# National Tsing Hua University

## Department of Electrical Engineering

## EE3660 Intro. to Digital Signal Processing, Spring 2020

# Homework Assignment #2: Chap2-4 Answer

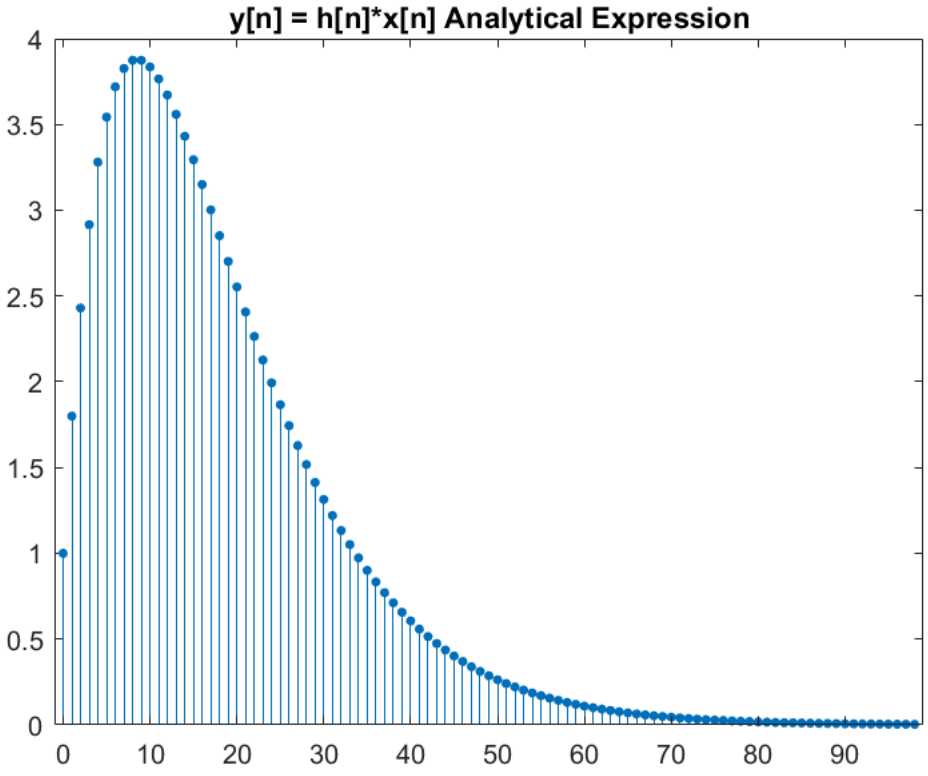
## Problem 1.

(a) $x[n] = h[n] = (0.9)^n u[n]$, $X(z) = H(z) = \dfrac{1}{1-0.9z^{-1}}$,

$$Y(z) = \frac{1}{(1-0.9z^{-1})^2} = \left[\frac{1}{0.9}\cdot z\right]\cdot\left[-z\cdot\frac{d}{dz}\left(\frac{1}{1-0.9z^{-1}}\right)\right] \Longrightarrow y[n] = (n+1)(0.9)^n u[n+1].$$

When $n = -1$, $y[n] = 0$. So, we can directly plot 99 non-zero samples ($N = 0 \sim 98$)

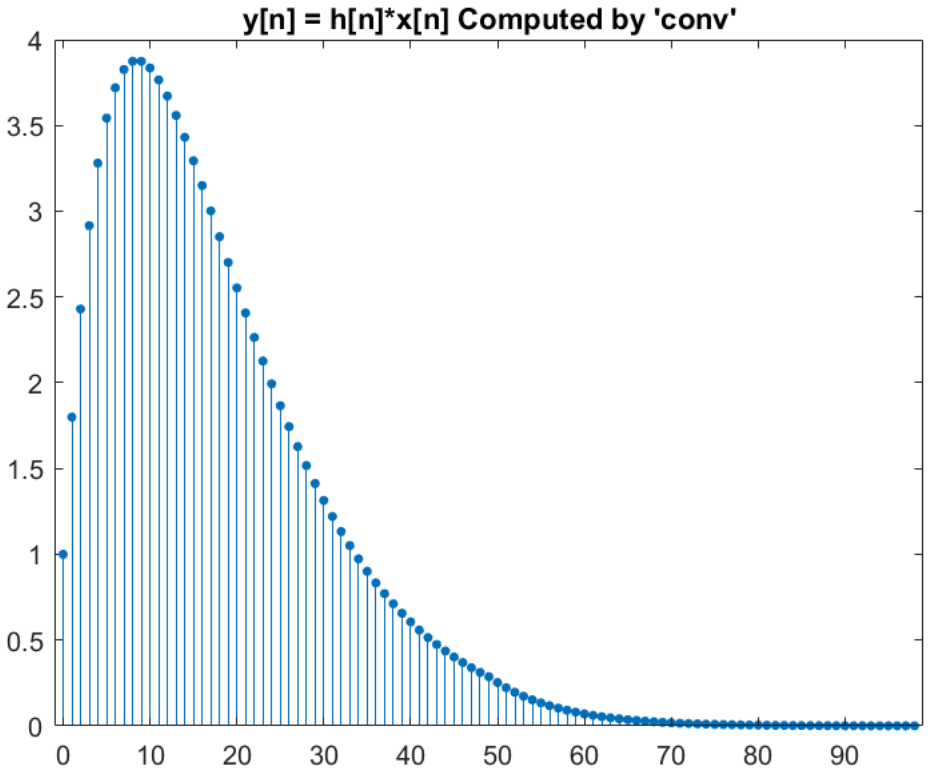
```
N = 0:98;
yn_analytic = (N+1).*(0.9.^N);
figure;
stem(N, yn_analytic,'filled', 'MarkerSize', 3);
title('y[n] = h[n]*x[n] Analytical Expression');
xlim([-1, 99]);
```



(b) Using the **conv** function to compute $y[n]$.

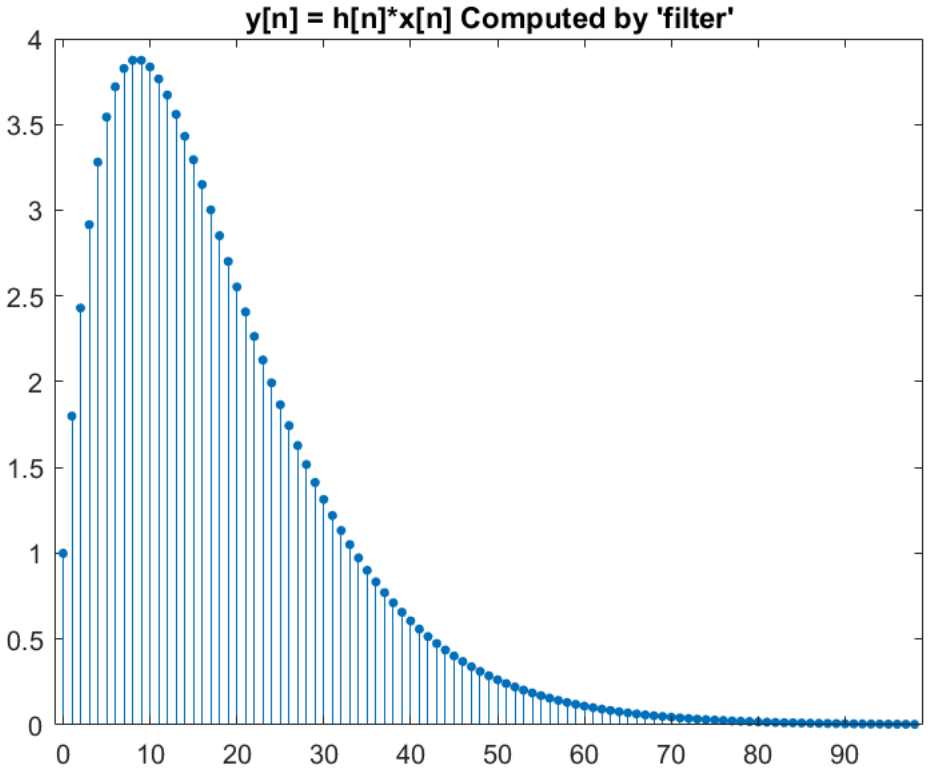
(When we do convolution with **M samples** $x[n]$ and **N samples** $h[n]$, we will get **M+N-1 samples** $y[n]$.)

```
xn = 0.9.^N;
hn = 0.9.^N;
yn_conv = conv(xn(1:50), hn(1:50));
figure;
stem(N, yn_conv,'filled', 'MarkerSize', 3);
title("y[n] = h[n]*x[n] Computed by 'conv'");
xlim([-1, 99]);
```



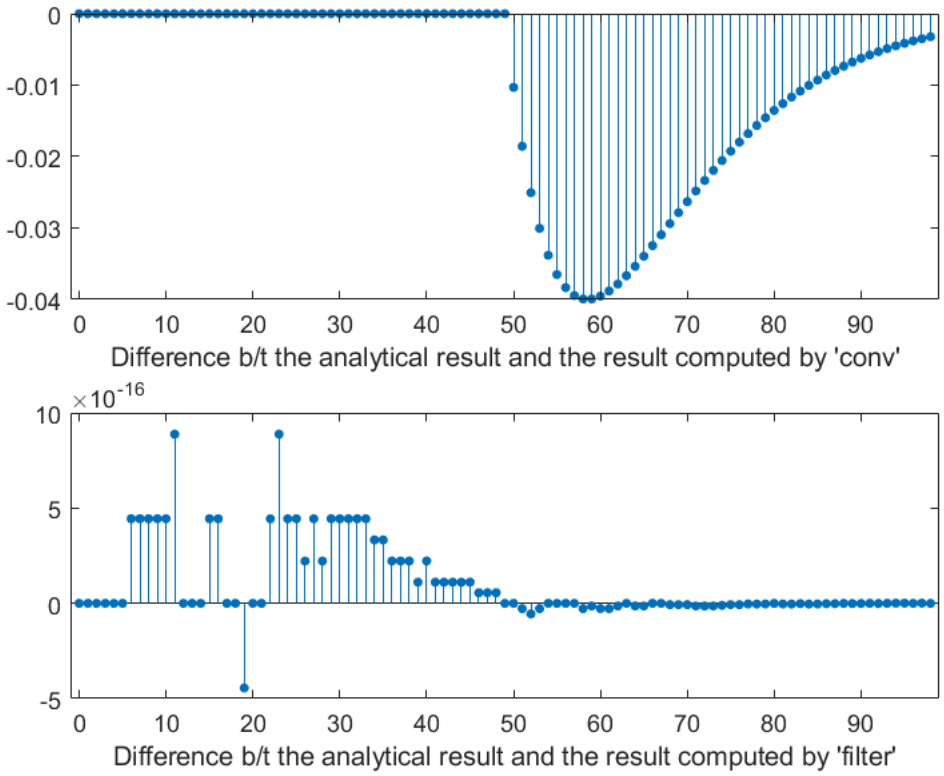

(c) Using the **filter** function to compute $y[n]$.

```
yn_filter = filter([1], [1, -0.9], xn);
figure;
stem(N, yn_filter,'filled', 'MarkerSize', 3);
title("y[n] = h[n]*x[n] Computed by 'filter'");
xlim([-1, 99]);
```




(d) (c) comes closer to (a). Because in (b) the tail parts (samples from n = 50) of both $x[n]$ and $h[n]$ are curtailed, the second part samples (samples from n = 50) of (b) differ from the ones in (a).

```
figure;
subplot(2, 1, 1);
stem(N, yn_conv-yn_analytic,'filled', 'MarkerSize', 3);
xlabel("Difference b/t the analytical result and the result computed by 'conv'");
xlim([-1, 99]);
subplot(2, 1, 2);
stem(N, yn_filter-yn_analytic,'filled', 'MarkerSize', 3);
xlabel("Difference b/t the analytical result and the result computed by 'filter'");
xlim([-1, 99]);
```



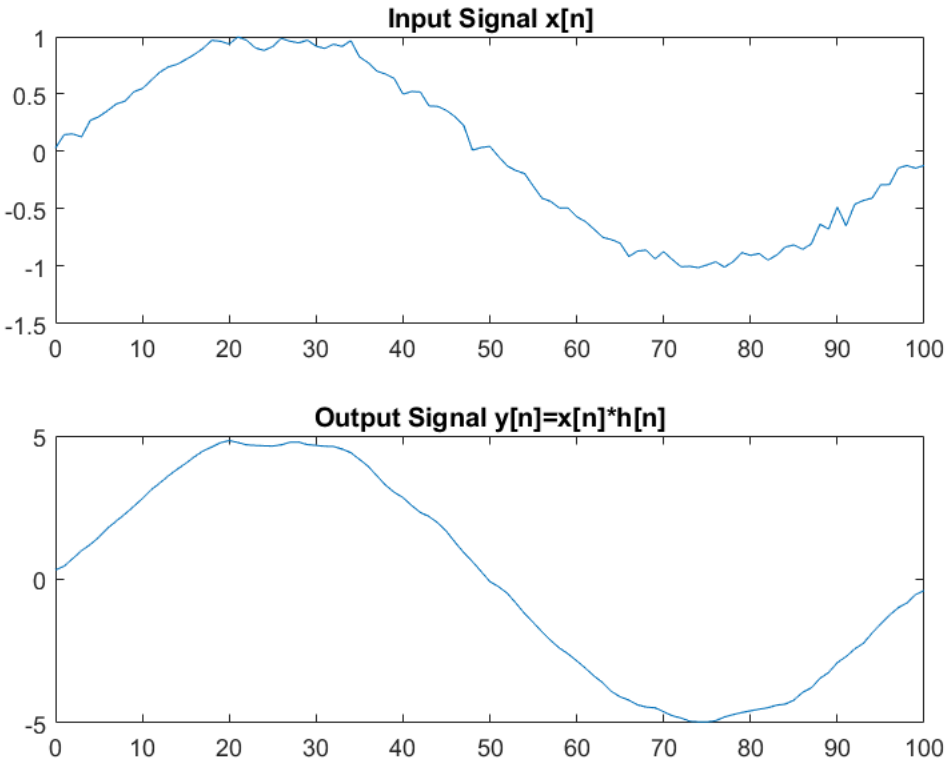

## Problem 2.

(a) By $A_x = \sum_n x[n]$, $A_h = \sum_n h[n]$, and equation 2.36, we can prove the area property of convolution as follow:

$$A_y = \sum_n y[n] = \sum_n \left(\sum_k x[k]h[n-k]\right) = \sum_k x[k]\sum_n h[n-k] = A_x A_h.$$

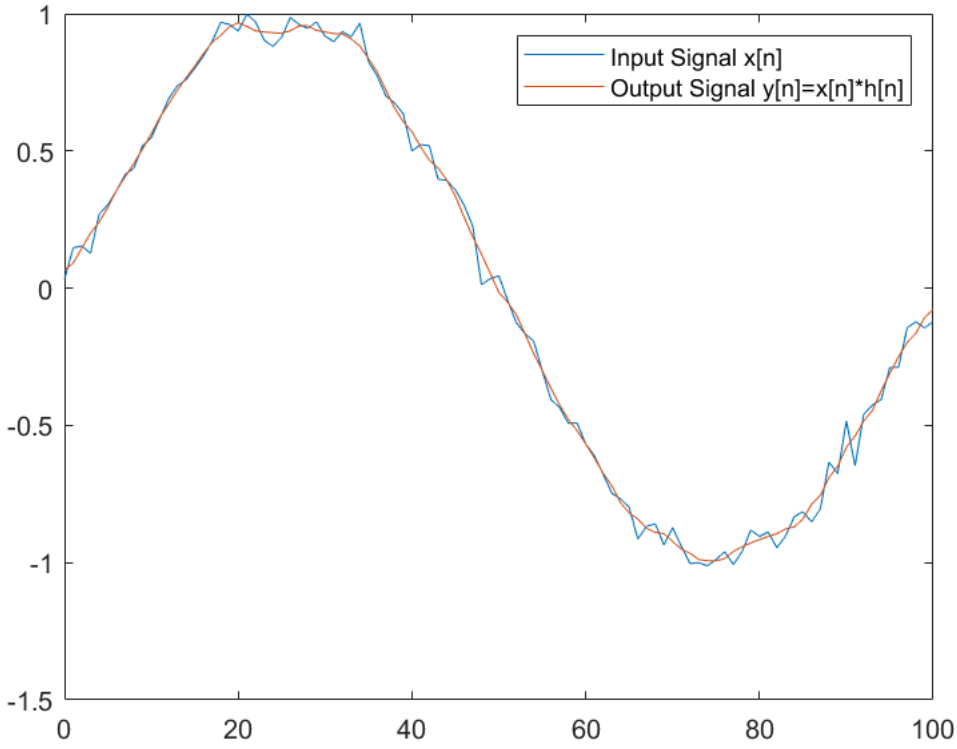
(b) $h = [1\ 1\ 1\ 1\ 1] \Longrightarrow A_h = \sum_n h[n] = 5$, we can show that the amplitude of the **output signal y[n]** is 5 times larger than **input signal x[n]**.

```
N = 101;
n = 0:N-1;
x = sin(2*pi*0.01*n) + 0.05*randn(1, N);
h = ones(1, 5);
y = conv(x, h, 'same');
figure;
subplot(2, 1, 1);
plot(n, x);
title('Input Signal x[n]');
subplot(2, 1, 2);
plot(n, y);
title('Output Signal y[n]=x[n]*h[n]');
```




(c) After normalization, $h = \left[\frac{1}{5}\ \frac{1}{5}\ \frac{1}{5}\ \frac{1}{5}\ \frac{1}{5}\right]$, and the absolute sum $A_h = \sum_n h[n] = 1$. The amplitude of the **output signal y[n]** is the same as the **input signal x[n]**.

```
h_norm = ones(1, 5)/5;
y_norm = conv(x, h_norm, 'same');
figure;
plot(n, x, n, y_norm);
legend(["Input Signal x[n]", "Output Signal y[n]=x[n]*h[n]"]);
```




(d) When $A_h = 5$, amplitude of the signal $y[n]$ in (b) is 5 times lager than $x[n]$. But when $A_h = 1$, amplitude of the signal $y[n]$ in (c) is preserved.

(We usually let $A_h = 1$, such that we can apply filter without magnitude distortion.)

## Problem 3.

(a) $x[n] = u[n] \Longrightarrow X(z) = \frac{1}{1 - z^{-1}}$, $y[n] = 2 \cdot \left(\frac{1}{3}\right)^n u[n] \Longrightarrow Y(z) = \frac{2}{1 - \left(\frac{1}{3}\right)z^{-1}}$, we can find the impulse response h[n] of the system:

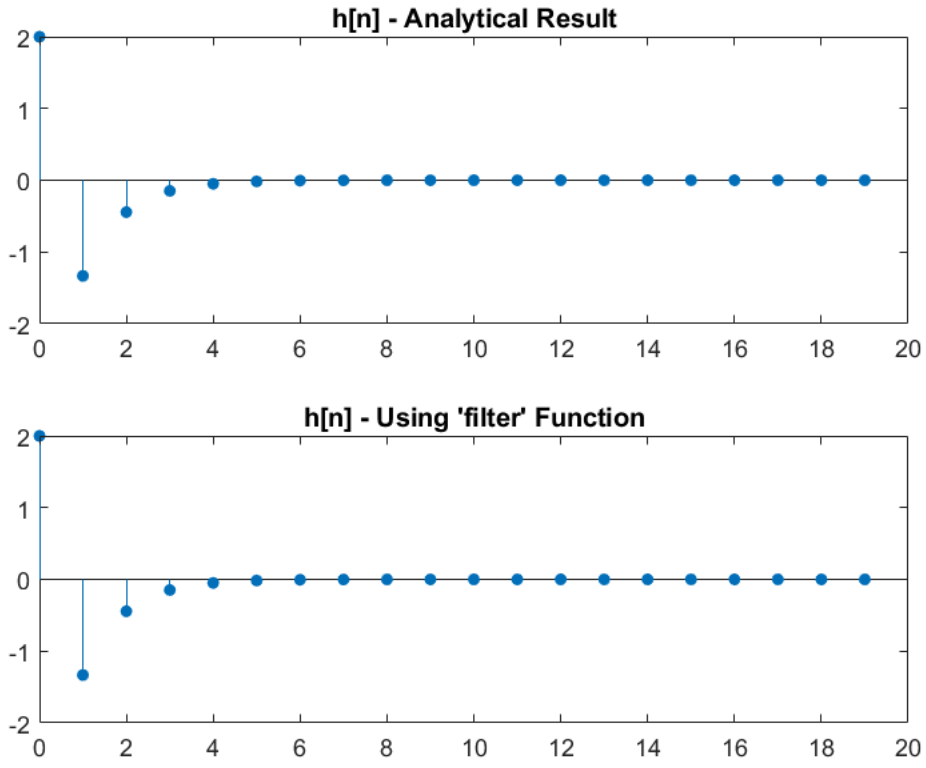
$$H(z) = \frac{Y(z)}{X(z)} = \frac{2 \cdot (1 - z^{-1})}{1 - \left(\frac{1}{3}\right)z^{-1}} = \frac{2 - 2z^{-1}}{1 - \left(\frac{1}{3}\right)z^{-1}} = 6 + \frac{-4}{1 - \left(\frac{1}{3}\right)z^{-1}}, \text{ ROC} : |z| > \frac{1}{3} \Longrightarrow h[n] = 6 \cdot \delta[n] - 4 \cdot \left(\frac{1}{3}\right)^n u[n].$$

```
n = 0:19;
impulse = n==0;
% analytical result
h_analytical = 6*impulse-4*(1/3).^n;

% use the "filter" function
b = [2 -2];
a = [1 -1/3];figure(2);
```

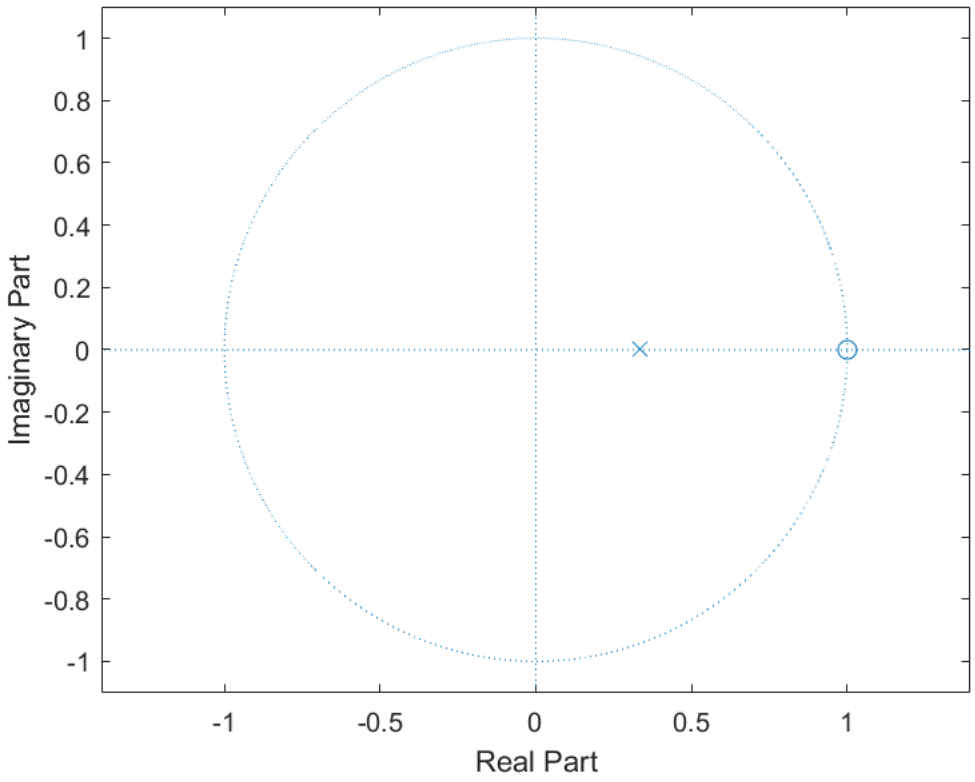
```
h_filter = filter(b, a, impulse);

figure;
subplot(2, 1, 1);
stem(n, h_analytical,'filled', 'MarkerSize', 4);
title("h[n] - Analytical Result");
subplot(2, 1, 2);
stem(n, h_filter,'filled', 'MarkerSize', 4);
title("h[n] - Using 'filter' Function");
```




(b) Use **`zplane`** function to find the locations of pole & zero

```
figure;
zplane(b, a);
```



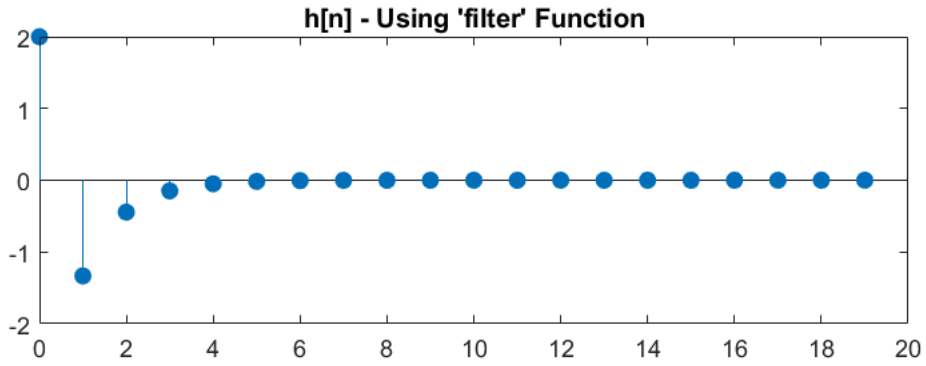

(c) Plot the **impulse response h[n]** using **`filter`** & **`stem`**, and compare with the plot obtained using the function **`impz`**.

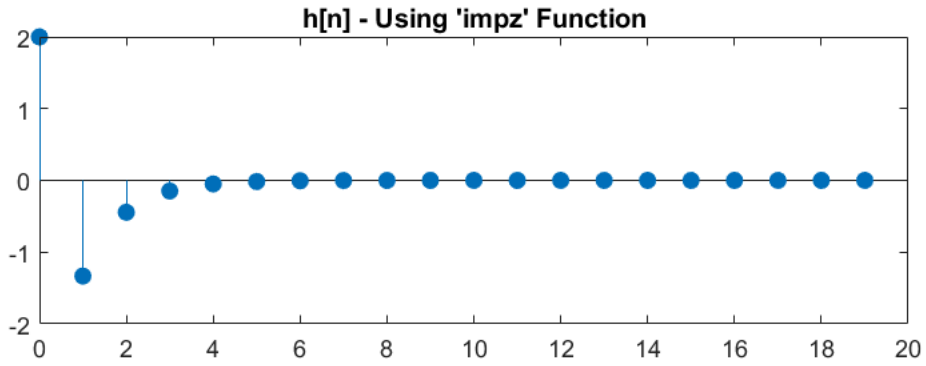
```
impulse = [1 zeros(1,19)];
%unitstep = [1 ones(1,19)];

% use the "filter" function
b = [2 -2];
a = [1 -1/3];
h_filter = filter(b, a, impulse);

% use the "impz" function
[h_impz, n] = impz(b, a, 20);

figure;
subplot(2, 1, 1);
stem(n, h_filter,'filled', 'MarkerSize', 6);
title("h[n] - Using 'filter' Function");
subplot(2, 1, 2);
stem(n, h_impz,'filled', 'MarkerSize', 6);
title("h[n] - Using 'impz' Function");
```




Compare the two sequence by using == operator:

```
h_impz'==h_filter
```

```
ans = 1×20 logical array
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

The comparison in (c) is equivalent, so we can easily use **`impz`** function to find the impulse response h[n] of a system.

(d) From the partial fraction expression $H(z) = \sum_{k=0}^{M-N} C_k z^{-k} + \sum_{k=1}^{N} \frac{r_k}{1 - p_k z^{-1}}$, we can use "**`residuez`**" to help us find the coefficient $r_k$, $p$, and $C_k$.

```
[r, p, C] = residuez(b, a)
```

```
r = -4
p = 0.3333
C = 6
```

By the coefficient $r_k$, $p$, and $C_k$, we can find $H(z) = 6 + \frac{-4}{1 - \left(\frac{1}{3}\right) z^{-1}} \Longrightarrow h[n] = 6 \cdot \delta[n] - 4 \cdot \left(\frac{1}{3}\right)^n u[n]$ (same as the analytical expression in (a)).

## Problem 4.

(a) We use the equation (3.97) $H(z)=\frac{b_0+b_1z^{-1}}{1+a_1z^{-1}+a_2z^{-2}}$. When the system has two **real and equal poles**, we have $a_1{}^2-4a_2=0\Longrightarrow p_{1,2}=-\frac{a_1}{2}$.

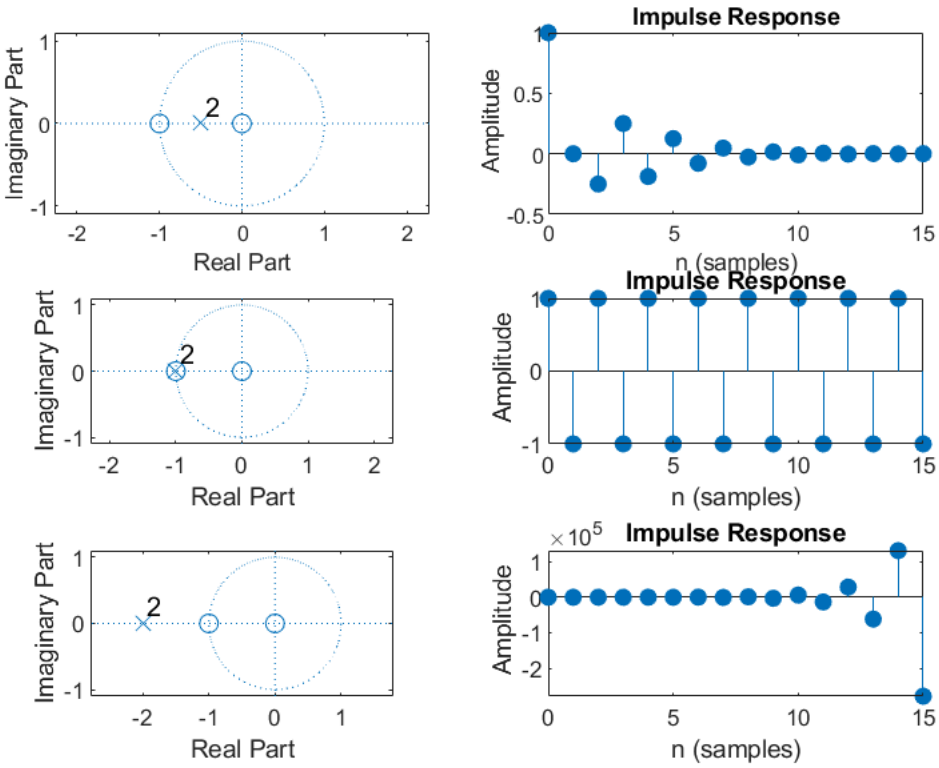
$$H(z)=\frac{b_0+b_1z^{-1}}{\left(1+\frac{a_1}{2}z^{-1}\right)^2}=\frac{b_0}{1+\frac{a_1}{2}z^{-1}}+\left(b_0-\frac{2b_1}{a_1}\right)\cdot\frac{-\frac{a_1}{2}z^{-1}}{\left(1+\frac{a_1}{2}z^{-1}\right)^2},\text{ROC}:|z|>\frac{|a_1|}{2}$$

$$\Longrightarrow h[n]=b_0\left(-\frac{a_1}{2}\right)^n u[n]+\left(b_0-\frac{2b_1}{a_1}\right)\cdot\left(-\frac{a_1}{2}\right)^n n\cdot u[n].$$

- If $|a_1|<2$, the system $h[n]$ is stable and its shape is decaying.
- If $|a_1|=2$, the system $h[n]$ is unstable and is envelope is constant.
- If $|a_1|>2$, the system $h[n]$ is unstable and its shape is growing.

(b) We consider a simple case: $b=[b_0=1,\ b_1=1]$ and $a=\left[a_0=1,\ a_1,a_2=\left(\frac{a_1}{2}\right)^2\right]$. Discuss the value of $a_1$ in three cases.

```
n = 0:19;
b0 = 1; b1 = 1;
b = [b0 b1];
% Three cases of a1
a0 = 1; a1 = 1; a2 = (a1/2)^2;
a_case1 = [1 a1 a2];
a0 = 1; a1 = 2; a2 = (a1/2)^2;
a_case2 = [1 a1 a2];
a0 = 1; a1 = 4; a2 = (a1/2)^2;
a_case3 = [1 a1 a2];
% hn = b0*(-a1/2).^n+(b0-2*b1/a1).*(-a1/2).^n.*n;
figure;
subplot(3, 2, 1);
zplane(b, a_case1);
subplot(3, 2, 2);
impz(b, a_case1, 16);
subplot(3, 2, 3);
zplane(b, a_case2);
subplot(3, 2, 4);
impz(b, a_case2, 16);
subplot(3, 2, 5);
zplane(b, a_case3);
subplot(3, 2, 6);
impz(b, a_case3, 16);
```




## Problem 5.

(a) From the LCCDE $y[n] = 2\cos(\omega_0)y[n-1] - y[n-2], y[-1] = 0, y[-2] = -A\sin(\omega_0)$, the constant coefficient $b_0 = b_1 = b_2 = 0, a_0 = 1, a_1 = -2\cos(\omega_0), a_2 = 1$.

we can use one-sided z transform in suplement:

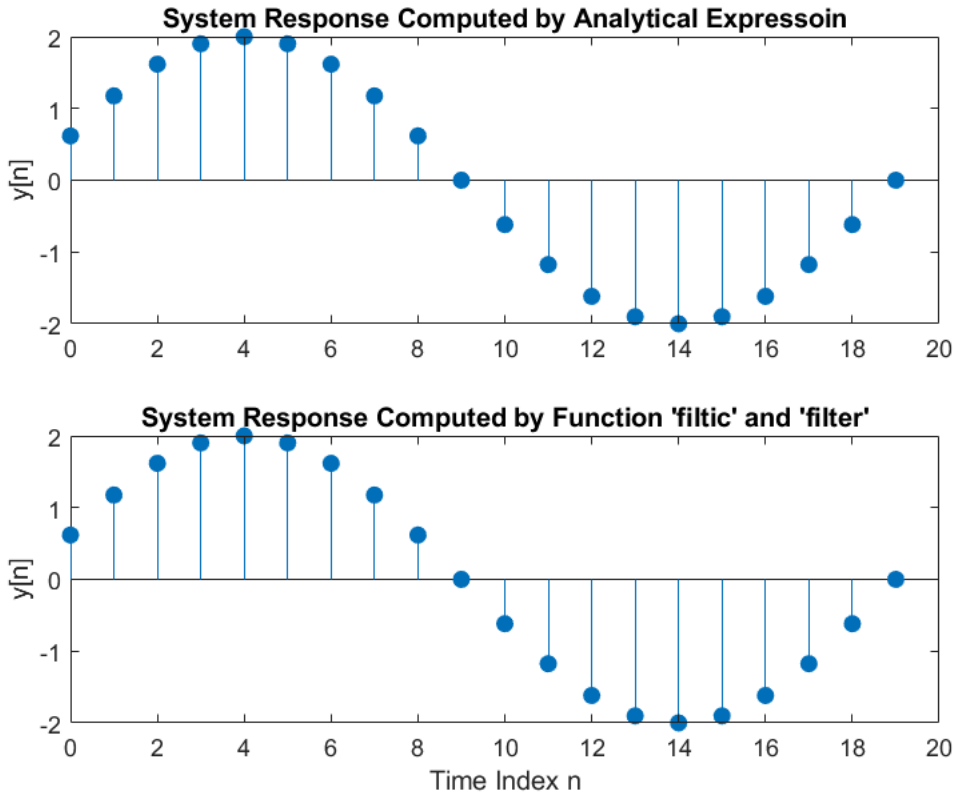
$$Y^+(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} X^+(z) + \frac{(b_2 x[-1] - a_2 y[-1])z^{-1} + (b_1 x[-1] + b_2 x[-2] - a_1 y[-1] - a_2 y[-2])}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{A\sin(\omega_0)}{1 - 2\cos(\omega_0)z^{-1} + a_2 z^{-2}},$$

$$\Longrightarrow y[n] = A\sin[(n+1)\omega_0] \cdot u[n+1] = A\sin[(n+1)\omega_0] \cdot u[n].$$

(b) Compare the analytical result with **`filtic`** + **`filter`** function

```
% Analytical Result:
n = 0:19;
A = 2;
w0 = 0.1*pi;
yn1 = A*sin(w0*(n+1));
% Matlab verification:
b = 0;
a = [1 -2*cos(w0) 1];
yi = [0 -A*sin(w0)];
xi = 0;
```

```
zi = filtic(b,a,yi,xi);
yn2 = filter(b,a,zeros(1,20),zi);
% Plot
figure;
subplot(211)
stem(n,yn1,'filled')
ylabel('y[n]','FontSize',8)
title('System Response Computed by Analytical Expressoin','FontSize',12)
subplot(212)
stem(n,yn2,'filled')
xlabel('Time Index n','FontSize',8)
ylabel('y[n]','fontsize',8)
title("System Response Computed by Function 'filtic' and 'filter'",'FontSize',12)
```




With this verification, we know how to use **filtic + filter** function to help us solve one-side z-tansform problem with specific initial conditions.

```
max(yn1-yn2)
```

```
ans = 4.0197e-15
```

In fact, there exist computation error less than ~ 4e-15, which is introduced by float64 (double) representation.

## Problem 6.

(a) Write a function `c=dtfs0(x)` which compute the DTFS coefficients (4.67) of a periodic signal

```
x = [3600 3497i 345 -14i 0.45 23-3i 99+87i]; % random complex input signal x[n]
```

```
c1 = dtfs0(x)
```

```
c1 = 1×7 complex
10^2 ×
   5.8106 + 5.0957i   8.9176 + 2.8736i   9.4012 - 0.8109i   7.4339 - 4.1895i ···
```

```
c2 = dtfs(x)
```

```
c2 = 1×7 complex
10^2 ×
   5.8106 + 5.0957i   8.9176 + 2.8736i   9.4012 - 0.8109i   7.4339 - 4.1895i ···
```

With a randomly chosen complex signal x[n], we have the same result in both **`dtfs`** and **`dtfs0`** function, but not exactly equivlent.

```
max(c1-c2)
```

```
ans = 2.2737e-13 + 1.4211e-14i
```

⋆Larger number (~$10^2$) leads to larger computation error (~$10^{-13}$), but the dynamic range(mantissa) of float64 is always around $10^{15} \sim 10^{16}$.

(b) Write a function `x=idtfs0(c)` which compute the inverse DTFS (4.63)

```
x1 = idtfs0(c1)
```

```
x1 = 1×7 complex
10^3 ×
   3.6000 + 0.0000i  -0.0000 + 3.4970i   0.3450 - 0.0000i   0.0000 - 0.0140i ···
```

```
x2 = idtfs(c2)
```

```
x2 = 1×7 complex
10^3 ×
   3.6000 - 0.0000i  -0.0000 + 3.4970i   0.3450 + 0.0000i   0.0000 - 0.0140i ···
```

we also have the same result in both **`idtfs`** and **`idtfs0`** function. Verification is done.

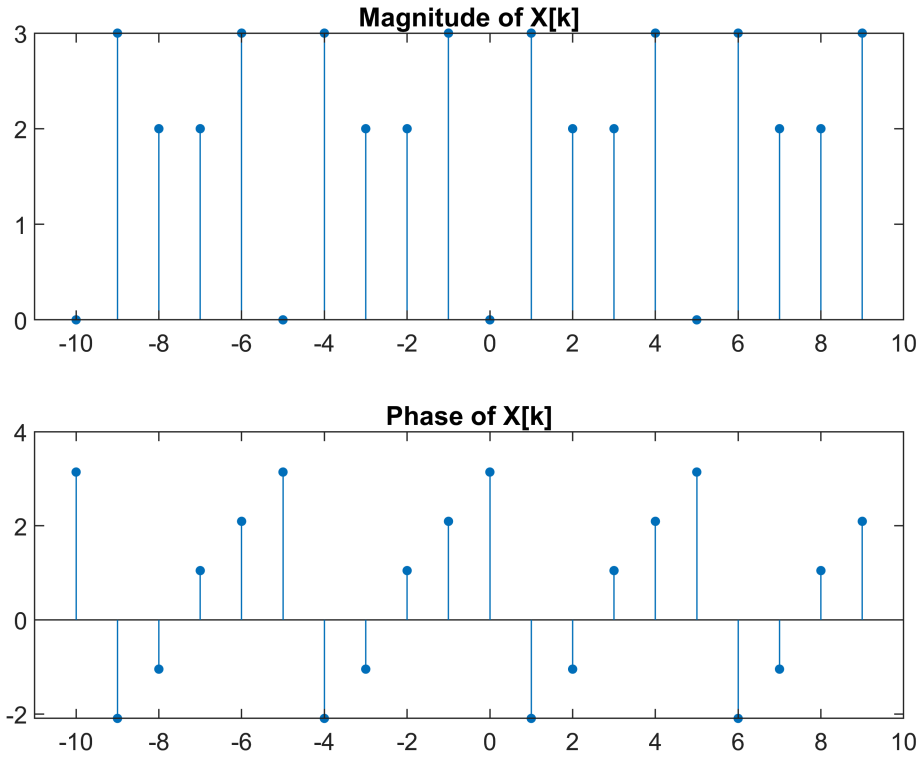
## Problem 7.

Directly Use the **`dtfs`** function to plot the magnitude and phase response

(a) $x_1[n] = 4 \cdot \cos\left(1.2\pi n + \frac{\pi}{3}\right) + 6 \cdot \sin\left(0.4\pi n - \frac{\pi}{6}\right)$ (period $N = 5$.)

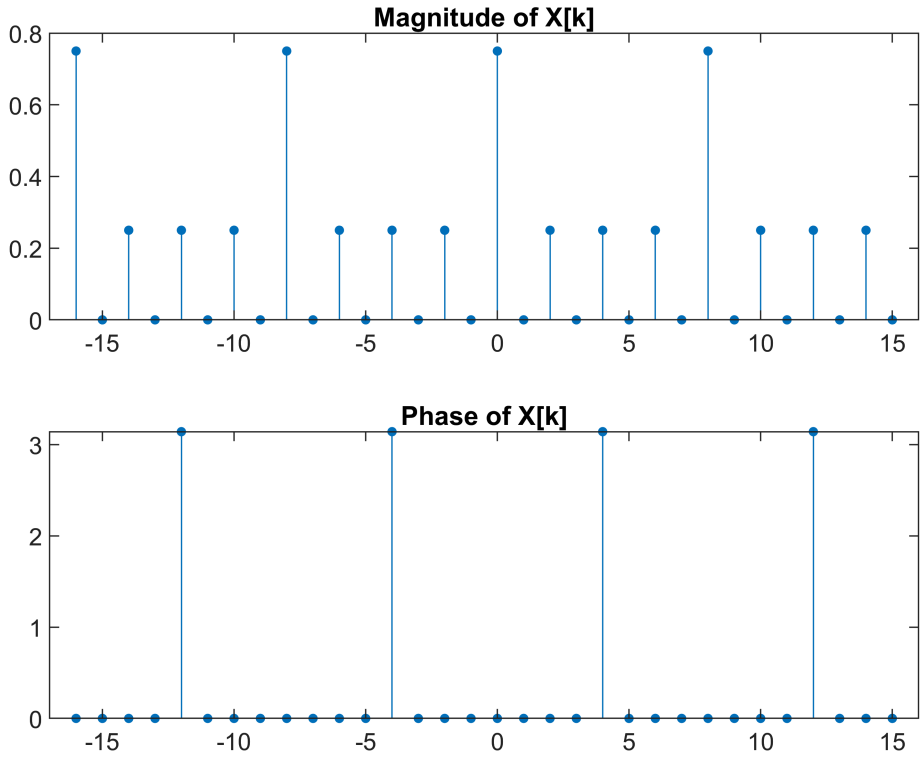
```
N = 5;
n = 0:N-1;
x1n = 4*cos(1.2*pi*n+pi/3)+6*sin(0.4*pi*n-pi/6);
X1k = dtfs(x1n);
figure;
subplot(2, 1, 1);
stem(-2*N:2*N-1, repmat(abs(X1k),[1 4]),'filled', 'MarkerSize', 3);
title('Magnitude of X[k]','FontSize',12)
```

```
xlim([-2*N-1, 2*N]);
subplot(2, 1, 2);
stem(-2*N:2*N-1, repmat(angle(X1k),[1 4]),'filled', 'MarkerSize', 3);
title('Phase of X[k]','FontSize',12)
xlim([-2*N-1, 2*N]);
```



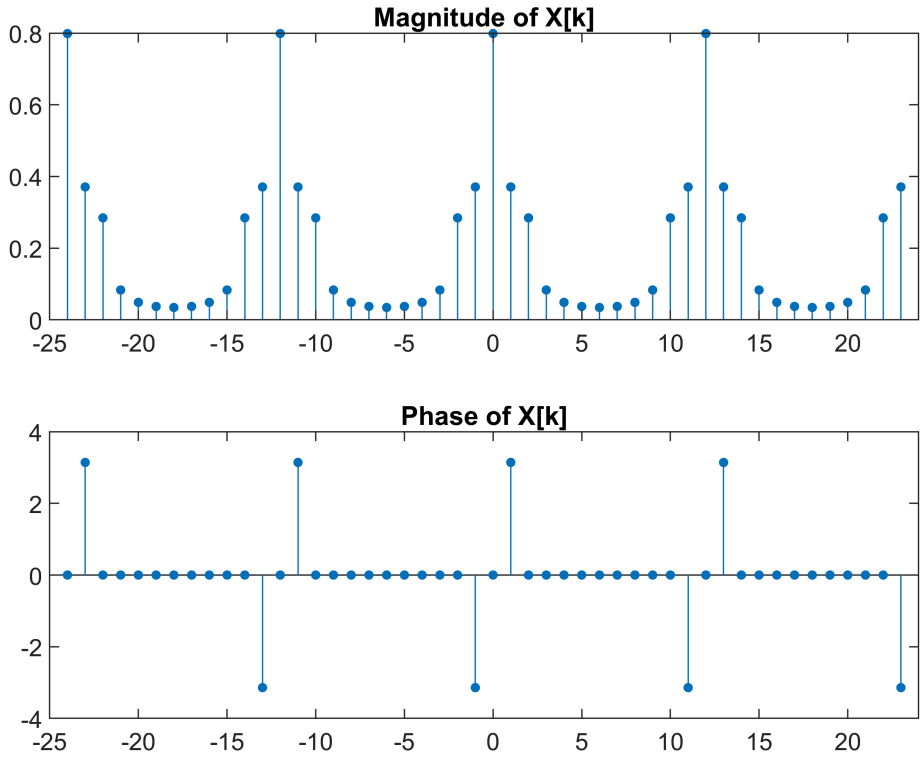

(b) $x_2[n] = \{1, 1, 0, 1, 1, 1, 0, 1\},\ 0 \le n \le 7$ (one period)

```
N = 8;
n = 0:N-1;
x2n = [1 1 0 1 1 1 0 1];
X2k = dtfs(x2n);
figure;
subplot(2, 1, 1);
stem(-2*N:2*N-1, repmat(abs(X2k),[1 4]),'filled', 'MarkerSize', 3);
title('Magnitude of X[k]','FontSize',12)
xlim([-2*N-1, 2*N]);
subplot(2, 1, 2);
stem(-2*N:2*N-1, repmat(angle(X2k),[1 4]),'filled', 'MarkerSize', 3);
title('Phase of X[k]','FontSize',12)
xlim([-2*N-1, 2*N]);
```




(c) $x_3[n] = 1 - \sin\left(\frac{\pi}{4}n\right), 0 \le n \le 11$ (one period)

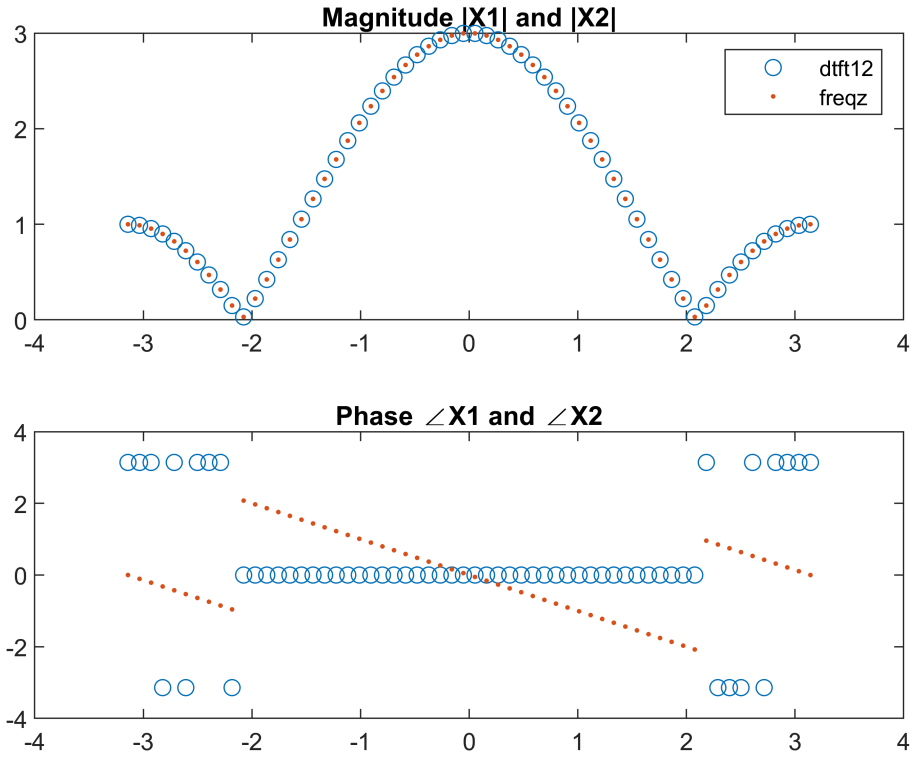
```
N = 12;
n = 0:N-1;
x3n = 1 - sin(0.25*pi*n);
X3k = dtfs(x3n);
figure;
subplot(2, 1, 1);
stem(-2*N:2*N-1, repmat(abs(X3k),[1 4]),'filled', 'MarkerSize', 3);
title('Magnitude of X[k]','FontSize',12)
xlim([-2*N-1, 2*N]);
subplot(2, 1, 2);
stem(-2*N:2*N-1, repmat(angle(X3k),[1 4]),'filled', 'MarkerSize', 3);
title('Phase of X[k]','FontSize',12)
xlim([-2*N-1, 2*N]);
```




## Problem 8.

(a) plot the magnitude |X1|(**dtft12**), |X2|(**freqz**) and the phase ∠X1(**dtft12**), ∠X2(**freqz**)

```
x=[1 1 1]; % n=-1,0,1
om=linspace(-pi, pi, 60);
X1=dtft12(x, -1, om); X2=freqz(x, 1, om);
subplot(2, 1, 1);
plot(om, abs(X1), 'o', om, abs(X2), '.');
title('Magnitude |X1| and |X2|','FontSize',12)
legend ("dtft12", "freqz");
subplot(2, 1, 2);
plot(om, angle(X1), 'o', om, angle(X2), '.');
title('Phase \angleX1 and \angleX2','FontSize',12)
```




(b) From the graph, we can show that $|\mathrm{X1}| = |\mathrm{X2}|$, but $\angle \mathrm{X1} \neq \angle \mathrm{X2}$.

Since we represent the index $n = [-1,\ 0,\ 1]$ in **`dtft12`** function, and the index $n = [0,\ 1,\ 2]$ in **`freqz`** function, we obtain the correct phase response by the **`dtft12`** function, and the phase response with a phase shifting effect by the **`freqz`** function.

(More details in the textbook p.169)

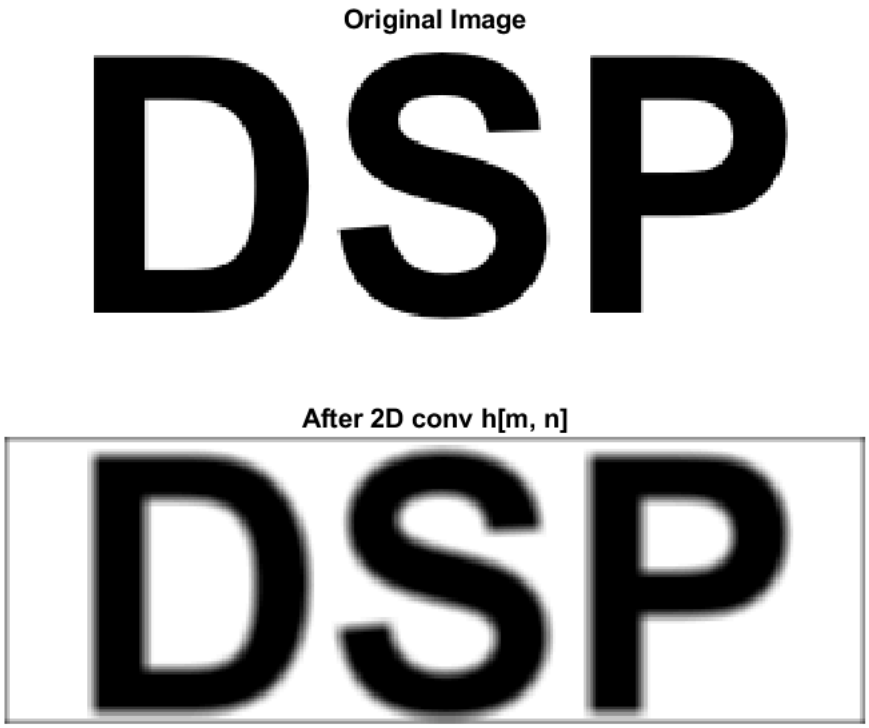
## Problem 9.

(a) After using the 5x5 square filter $h[m,\ n]$, we will notice two effects:

- Square filter(average filter) blurs the image. (Larger size of average filter cause more blurring effect.)
- We have boundary artifact in processing image signal boundary. (We often handle these problems with reflect/symmetric padding.)

```
% load image file
img = imread('DSP.png');
% apply 2D convolution
h = ones(5)/25;
img_blur = uint8(conv2(double(img), h, 'same'));
% plot
figure;
subplot(211);
imshow(img);
title('Original Image');
```

```
subplot(212);
imshow(img_blur);
title('After 2D conv h[m, n]');
```




(b) **Sobel filter** is a famous and useful filter in the image processing, which can use to **detect edges** or **emphasize boundary**.

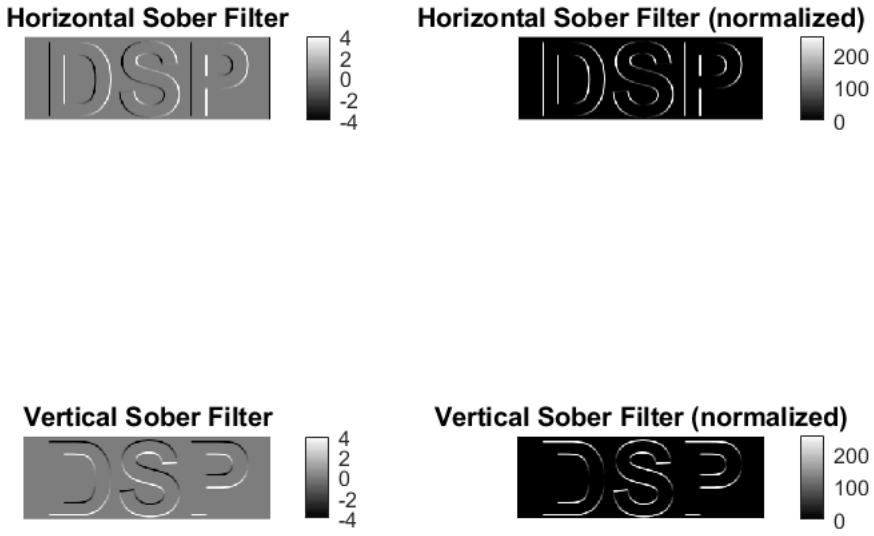
We can use the **horizontal Sobel filter** $h_1[m, n]$ and the **vertical Sobel filter** $h_2[m, n]$ to emphasize horizontal and vertical edges respectively.

(The mathematical meaning of the Sobel filter is just an one-order difference equation $y[n] = \frac{x[n+1] - x[n-1]}{2}$)

(When we do image proccessing, we often nomalize the pixel range from **unit8-[0, 255]** to **float64-[0.0, 1.0]**)

```
% Apply Horizontal & Vertical Sobel filters using conv2
h2 = [1 0 -1; 2 0 -2; 1 0 -1];
h3 = [1 0 -1; 2 0 -2; 1 0 -1]';
img_conv_h1 = conv2(double(img)/255.0, h2, 'same');
img_conv_h2 = conv2(double(img)/255.0, h3, 'same');
% plot
figure;
subplot(221);
imshow(img_conv_h1, [min(img_conv_h1, [], 'all'), max(img_conv_h1, [], 'all')])
colorbar;
title("Horizontal Sober Filter");
subplot(222);
imshow(uint8(abs(img_conv_h1*255.0/4)))
```

```
colorbar;
title("Horizontal Sober Filter (normalized)");
subplot(223);
imshow(img_conv_h2, [min(img_conv_h2, [], 'all'), max(img_conv_h2, [], 'all')])
colorbar;
title("Vertical Sober Filter");
subplot(224);
imshow(uint8(abs(img_conv_h2*255.0/4)))
colorbar;
title("Vertical Sober Filter (normalized)");
```



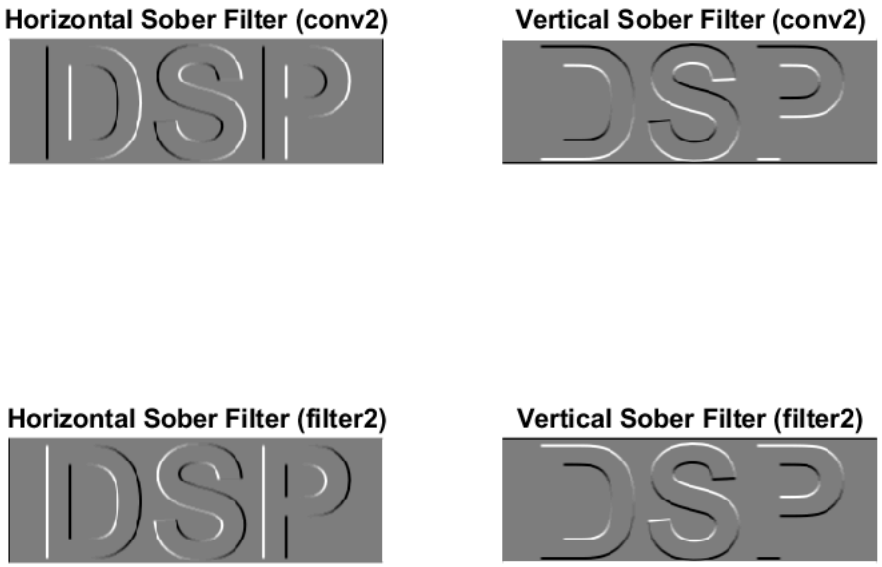

(c) **`filter2(H,X,shape)`** is equivalent to **`conv2(X,rot90(H,2),shape)`**

When the 2D filter we applied is symmetric, the results of using **`filter2`** function and **`conv2`** function are equivalent.

But if the 2D filter we applied is non-symmetric, we should be careful in using these two functions.

```
% Apply Horizontal & Vertical Sobel filters using filter2
img_filt_h1 = filter2(h2, double(img)/255.0);
img_filt_h2 = filter2(h3, double(img)/255.0);
% plot
figure;
subplot(221);
imshow(img_conv_h1, [min(img_conv_h1, [], 'all'), max(img_conv_h1, [], 'all')])
title("Horizontal Sober Filter (conv2)");
subplot(222);
imshow(img_conv_h2, [min(img_conv_h2, [], 'all'), max(img_conv_h2, [], 'all')])
title("Vertical Sober Filter (conv2)");
```

```
subplot(223);
imshow(img_filt_h1, [min(img_filt_h1, [], 'all'), max(img_filt_h1, [], 'all')])
title("Horizontal Sober Filter (filter2)");
subplot(224);
imshow(img_filt_h2, [min(img_filt_h2, [], 'all'), max(img_filt_h2, [], 'all')])
title("Vertical Sober Filter (filter2)");
```




## Problem 10.

```
% load audio file and play
[y,Fs] = audioread('handel.wav');
playerObj = audioplayer(y,Fs);
play(playerObj);
```

(a) Subsample the audio sequence y with factor 2 and play with Fs/2

```
% reduces the sampling rate by a factor of two & play with Fs/2
n_d2 = 1:2:length(y);
y_d2 = y(n_d2);
playerObj = audioplayer(y_d2,Fs/2);
play(playerObj);
```

(b) Subsample the audio sequence y with factor 4 and play with Fs/4

```
% reduces the sampling rate by a factor of four & play with Fs/4
n_d4 = 1:4:length(y);
y_d4 = y(n_d4);
```

```
playerObj = audioplayer(y_d4,Fs/4);
play(playerObj);
```

(c) When we directly subsample the sequence without any preprocessing, high frequency components will disappear but remain some additional low-frequency artifacts, which is called "**Aliasing**".

(If we just want to eliminate high frequency component, and preserve low frequency component as before, we need apply low-pass filter before subsampling. The cut-off frequency is related to **Nyquist rate** (you will see it in Chapter 6).)

```
Y = dtfs(y);
Y_d2 = dtfs(y_d2);
Y_d4 = dtfs(y_d4);
% plot
figure;
subplot(311);
plot(abs(Y(1:round(length(Y)/2))));
xlim([0, round(length(y)/2)]);
xlabel('k')
subplot(312);
plot(abs(Y_d2(1:round(length(Y_d2)/2))));
xlim([0, round(length(y)/2)]);
xlabel('k')
subplot(313);
plot(abs(Y_d4(1:round(length(Y_d4)/2))));
xlim([0, round(length(y)/2)]);
xlabel('k')
```