier Transform. *Proceedings of the IEEE*, **55**(10):1675–1677, October 1967.

J. Cooley, P. Lewis, and P. Welch. The finite Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, **17**(2):77–85, June 1969.

J. Cooley and J. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, **19**(90):297–301, 1965.

A. J. Coulson. A generalization of nonuniform bandpass sampling. *IEEE Transactions on Signal Processing*, **43**(3):694–704, March 1995.

R. E. Crochiere. A general program to perform sampling rate conversion of data by rational ratios. In *Programs for Digital Signal Processing*, pages 8.2.1–8.2.7. IEEE Press, New York, 1979.

R. E. Crochiere and L. R. Rabiner. Interpolation and decimation of digital signals – a tutorial review. *Proceedings of the IEEE*, **69**(3):300–331, March 1981.

R. E. Crochiere and L. R. Rabiner. *Multirate Digital Signal Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 1983.

A. Croisier, D. Esteban, and C. Galand. Perfect channel splitting by use of interpolation /decimation/tree decomposition techniques. In *International conference on Information Sciences and Systems*, 191–195, 1976.

R. W. Daniels. *Approximation Methods for Electronic Filter Design*. McGraw-Hill, New York, NY, 1974.

I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, **41**(7):909–996, 1988.

W. Davenport. *Probability and Random Processes*. McGraw-Hill, New York, NY, 1970.

W. Davenport and W. Root. *An Introduction to the Theory of Random Signals and Noise*. Wiley-IEEE Press, New York, NY, 1987.

R. C. Dorf and R. H. Bishop. *Modern Control Systems*. Prentice Hall, Upper Saddle River, NJ, 11th edition, 2008.

P. Duhamel. Implementation of "Split-radix" FFT algorithms for complex, real, and real-symmetric data. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **34**(2):285–295, April 1986.

P. Duhamel and M. Vetterli. Fast Fourier Transforms: A tutorial review and a state of the art. *Signal Processing*, **19**(4):259–299, 1990.

D. Evans. An improved digit-reversal permutation algorithm for the fast Fourier and Hartley transforms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **35**(8):1120–1125, August 1987.

W. Fischer. *Digital Video and Audio Broadcasting Technology: A Practical Engineering Guide*. Springer-Verlag, Berlin, Second edition, 2008.

N. Fliege. *Multirate Digital Signal Processing*. Wiley, New York, NY, 1994.

J. Foley, A. van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice in C*. Addison-Wesley, New York, 2nd edition, 1995.

F. Franchetti, M. Puschel, Y. Voronenko, S. Chellappa, and J. Moura. Discrete Fourier Transform on multicore. *IEEE Signal Processing Magazine*, **26**(6):90–102, November 2009.

M. Frigo and S. G. Johnson. FFTW: An adaptive software architecture for the FFT. In *IEEE ICASSP 1998*, volume 3, 1381–1384, 1998.

M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, **93**(2):216–231, February 2005.

K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, San Diego, 2nd edition, 1990.

A. Gersho. Principles of quantization. *IEEE Transactions on Circuits and Systems*, **25**(7):427–436, July 1978.

G. Goertzel. An algorithm for the evaluation of finite trigonometric series. *American Math. Monthly*, **65**:34–35, January 1958.

B. Gold and C. Rader. *Digital Processing of Signals*. McGraw-Hill, New York, NY, 1969.

D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, **23**:5–48, 1991.

R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, Upper Saddle River, NJ, 3rd edition, 2008.

G. A. Gray and G. W. Zeoli. Quantization and saturation noise due to analog-to-digital conversion. *IEEE Transactions on Aerospace and Electronic Systems*, AES-7(1):222–223, January 1971.

R. Gray and L. Davisson. *An Introduction to Statistical Signal Processing*. Cambridge University Press, Cambridge, 2004.

R. M. Gray. Quantization noise spectra. *IEEE Transactions on Information Theory*, **36**(6):1220–1244, November 1990.

E. A. Guillemin. *Synthesis of Passive Networks*. Wiley, New York, 1957.

H. Guo, G. Sitton, and C. Burrus. The Quick Fourier Transform: An FFT based on symmetries. *IEEE Transactions on Signal Processing*, **46**(2):335–341, February 1998.

R. Hamming. *Numerical Methods for Scientists and Engineers*. Dover Publications, Inc., New York, NY, 2nd edition, 1973.

D. C. Hanselman and B. L. Littlefield. *Mastering MATLAB*. Prentice Hall, Upper Saddle River, 2005.

M. W. Hauser. Principles of oversampling A/D conversion. *Journal of the Audio Engineering Society*, **39**(1/2):3–26, 1991.

S. Haykin. *Adaptive Filter Theory*. Prentice Hall, Upper Saddle River, New Jersey, 4th edition, 2002.

S. Haykin and B. Van Veen. *Signals and Systems*. John Wiley & Sons, New York, NY, 2nd edition, 2003.

M. Heideman, D. Johnson, and C. Burrus. Gauss and the history of the Fast Fourier Transform. *IEEE ASSP Magazine*, **1**(4):14–21, October 1984.

D. F. Hoeschele. *Analog-to-Digital and Digital-to-Analog Conversion Techniques*. Wiley, New York, 1994.

Robert Hogg and Elliot Tanis. *Probability and Statistical Inference*. Prentice Hall, Upper Saddle River, New Jersey, 7th edition, 2005.

C-C. Hsiao. Polyphase filter matrix for rational sampling rate conversions. In *IEEE ICASSP 1987*, volume **12**, 2173–2176, April 1987.

V. Ingle and J. Proakis. *Digital Signal Processing using MATLAB*. Thomson, Singapore, 2007.

F. Itakura and S. Saito. A statistical method for estimation of speech spectral density and formant frequencies. *Electronic Communications Japan*, **53**-A(1):36–43, 1971.

L. B. Jackson. On the interaction of roundoff noise and dynamic range in digital filters. *Bell System Technical Journal*, **49**:159–184, February 1970a.

L. B. Jackson. Roundoff noise analysis for fixed-point digital filters realized in cascade or parallel form. *IEEE Transactions on Audio and Electroacoustics*, **18**:107–122, June 1970b.

L. B. Jackson. *Digital Filters and Signal Processing*. Kluwer Academic Publishers, Boston, 3rd edition, 1996.

A. Jain. *Fundamentals of Digital Image Processing*. Prentice Hall, Upper Saddle River, 1989.

N. Jayant and P. Noll. *Digital Coding of Waveforms*. Prentice Hall, Upper Saddle River, 1984.

G. Jenkins and D. Watts. *Spectral Analysis and Its Applications*. Holden-Day, San Francisco, 1968.

D. Johnson and D. Dudgeon. *Array Signal Processing: Concepts and Techniques*. Prentice Hall, Upper Saddle River, New Jersey, 1993.

R. Johnson and D. Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, New Jersey, 6th edition, 2007.

S. G. Johnson and M. Frigo. A modified split-radix FFT with fewer arithmetic operations. *IEEE Transactions in Signal Processing*, **55**(1):111–119, 2007.

J. Johnston. A filter family designed for use in quadrature mirror filter banks. In *IEEE ICASSP 1980*, volume **5**, 291–294, April 1980.

J. Kaiser. Nonrecursive digital filter design using the i_0-sinh window function. In *Proc. 1974 IEEE ISCAS*, San Francisco, 1974.

D. W. Kammler. *First Course in Fourier Analysis*. Prentice Hall, Upper Saddle River, 2000.

L. Karam, J. McClellan, I. Selescick, and C. S. Burrus. Digital filtering. In V. Madisetti, editor, *The Digital Signal Processing Handbook*, volume **1**. CRC Press, 2009.

Alan H. Karp. Bit reversal on uniprocessors. *SIAM Review*, **38**(1):1–26, 1996.

S. Kay. *Fundamentals of Statistical Processing: Detection Theory*. Prentice Hall, Upper Saddle River, 1998.

W. Kester, editor. *The Data Conversion Handbook*. Newnes, Amsterdam, 2005.

R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **29**(6):1153–1160, December 1981.

S. M. Kuo and W. Gan. *Digital Signal Processors*. Prentice Hall, Upper Saddle River, 2005.

S. M. Kuo, B. H. Lee, and W. Tian. *Real-Time Digital Signal Processing*. Wiley, New York, NY, 2nd edition, 2006.

T. Laakso, V. Valimaki, M. Karjalainen, and U. Laine. Splitting the unit delay. *IEEE Signal Processing Magazine*, pages 30–60, January 1996.

H. Lam. *Analog and Digital Filters: Design and Realization*. Prentice Hall, Englewood Cliffs, New Jersey, 1979.

B. P. Lathi. *Linear Systems and Signals*. Oxford University Press, New York, 2005.

A. Leon-Garcia. *Probability, Statistics, and Random Processes For Electrical Engineering*. Prentice Hall, Upper Saddle River, New Jersey, Third edition, 2008.

D. Linden. A discussion of sampling theorems. *Proceedings of the IRE*, **47**(7):1219–1226, July 1959.

J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, **63**(4):561–580, April 1975.

J. Makhoul. Stable and efficient lattice methods for linear prediction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **25**:423–428, October 1977.

D. Manolakis, V. Ingle, and S. Kogon. *Statistical and Adaptive Signal Processing*. Artech House, Boston, 2005.

D. Marco and D. L. Neuhoff. The validity of the additive noise model for uniform scalar quantizers. *IEEE Transactions on Information Theory*, **51**(5):1739–1755, May 2005.

J. McClellan and T. Parks. A united approach to the design of optimum FIR linear-phase digital filters. *IEEE Transactions on Circuit Theory*, **20**(6):697–701, November 1973.

J. McClellan, T. Parks, and L. Rabiner. A computer program for designing optimum FIR linear phase digital filters. *IEEE Transactions on Audio and Electroacoustics*, **21**(6):506–526, December 1973.

J. H. McClellan and T. W. Parks. A personal history of the Parks–McClellan algorithm. *IEEE Signal Processing Magazine*, **22**(2):82–86, March 2005.

R. Meyer and K. Schwarz. FFT implementation on DSP-chips: theory and practice. In *IEEE ICASSP 1990*, volume **3**, 1503–1506, April 1990.

F. Mintzer. On half-band, third-band, and Nth-band FIR filters and their design. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **30**(5):734–738, October 1982.

F. Mintzer. Filters for distortion-free two-band multirate filter banks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **33**(3):626–630, June 1985.

D. Mitchell and A. Netravali. Reconstruction filters in computer graphics. *Computer Graphics*, **22**(4):221–228, August 1988.

S. K. Mitra. *Digital Signal Processing*. McGraw-Hill, New York, NY, 3rd edition, 2006.

F. R. Moore. *Elements of Computer Music*. Prentice Hall, Upper Saddle River, New Jersey, 1990.

J. A. Moorer. About this reverberation business. *Computer Music Journal*, **3**(2):13–28, 1979.

Y. Neuvo, D. Cheng-Yu, and S. Mitra. Interpolated finite impulse response filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **32**(3):563–570, June 1984.

G. Oetken, T. Parks, and H. Schussler. New results in the design of digital interpolators. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **23**(3):301–309, March 1975.

A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2010.

A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals and Systems*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 1997.

A. V. Oppenheim and C. J. Weinstein. Effects of finite register length in digital filtering and the Fast Fourier Transform. *Proceedings of the IEEE*, **60**(8):957–976, August 1972.

R. Pachon and L. Trefethen. Barycentric-Remez algorithms for best polynomial approximation in the chebfun system. *BIT Numer Math*, **49**:721–741, 2009.

A. Papoulis. On the approximation problem in filter design. *IRE National Convention Record*, **5**:175–185, 1957.

A. Papoulis. *The Fourier Integral and Its Applications*. McGraw-Hill Companies, New York, 1962.

A. Papoulis. *Signal Analysis*. McGraw-Hill, New York, NY, 1977.

A. Papoulis and S. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, New York, 2002.

T. Parks and J. McClellan. Chebyshev approximation for nonrecursive digital filters with linear phase. *IEEE Transactions on Circuit Theory*, **19**(2):189–194, March 1972.

T. W. Parks and C. Burrus. *Digital Filter Design*. Wiley, New York, NY, 1987.

D. Percival and A. Walden. *Spectral Analysis for Physical Applications*. Cambridge University Press, Cambridge, 1993.

B. Porat. *A Course in Digital Signal Processing*. Wiley, New York, 1997.

M. Powell. *Approximation Theory and Methods*. Cambridge University Press, Cambridge, 1981.

W. K. Pratt. *Digital Image Processing*. Wiley, New York, NY, 4th edition, 2007.

W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes*. Cambridge University Press, London, UK, 3rd edition, 2007.

M. B. Priestley. *Spectral Analysis and Time Series*. Academic Press, Amsterdam, 1981.

J. G. Proakis and D. G. Manolakis. *Digital Signal Processing*. Prentice Hall, Upper Saddle River, NJ, 4th edition, 2007.

L. Rabiner, J. Kaiser, and R. Schafer. Some considerations in the design of multiband finite-impulse-response digital filters. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **22**(6):462–472, December 1974a.

L. R. Rabiner and B. Gold. *Theory and Application of Digital Signal Processing*. Prentice Hall, Upper Saddle River, 1975.

L. R. Rabiner, B. Gold, and C. A. McGonegal. An approach to the approximation problem for nonrecursive digital filters. *IEEE Transactions on Audio and Electroacoustics*, **18**(2):83–97, June 1970.

L. R. Rabiner, J. F. Kaiser, O. Hermann, and M. T. Dolan. Some comparisons between FIR and IIR digital filters. *Bell System Technical Journal*, **53**:305–331, 1974b.

L. R. Rabiner, J. H. McClellan, and T. W. Parks. FIR digital filter design techniques using weighted Chebyshev approximation. *Proceedings of the IEEE*, **63**(4):595–610, April 1975.

L. R. Rabiner and R. W. Schafer. *Theory and Application of Digital Speech Processing*. Prentice Hall, Englewood Cliffs, New Jersey, 2010.

L. R. Rabiner, R. W. Schafer, and C. M. Rader. The Chirp z-Transform algorithm. *IEEE Transactions on Audio and Electroacoustics*, **17**:86–92, June 1969.

C. M. Rader. Discrete Fourier Transforms when the number of data samples is prime. *IEEE Proceedings*, **56**:1107–1108, June 1968.

M. Rice. *Digital Communications: A Discrete-Time Approach*. Prentice Hall, Upper Saddle River, New Jersey, 2009.

M. A. Richards. *Fundamentals of Radar Signal Processing*. McGraw-Hill, New York, NY, 2005.

J. Rodriguez. An improved FFT digit-reversal algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**(8):1298–1300, August 1989.

S. Ross. *Introduction to Probability and Statistics for Engineers and Scientists*. Academic Press, Amsterdam, 3rd edition, 2004.

T. Saramaki. Finite impulse response filter design. In S. K. Mitra and J. F. Kaiser, editors, *Handbook for Digital Signal Processing*. Wiley, New York, 1993.

R. W. Schafer and L. R. Rabiner. A digital signal processing approach to interpolation. *Proceedings of the IEEE*, **61**(6):692–702, June 1973.

J. C. Schatzman. Accuracy of the Discrete Fourier Transform and the Fast Fourier Transform. *SIAM Journal on Scientific Computing*, **17**(5):1150–1166, 1996.

R. Schreier and G. Temes. *Understanding Delta-Sigma Data Converters*. Wiley-IEEE Press, New York, NY, 2005.

M. R. Schroeder and B. F. Logan. "Colorless" artificial reverberation. *IEEE Transactions on Audio*, **9**(6):209–214, November 1961.

I. W. Selesnick, M. Lang, and C. S. Burrus. Constrained least square design of FIR filters without specified transition bands. *IEEE Transactions on Signal Processing*, **44**(8):1879–1892, August 1996.

C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, **37**(1):10–21, January 1949.

D. J. Shpak and A. Antoniou. A generalized Remez method for the design of FIR digital filters. *IEEE Transactions on Circuits and Systems*, **37**(2):161–174, February 1990.

R. Singleton. An algorithm for computing the mixed radix Fast Fourier Transform. *IEEE Transactions on Audio and Electroacoustics*, **17**(2):93–103, June 1969.

D. Slepian. Prolate spheroidal wave functions, Fourier analysis, and uncertainty. V: The discrete case. *Bell System Technical Journal*, **57**(5):1371–1430, May-June 1978.

M. Smith and T. Barnwell. A procedure for designing perfect reconstruction filter-banks for tree structured subband coders. In *IEEE ICASSP 1984*, pp. 27.1.1.–4, San Diego, March 1984.

H. Sorensen, M. Heideman, and C. Burrus. On computing the Split-radix FFT. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **34**(1):152–156, February 1986.

A. Sripad and D. Snyder. A necessary and sufficient condition for quantization errors to be uniform and white. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **25**(5):442–448, October 1977.

H. Stark and J. Woods. *Probability and Random Processes with Applications to Signal Processing*. Prentice Hall, Upper Saddle River, New Jersey, 2002.

K. Steiglitz, T. W. Parks, and J. F. Kaiser. METEOR: A constraint-based FIR filter design program. *IEEE Transactions on Signal Processing*, **40**(8):1901–1909, August 1992.

P. Stoica and R. Moses. *Spectral Analysis of Signals*. Prentice Hall, Upper Saddle River, 2005.

G. Strang. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, 1986.

G. Strang and T. Nguyen. *Wavelets and Filter Banks: Theory and Design*. Wellesley-Cambridge Press, Wellesley, MA, 1996.

A. Stuart and J. Keith Ord. *Keldall's Advanced Theory of Statistics*. Oxford University Press, New York, 1991.

G. Turin. An introduction to matched filters. *IEEE Transactions on Information Theory*, **3**:311–329, June 1960.

M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 22–38, November 1999.

P. P. Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proceedings of the IEEE*, **78**(1):56–93, January 1990.

P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, Englewood Cliffs, New Jersey, 1993.

P. P. Vaidyanathan and T. Q. Nguyen. A trick for the design of FIR half-band filters. *IEEE Transactions on Circuits and Systems*, **34**(3):297–300, March 1987.

C. F. Van Loan. *Computational Frameworks for the Fast Fourier Transform*. SIAM, Philadelphia, PA, USA, 1992.

C. F. Van Loan. *Introduction to Scientific Computing*. Prentice Hall, Upper Saddle River, NJ, 2nd edition, 2000.

R. G. Vaughan, N. L. Scott, and D. R. White. The theory of bandpass sampling. *IEEE Transactions on Signal Processing*, **39**(9):1973–1984, September 1991.

M. Vetterli and C. Herley. Wavelets and filter banks: theory and design. *IEEE Transactions on Signal Processing*, **40**(9):2207–2232, September 1992.

J. S. Walker. *Fourier Analysis*. Oxford University Press, New York, 1988.

Z. Wang, J. Soltis, and W. Miller. Improved approach to interpolation using the FFT. *Electronics Letters*, **28**(25):2320–2322, December 1992.

L. Weinberg. *Network Analysis and Synthesis*. R. E. Krieger Publishing Co., Huntington, NY, 1975.

C. Weinstein. Roundoff noise in floating point Fast Fourier Transform computation. *IEEE Transactions on Audio and Electroacoustics*, **17**(3):209–215, September 1969.

P. Welch. A fixed-point Fast Fourier Transform error analysis. *IEEE Transactions on Audio and Electroacoustics*, **17**(2):151–157, June 1969.

T. B. Welch, C. H. G. Wright, and M. G. Morrow. *Real-Time Digital Signal Processing*. Taylor and Francis, Boca Raton, USA, 2006.

K. Williston, editor. *Digital Signal Processing: World Class Design*. Newnes, Amsterdam, 2009.

S. Winograd. On computing the Discrete Fourier Transform. *Mathematics of Computation*, **32**(141):175–199, 1978.

J. Woods. *Multidimensional Signal, Image, and Video Processing and Coding*. Academic Press, San Diego, 2006.

R. Woods, J. McAllister, G. Lightbody, and Y. Yi. *FPGA-based Implementation of Signal Processing Systems*. Wiley, New York, NY, 2008.

R. Yavne. An economical method for calculating the Discrete Fourier Transform. In *Proc. AFIPS Fall Joint Computer Conf.*, volume **33**, 115–125, 1968.

D. Zill and P. Shanahan. *A First Course in Complex Analysis with Applications*. Jones and Bartlett Publishers, LLC, Sudbury, MA USA, 2nd edition, 2009.

U. Zölder. *Digital Audio Signal Processing*. Wiley, New York, NY, 2nd edition, 2008.

A. I. Zverev. *Handbook of Filter Synthesis*. Wiley, New York, 1967.

INDEX