Chap8
Computation of the Discrete Fourier Transform

Chao-Tsung Huang

National Tsing Hua University
Department of Electrical Engineering

Chap 8 Computation of DFT

- 8.1 Direct computation of the DFT
- 8.2 The FFT idea using a matrix approach
- 8.3 Decimation-in-time FFT algorithms
- 8.4 Decimation-in-frequency FFT algorithms
- 8.5 Generalizations and additional FFT algorithms
- 8.6 Practical considerations

Direct Computation of DFT

DFT

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, 1, \ldots, N-1$$

N^2 multiplications + $N(N-1)$ additions
$\Rightarrow O(N^2)$

IDFT

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn}. \quad n = 0, 1, \ldots, N-1$$

$$x[n] = \frac{1}{N} \left[\sum_{k=0}^{N-1} X^*[k] W_N^{kn} \right]^*. \quad n = 0, 1, \ldots, N-1$$

Redundancy in DFT coefficient

Example (N=8)

$$
k \quad
\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix}
=
\begin{array}{c}
\begin{array}{cccccccc} & & \text{Even n} & & & \text{Odd n} & & \end{array} \\
\begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & W_8 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\
1 & W_8^2 & W_8^4 & W_8^6 & 1 & W_8^2 & W_8^4 & W_8^6 \\
1 & W_8^3 & W_8^6 & W_8 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\
1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 \\
1 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8 & W_8^6 & W_8^3 \\
1 & W_8^6 & W_8^4 & W_8^2 & 1 & W_8^6 & W_8^4 & W_8^2 \\
1 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8
\end{bmatrix} \\
n
\end{array}
\begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}
$$

$$W_N^{(k+N/2)n} = (-1)^n W_N^{kn}$$

FFT idea (reorder in time)

$$
\begin{bmatrix} X[0] \\ X[1] \\ X[2] \\ X[3] \\ X[4] \\ X[5] \\ X[6] \\ X[7] \end{bmatrix}
=
\left[\begin{array}{cccc|cccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & W_8^2 & W_8^4 & W_8^6 & W_8 & W_8^3 & W_8^5 & W_8^7 \\
1 & W_8^4 & 1 & W_8^4 & W_8^2 & W_8^6 & W_8^2 & W_8^6 \\
1 & W_8^6 & W_8^4 & W_8^2 & W_8^3 & W_8 & W_8^7 & W_8^5 \\
\hline
1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\
1 & W_8^2 & W_8^4 & W_8^6 & -W_8 & -W_8^3 & -W_8^5 & -W_8^7 \\
1 & W_8^4 & 1 & W_8^4 & -W_8^2 & -W_8^6 & -W_8^2 & -W_8^6 \\
1 & W_8^6 & W_8^4 & W_8^2 & -W_8^3 & -W_8 & -W_8^7 & -W_8^5
\end{array}\right]
\begin{bmatrix} x[0] \\ x[2] \\ x[4] \\ x[6] \\ x[1] \\ x[3] \\ x[5] \\ x[7] \end{bmatrix}
$$

$$
\left[\begin{array}{c} \boldsymbol{X}_{\mathrm{T}} \\ \hline \boldsymbol{X}_{\mathrm{B}} \end{array}\right]
=
\left[\begin{array}{c|c} \boldsymbol{W}_4 & \boldsymbol{D}_8 \boldsymbol{W}_4 \\ \hline \boldsymbol{W}_4 & -\boldsymbol{D}_8 \boldsymbol{W}_4 \end{array}\right]
\left[\begin{array}{c} \boldsymbol{x}_{\mathrm{E}} \\ \hline \boldsymbol{x}_{\mathrm{O}} \end{array}\right]
$$

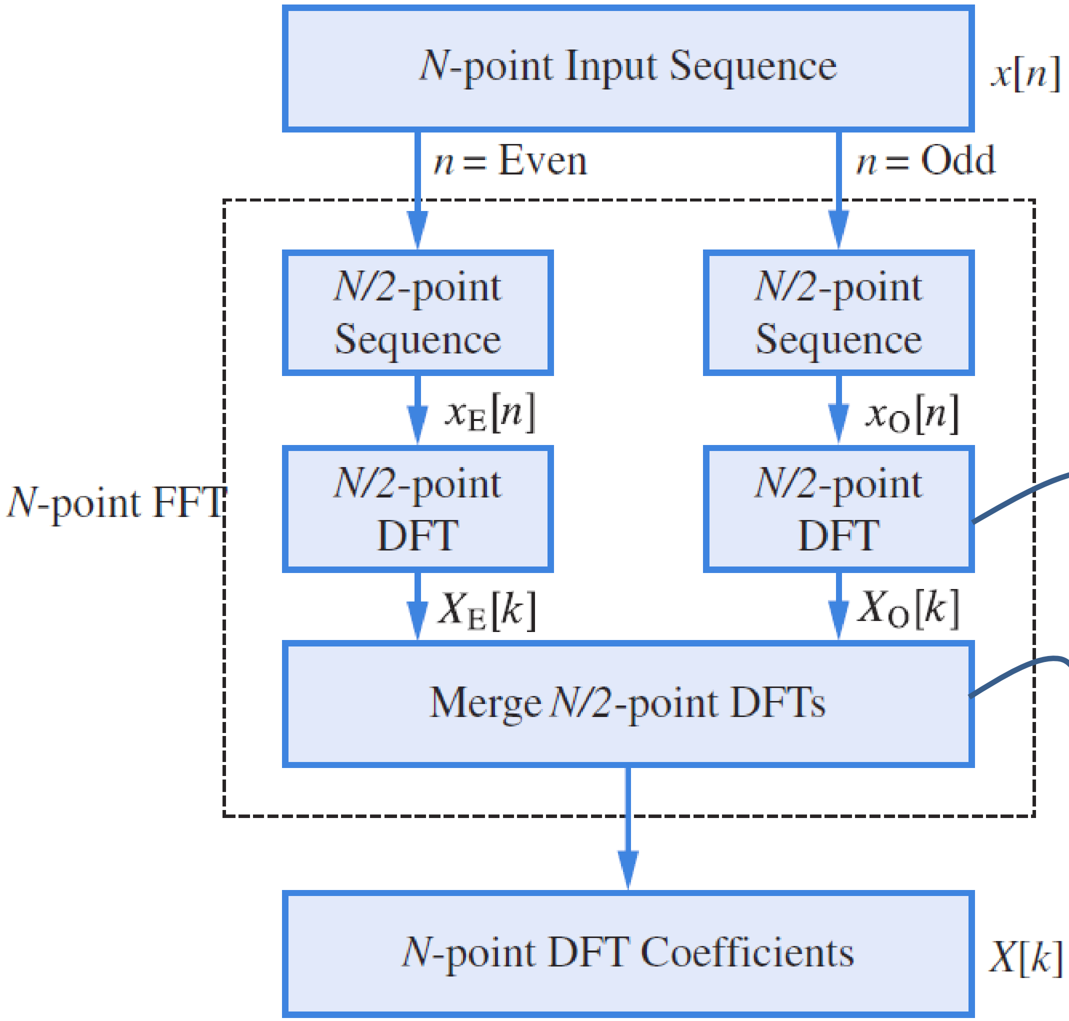
4-point DFT

$$
W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & W_4 & W_4^2 & W_4^3 \\ 1 & W_4^2 & 1 & W_4^2 \\ 1 & W_4^3 & W_4^2 & W_4 \end{bmatrix}
$$

Diagonal matrix

$$
D_8 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & W_8 & 0 & 0 \\ 0 & 0 & W_8^2 & 0 \\ 0 & 0 & 0 & W_8^3 \end{bmatrix}
$$

Divide and conquer



Complexity for $N = 2^{v}$:
$\frac{N}{2}\log N$ multiplications +
$N\log N$ additions
$\Rightarrow O(N\log N)$

$$\mathbf{X}_{\mathrm{E}} = \mathbf{W}_{\frac{N}{2}}\mathbf{x}_{\mathrm{E}}$$

$$\mathbf{X}_{\mathrm{O}} = \mathbf{W}_{\frac{N}{2}}\mathbf{x}_{\mathrm{O}}$$

$$\mathbf{X}_{\mathrm{T}} = \mathbf{X}_{\mathrm{E}} + \mathbf{D}_{N}\mathbf{X}_{\mathrm{O}}$$

$$\mathbf{X}_{\mathrm{B}} = \mathbf{X}_{\mathrm{E}} - \mathbf{D}_{N}\mathbf{X}_{\mathrm{O}}$$

FFT idea (reorder in frequency)

$$\begin{bmatrix} X[0] \\ X[2] \\ X[4] \\ X[6] \\ X[1] \\ X[3] \\ X[5] \\ X[7] \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W_8^2 & W_8^4 & W_8^6 & 1 & W_8^2 & W_8^4 & W_8^6 \\ 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 & 1 & W_8^4 \\ 1 & W_8^6 & W_8^4 & W_8^2 & 1 & W_8^6 & W_8^4 & W_8^2 \\ 1 & W_8 & W_8^2 & W_8^3 & W_8^4 & W_8^5 & W_8^6 & W_8^7 \\ 1 & W_8^3 & W_8^6 & W_8 & W_8^4 & W_8^7 & W_8^2 & W_8^5 \\ 1 & W_8^5 & W_8^2 & W_8^7 & W_8^4 & W_8 & W_8^6 & W_8^3 \\ 1 & W_8^7 & W_8^6 & W_8^5 & W_8^4 & W_8^3 & W_8^2 & W_8 \end{bmatrix} \begin{bmatrix} x[0] \\ x[1] \\ x[2] \\ x[3] \\ x[4] \\ x[5] \\ x[6] \\ x[7] \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{X}_{\mathrm{E}} \\ \hline \boldsymbol{X}_{\mathrm{O}} \end{bmatrix} = \left[\begin{array}{c|c} \boldsymbol{W}_4 & \boldsymbol{W}_4 \\ \hline \boldsymbol{W}_4\boldsymbol{D}_8 & -\boldsymbol{W}_4\boldsymbol{D}_8 \end{array}\right] \begin{bmatrix} \boldsymbol{x}_{\mathrm{T}} \\ \hline \boldsymbol{x}_{\mathrm{B}} \end{bmatrix}$$

$$\boldsymbol{X}_{\mathrm{E}} = \boldsymbol{W}_{\frac{N}{2}}\, \boldsymbol{v}, \qquad \boldsymbol{v} \triangleq \boldsymbol{x}_{\mathrm{T}} + \boldsymbol{x}_{\mathrm{B}},$$

$$\boldsymbol{X}_{\mathrm{O}} = \boldsymbol{W}_{\frac{N}{2}}\, \boldsymbol{z}, \qquad \boldsymbol{z} \triangleq \boldsymbol{D}(\boldsymbol{x}_{\mathrm{T}} - \boldsymbol{x}_{\mathrm{B}}).$$

Decimation-in-time FFT

Divide

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \quad k = 0, 1, \ldots, N-1$$

Decimation in time

$$= \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, W_N^{k(2m)} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, W_N^{k(2m+1)}$$

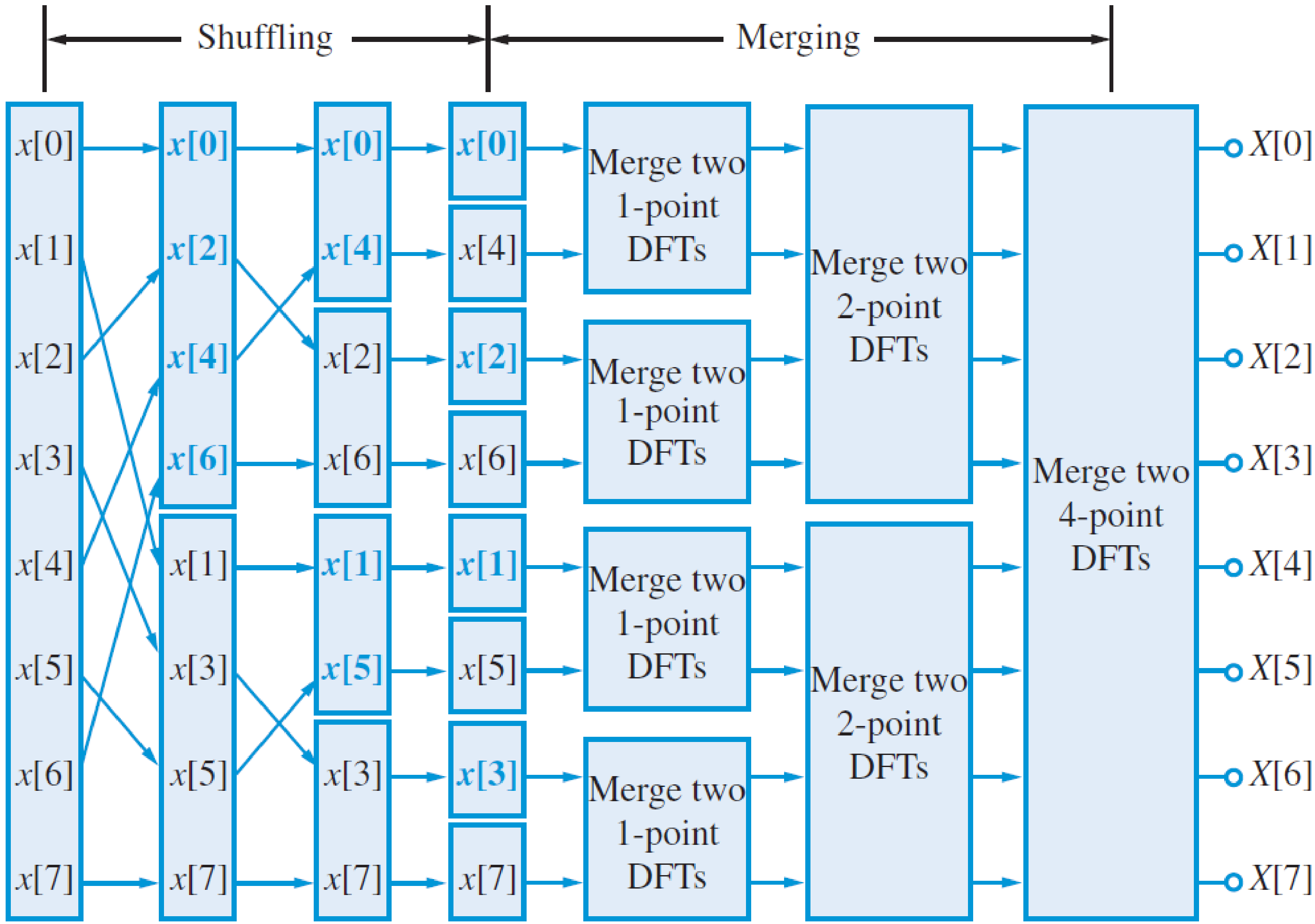
$$= \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x[2m]\, W_N^{k(2m)}}_{\substack{\text{N/2-point DFT} \\ A[k]}} + W_N^k \underbrace{\sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, W_N^{k(2m)}}_{\substack{\text{N/2-point DFT} \\ B[k]}}$$

Merge

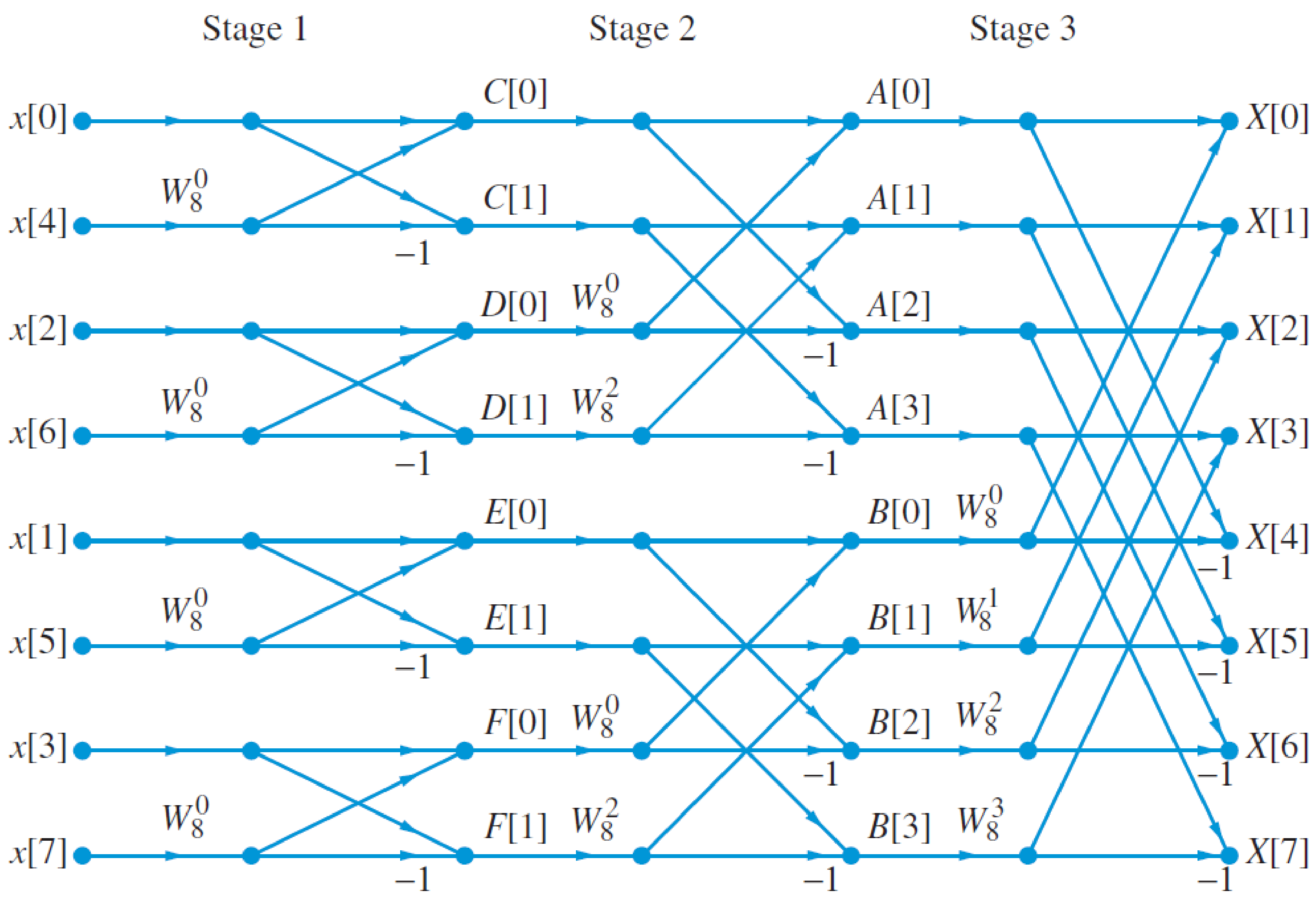
$$X[k] = A[k] + W_N^k B[k], \quad k = 0, 1, \ldots, \frac{N}{2} - 1$$

$$X\left[k + \frac{N}{2}\right] = A[k] - W_N^k B[k]. \quad k = 0, 1, \ldots, \frac{N}{2} - 1$$

Shuffle and merge



Butterfly computation flow



Bit-reverse order

In-place computation

Butterfly computation flow

Bit-reverse order

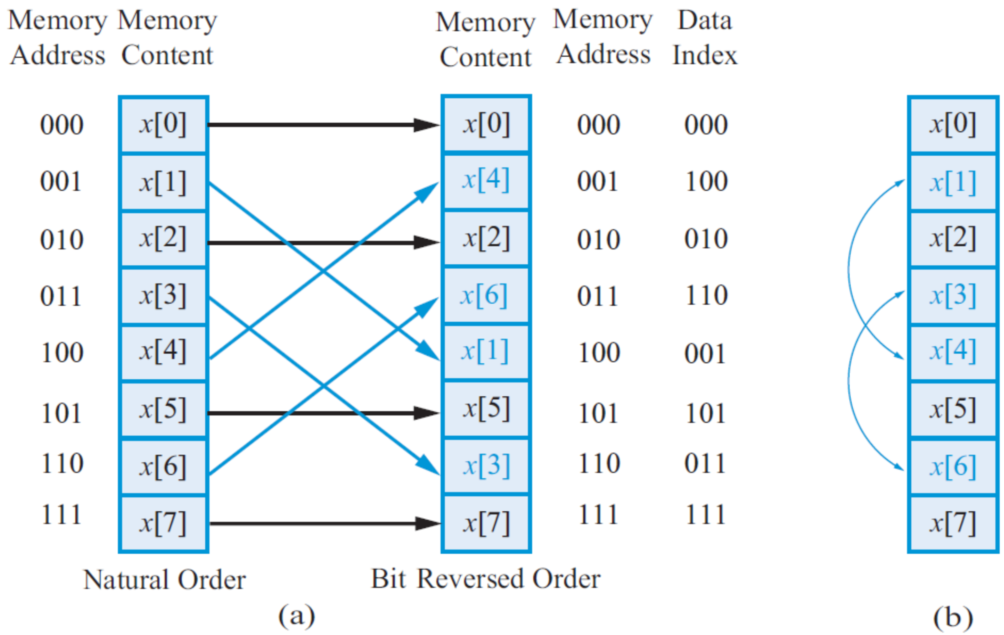
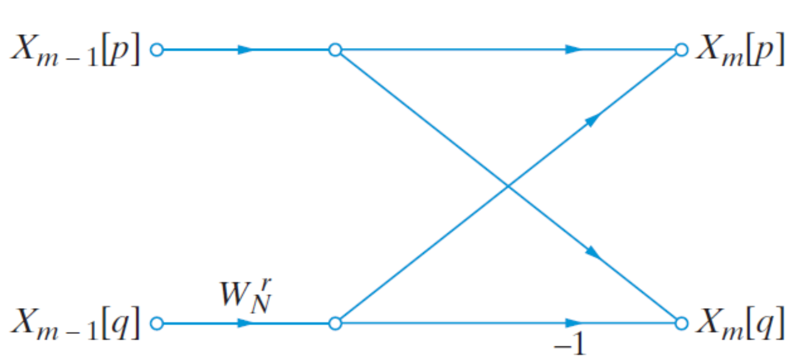


Figure 8.7 Shuffling a sequence with $N = 8$ samples by bit-reversal indexing: (a) shuffling using two arrays, and (b) in-place shuffling.

In-place computation



Decimation-in-frequency FFT

Divide

$$X[2k] = \sum_{n=0}^{\frac{N}{2}-1} x[n] W_{\frac{N}{2}}^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left[n + \frac{N}{2}\right] W_{\frac{N}{2}}^{k\left(n+\frac{N}{2}\right)}$$

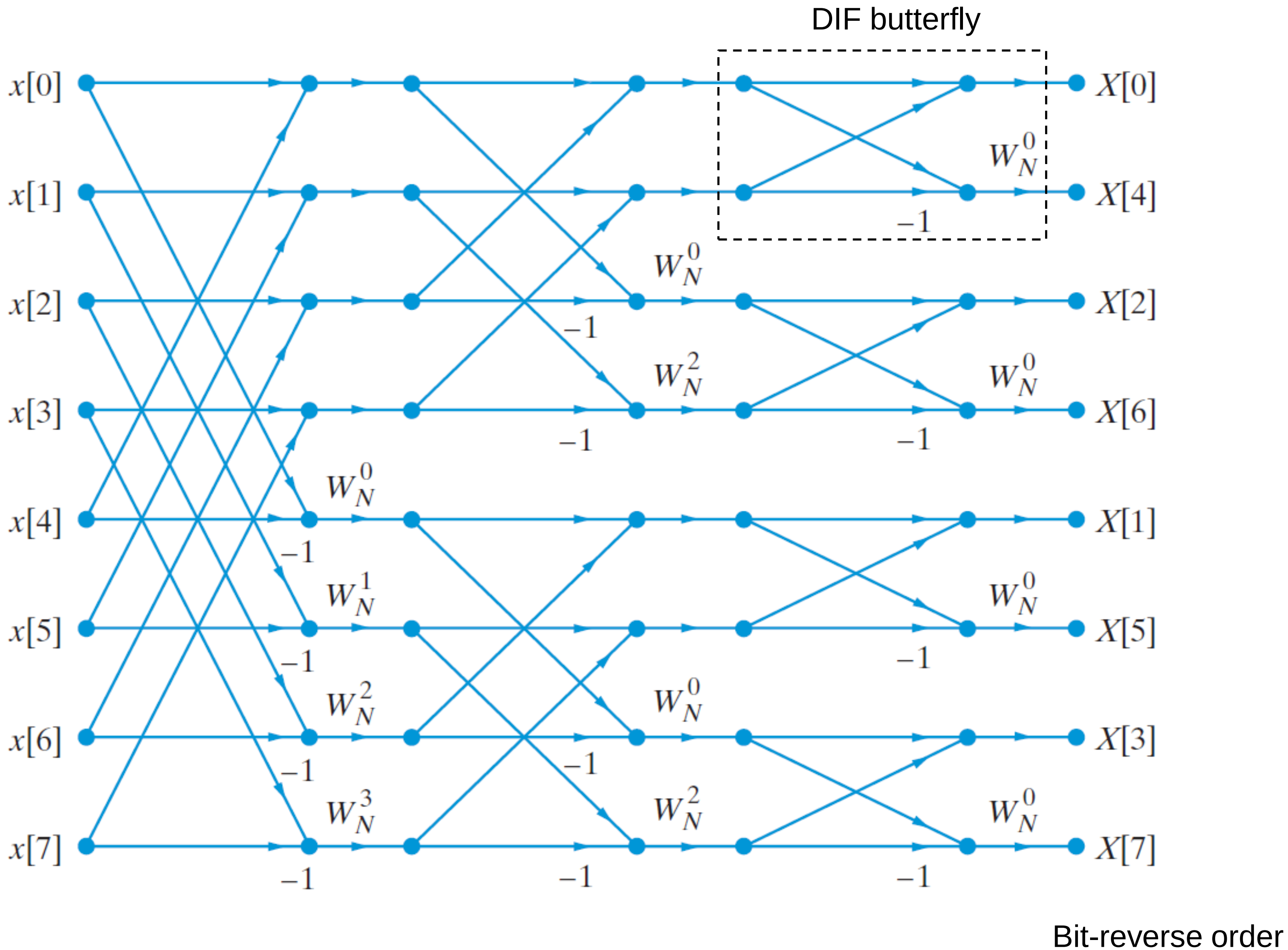
$$= \sum_{n=0}^{\frac{N}{2}-1} x[n] W_{\frac{N}{2}}^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x\left[n + \frac{N}{2}\right] W_{\frac{N}{2}}^{kn}$$

$$= \underbrace{\sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + x\left[n + \frac{N}{2}\right]\right) W_{\frac{N}{2}}^{kn}}_{\text{Merge + N/2-point DFT } A[k]}. \quad k = 0, 1, \ldots, \frac{N}{2} - 1$$

Decimation in frequency

$$X[2k+1] = \underbrace{\sum_{n=0}^{\frac{N}{2}-1} \left[\left(x[n] - x\left[n + \frac{N}{2}\right]\right) W_N^n\right] W_{\frac{N}{2}}^{kn}}_{\text{Merge + N/2-point DFT } B[k]}$$

Butterfly computation flow



Cooley-Tuckey algorithm (general approach)

Setting

$$N = N_1 N_2$$ Composite number

$$n = N_2 n_1 + n_2, \quad n_1 = 0, 1, \ldots, N_1 - 1, \quad n_2 = 0, 1, \ldots, N_2 - 1$$

Divide x[n] in N_2 sub-sequences of length N_1

$$k = k_1 + N_1 k_2. \quad k_1 = 0, 1, \ldots, N_1 - 1, \quad k_2 = 0, 1, \ldots, N_2 - 1$$

Divide X[k] in N_1 sub-sequences of length N_2

$$\begin{aligned} nk &= (N_2 n_1 + n_2)(k_1 + N_1 k_2) \\ &= N n_1 k_2 + N_1 n_2 k_2 + N_2 n_1 k_1 + n_2 k_1 \end{aligned}$$

$$W_N^N = 1, \; W_N^{N_1} = W_{N_2}, \; W_N^{N_2} = W_{N_1}$$

Cooley-Tuckey algorithm (general approach)

Decomposition

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_N^{(N_2 n_1 + n_2)(k_1 + N_1 k_2)}$$

$$\Downarrow$$

Overall complexity: $O(N(N_1 + N_2 + 1))$

$$X[k_1 + N_1 k_2] = \underbrace{\sum_{n_2=0}^{N_2-1} \left[\underbrace{\left(\sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{k_1 n_1} \right)}_{X_{n_2}[k_1]} W_N^{k_1 n_2} \right] W_{N_2}^{k_2 n_2}}$$

Element-wise multiplication (complexity: $O(N)$)

N_2-times N_1-point DFT (complexity: $O(NN_1)$) $X_{n_2}[k_1]$

N_1-times N_2-point DFT (complexity: $O(NN_2)$)

Cooley-Tuckey algorithm (general approach)

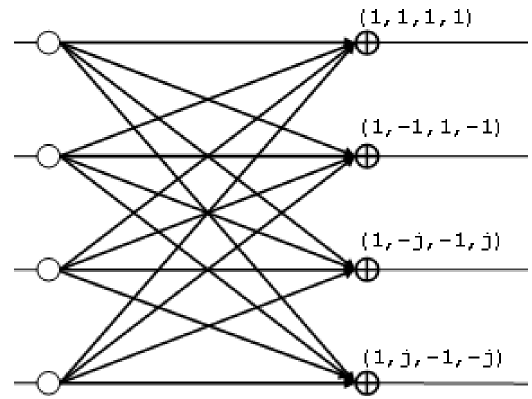
Procedure

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.

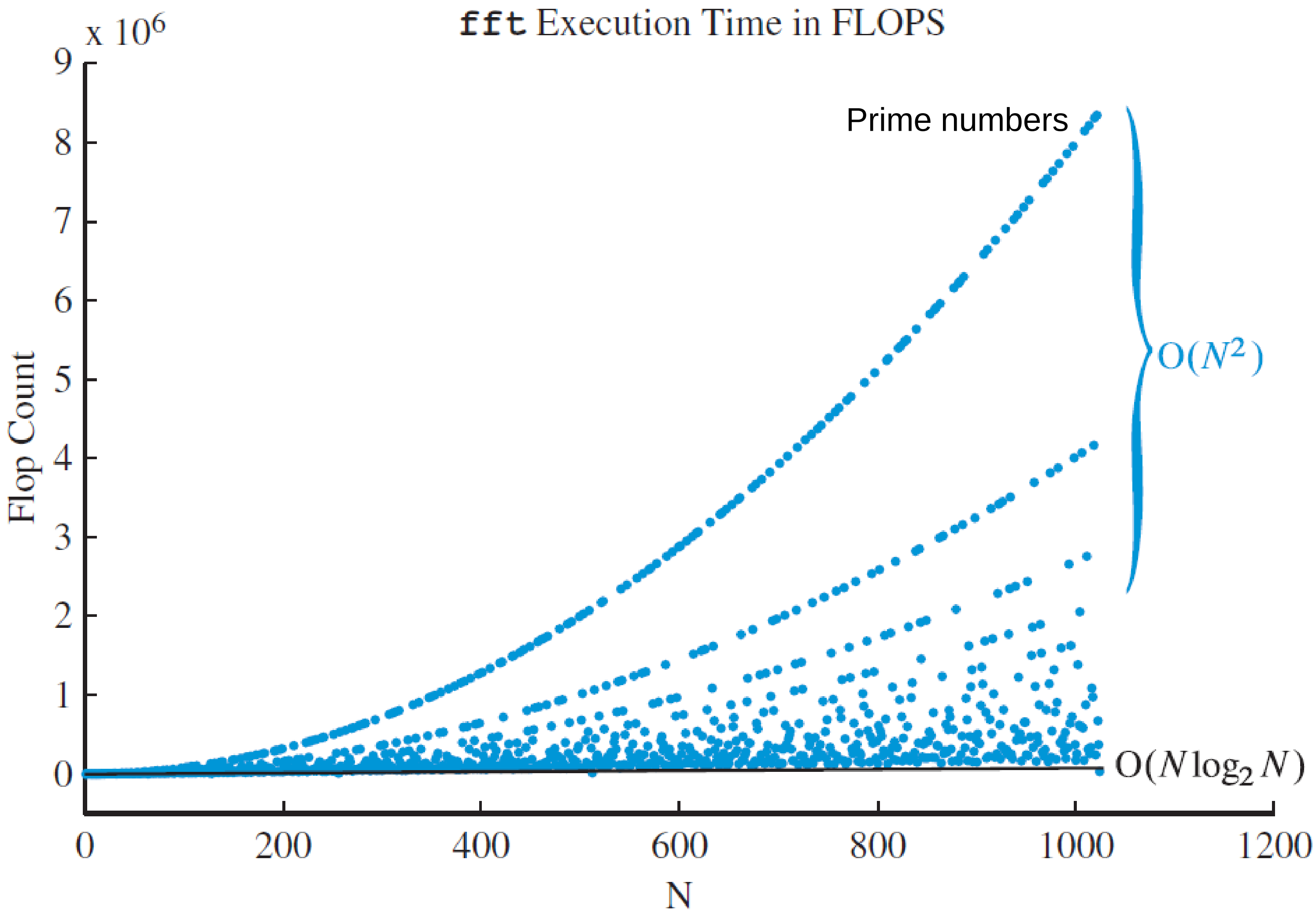
Special cases

- Decimation-in-time FFT: $N_1=N/2$, $N_2=2$
- Decimation-in-frequency FFT: $N_1=2$, $N_2=N/2$
- Radix-2 FFT: $N=2^v$
- Radix-4 FFT: $N=4^v$
  - Half as many stages as radix-2
  - 4-point DFT still doesn't require multiplications
  - Half as many multiplications as radix-2

$$\begin{bmatrix} W^0 & W^0 & W^0 & W^0 \\ W^0 & W^1 & W^2 & W^3 \\ W^0 & W^2 & W^4 & W^6 \\ W^0 & W^3 & W^6 & W^9 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix}$$



Computation time

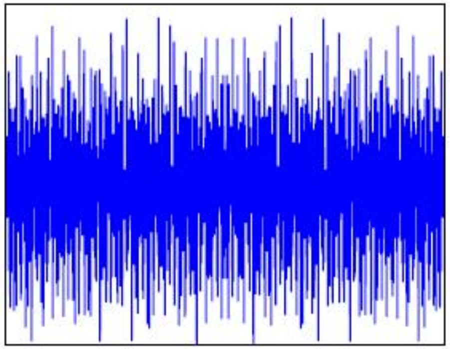


Appendix
Sparse Fast Fourier Transform

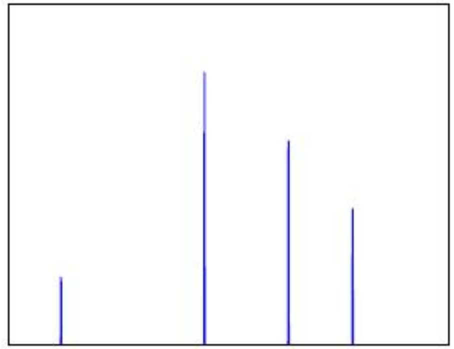
"100 Top Stories of 2013: 34. Better Math Makes Faster Data Networks", *Discover Magazine*, 2013.

https://groups.csail.mit.edu/netmit/sFFT/index.html
https://en.wikipedia.org/wiki/Sparse_Fourier_transform

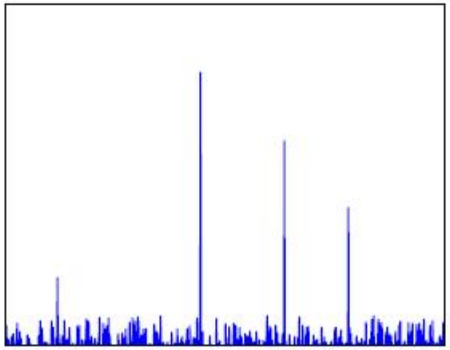
Frequency sparsity in several applications



Time Signal



Frequency
(Exactly sparse)

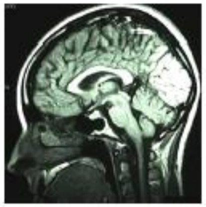


Frequency
(Approximately sparse)

Sparsity is common:

Audio

Video

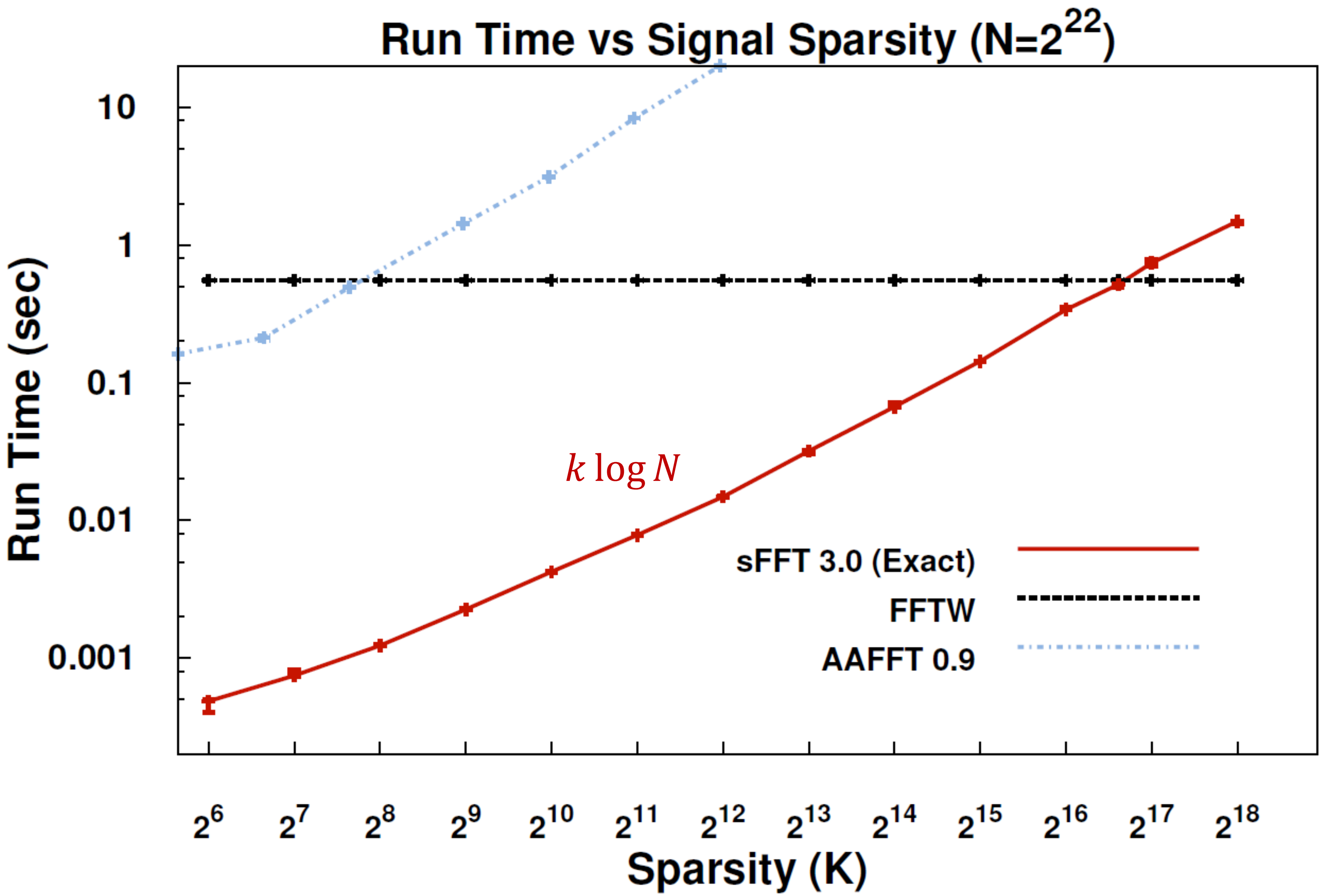


Medical Imaging

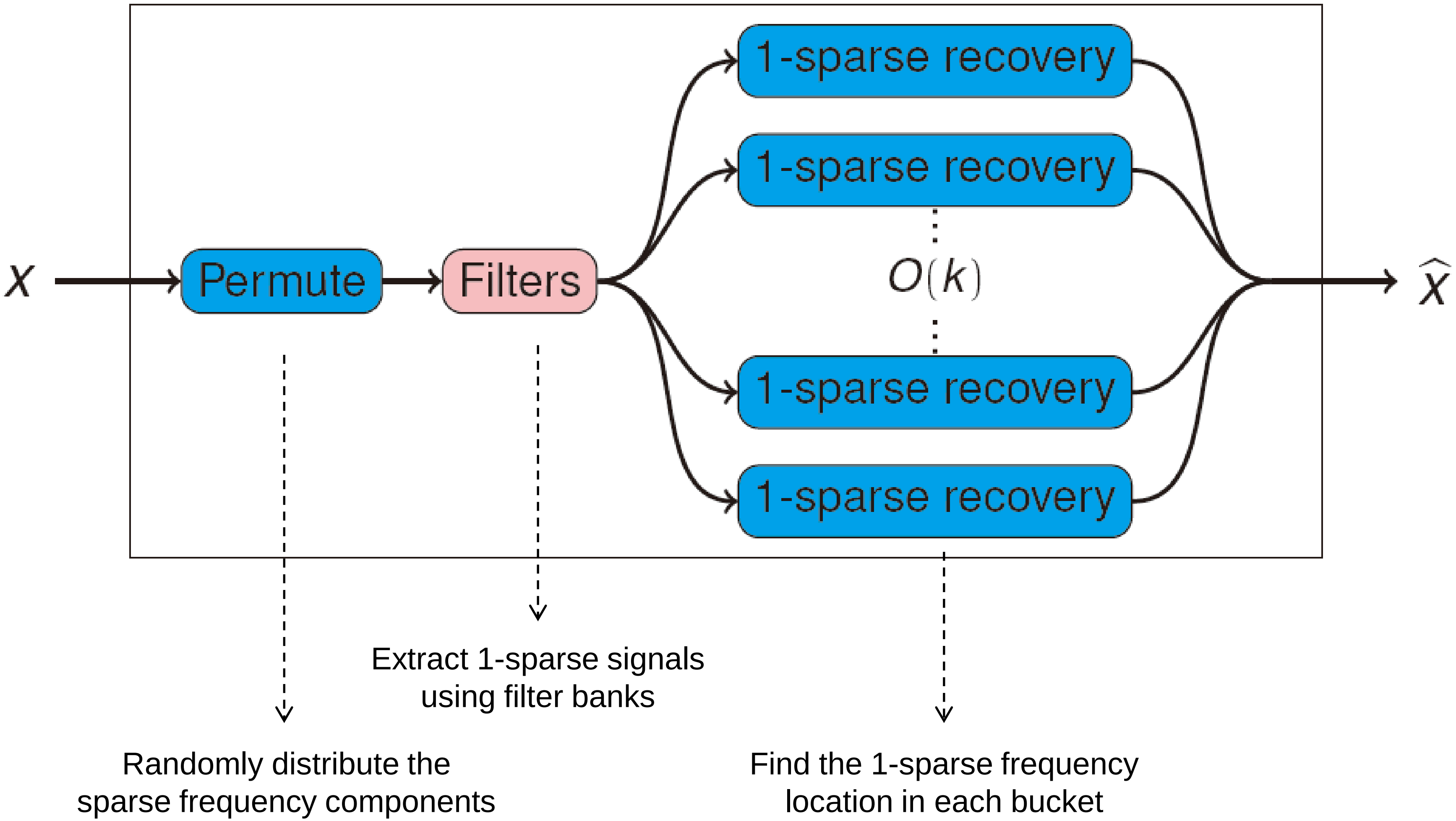
Radar

GPS

Oil Exploration


Run Time vs Signal Sparsity (N=2^22)
Run Time (sec)
10
1
0.1
0.01
0.001
$k \log N$
sFFT 3.0 (Exact)
FFTW
AAFFT 0.9
2^6 2^7 2^8 2^9 2^{10} 2^{11} 2^{12} 2^{13} 2^{14} 2^{15} 2^{16} 2^{17} 2^{18}
Sparsity (K)

Algorithm flow



Quick view of sFFT

Random Permutation

(c, d are random)

$$X[k] = \sum_{n} x[n] e^{-j2\pi \frac{k}{N} n}$$

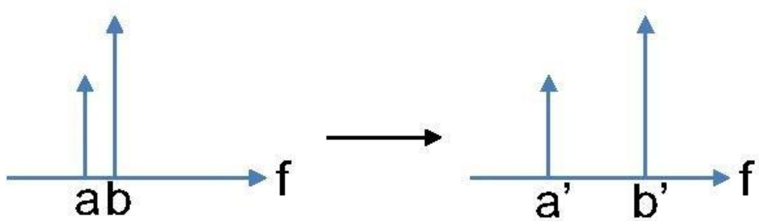
$$\downarrow$$

$$x[n] \longrightarrow x'[n'] e^{-j2\pi \frac{d}{N} n'}$$

$$x'[n'] = x[n], n = cn', c \bmod N \equiv 1$$

$$X[k] = \sum_{n'} x'[n'] e^{-j2\pi \frac{d}{N} n'} e^{-j2\pi \frac{k}{N} cn'} = \sum_{n'} x'[n'] e^{-j2\pi \frac{ck+d}{N} n'} = X'[k']$$

$$k' = ck + d$$



Spread all frequency components as a uniform distribution

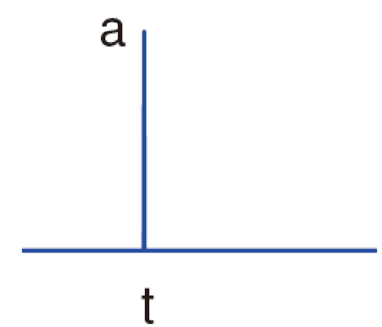
Quick view of sFFT

1-sparse recovery

Lemma

We can compute a 1-sparse $\hat{x}$ in $O(1)$ time.

$\hat{x}$:



$$\hat{x}_i = \begin{cases} a & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

- Then $x = (a, a\omega^t, a\omega^{2t}, a\omega^{3t}, \ldots, a\omega^{(n-1)t})$.

$$x_0 = a \qquad\qquad x_1 = a\omega^t$$

- $x_1 / x_0 = \omega^t \implies t.$ ■

More time shifts can be tested to increase SNR.