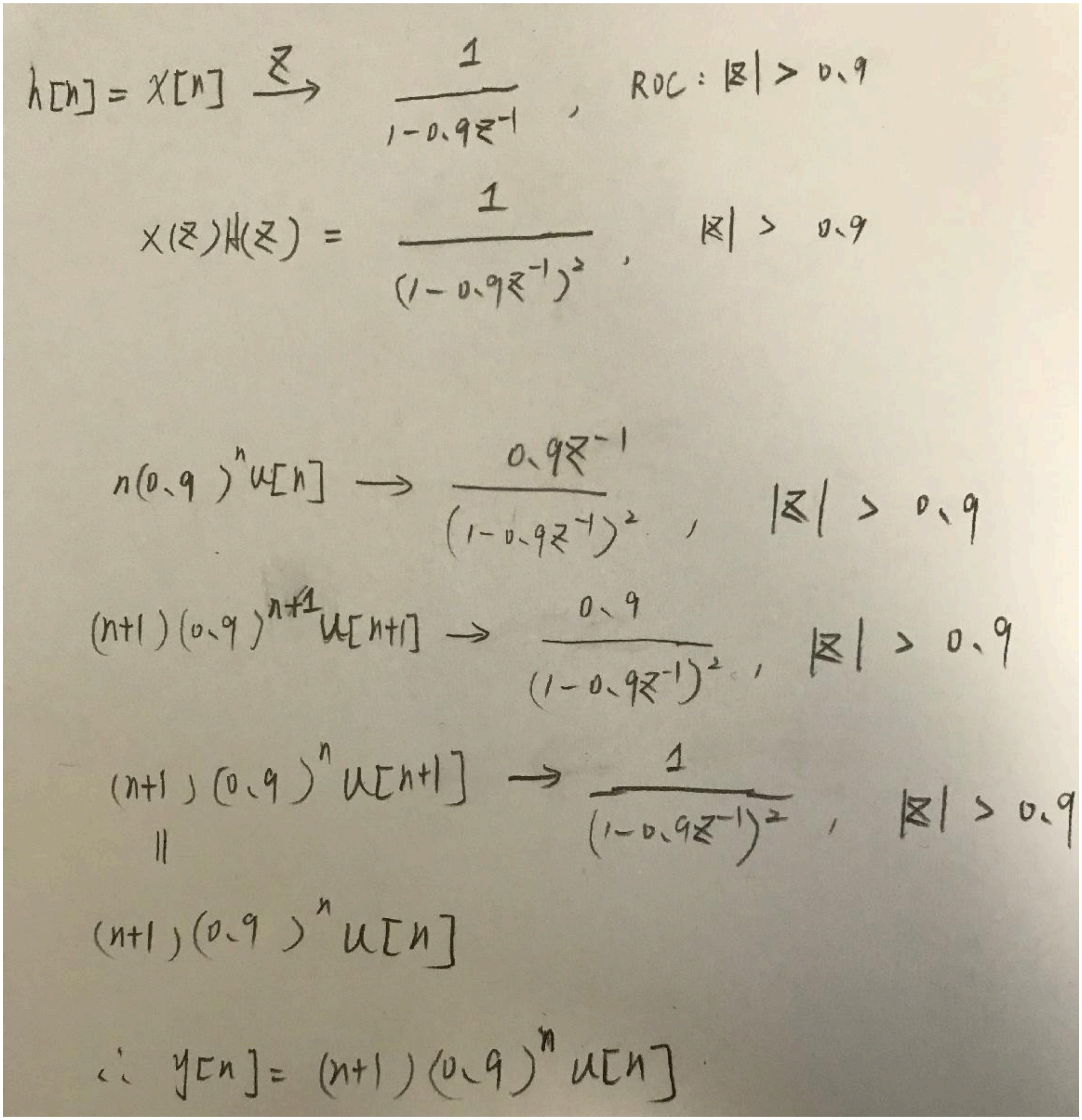
DSP HW2 Program Assignment

1. Let x[n] = h[n] = $(0.9)^n$ u[n] and y[n] = x[n]*h[n]

(a) Determine y[n] analytically and plot the first 99 non-zero samples using the stem function



$$h[n] = x[n] \xrightarrow{z} \frac{1}{1-0.9z^{-1}}, \quad \text{ROC}: |z| > 0.9$$

$$X(z)H(z) = \frac{1}{(1-0.9z^{-1})^2}, \quad |z| > 0.9$$

$$n(0.9)^n u[n] \rightarrow \frac{0.9z^{-1}}{(1-0.9z^{-1})^2}, \quad |z| > 0.9$$

$$(n+1)(0.9)^{n+1} u[n+1] \rightarrow \frac{0.9}{(1-0.9z^{-1})^2}, \quad |z| > 0.9$$

$$(n+1)(0.9)^n u[n+1] \rightarrow \frac{1}{(1-0.9z^{-1})^2}, \quad |z| > 0.9$$

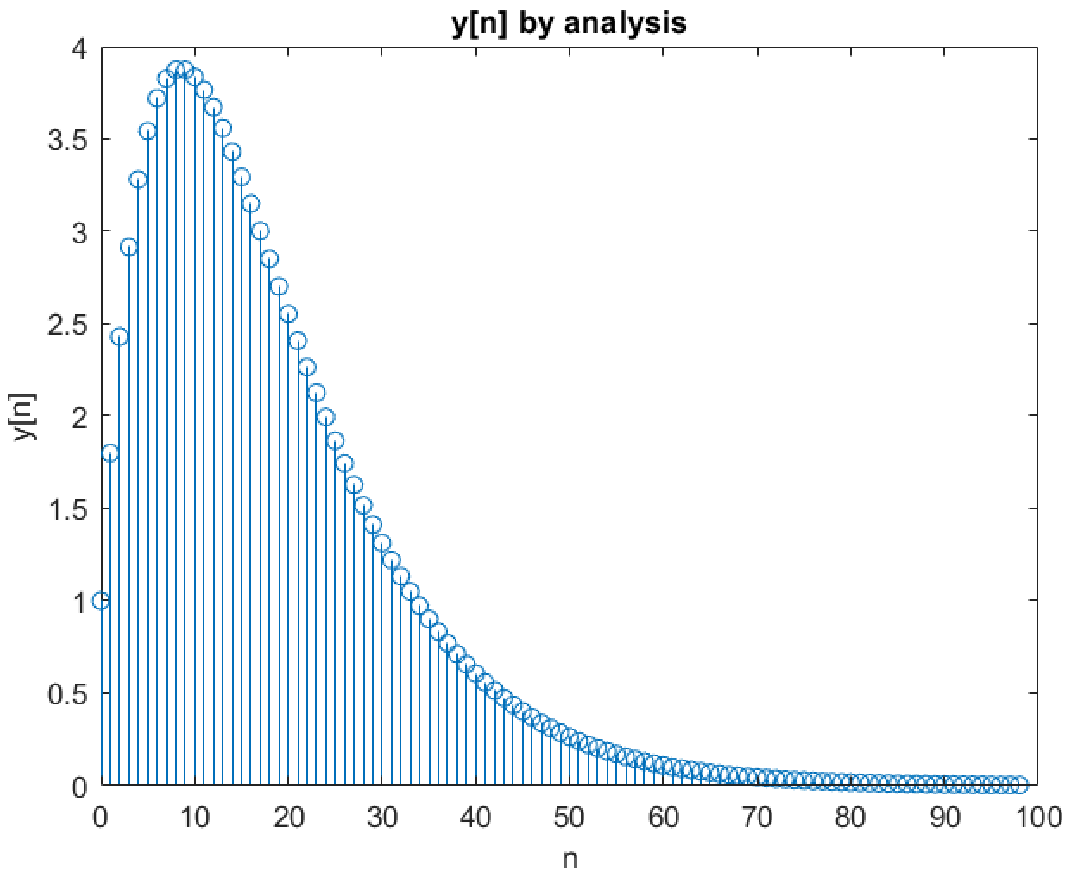
$$\| $$

$$(n+1)(0.9)^n u[n]$$

$$\therefore y[n] = (n+1)(0.9)^n u[n]$$

```
N = 99;
y = zeros(1, 99);  %pre-allocate y
```

```
for n = 1:N
    y(n) = n*0.9^(n-1);
end
index = 0:98;
stem(index, y);
xlabel("n");
ylabel("y[n]")
title("y[n] by analysis");
```

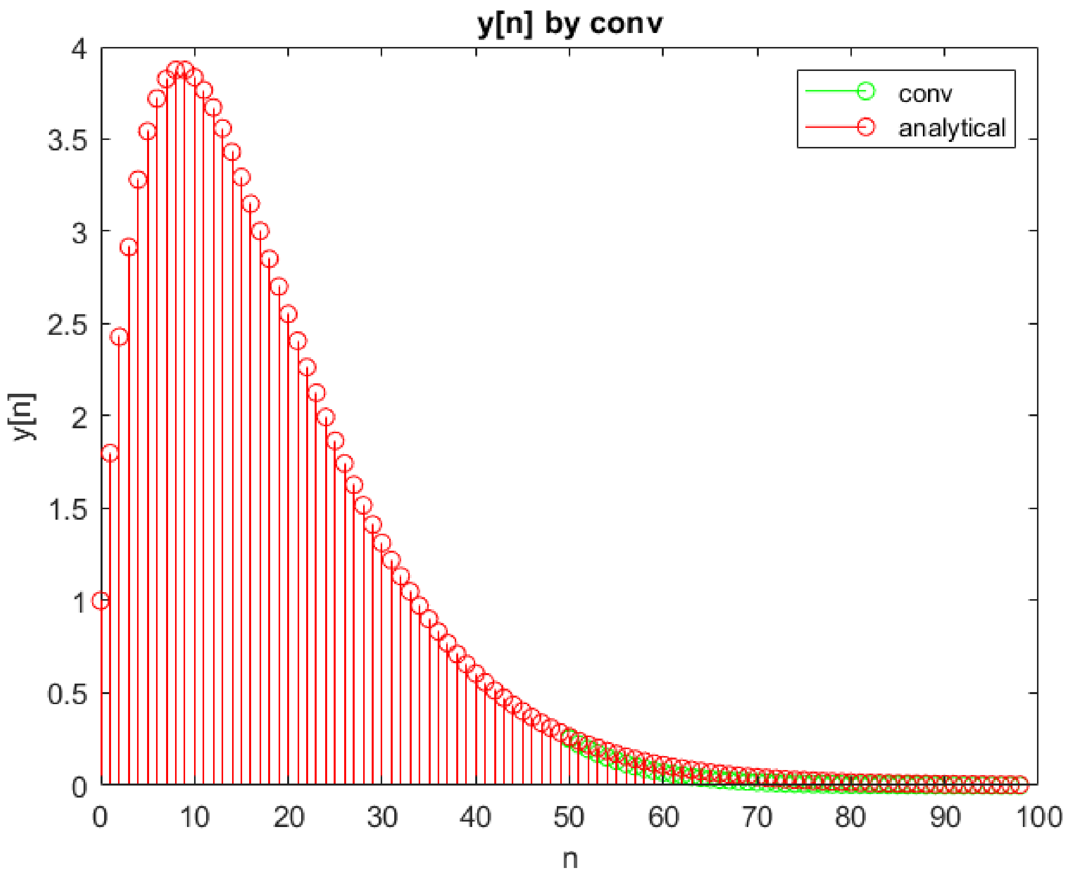


(b) Take first 50 samples of x[n] and h[n]. Compute and plot y[n] using the conv function.

I draw the analytical result from (a) and signal from thie problem together to compare their difference.

```
N = 50;
f = zeros(1, 50);  %pre-allocate f
for n = 1:N
    f(n) = 0.9^(n-1);
end
x = f(1:50);
h = f(1:50);
y_b = conv(x, h);
plot_b = stem(index, y_b, 'g');
xlabel("n");
ylabel("y[n]")
```

```
title("y[n] by conv");
hold on
plot_a = stem(index, y, 'r');
hold off
legend([plot_b plot_a],{'conv','analytical'})
```

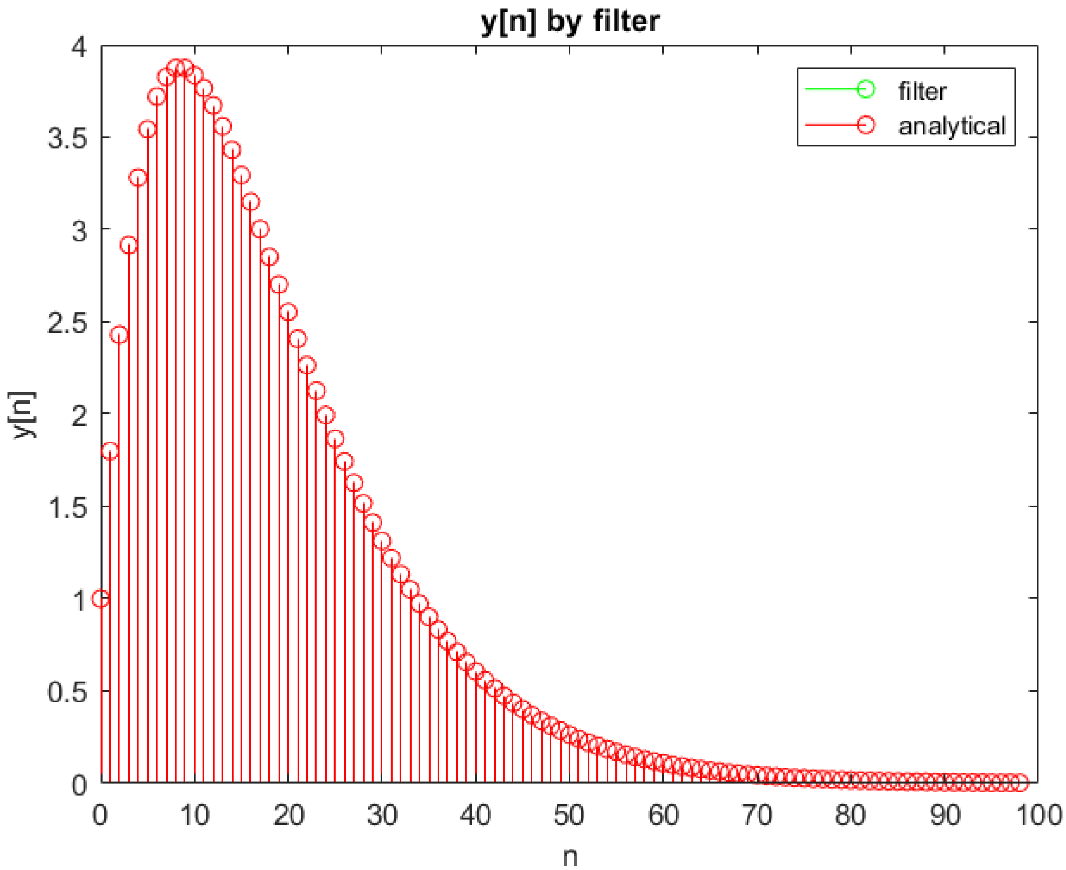


(c) Using the filter function, determine and plot the first 99 samples of y[n].

I draw the analytical result from (a) and signal from thie problem together to compare their difference.

```
N = 99;
f = zeros(1, 99);  %pre-allocate f
for n = 1:N
    f(n) = 0.9^(n-1);
end
h = f;
x = f;
y_c = filter(h, 1, x); %system = h/1
plot_c = stem(index, y_c, 'g');
xlabel("n");
ylabel("y[n]")
title("y[n] by filter");
hold on
plot_a = stem(index, y, 'r');
hold off
```

```
legend([plot_c plot_a],{'filter','analytical'})
```



(d) Which of the outputs in (b) and (c) come close to that in (a)? Explain

As graph shown in (b) and (c), there is basically no difference between analytical result and filter function result. And there are some slight differences between analytical and conv function result at n > 50. So the output in (c) comes close to that in (a). I think the reason is filter can apply a system using impulse response coefficient to the input, which will output the the same result as what I analyze in (a). But conv function in (b) only takes the first 50 points although points at n > 50 is small, it slightly the output result as shown in (b)

2. The sum $A_x = \sum_n x[n]$ 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= $A_x A_h$ (derive in the live script)

$$y[n]=\sum_k x[k]h[n-k]\Longrightarrow A_y=\sum_m y[m]=\sum_m\sum_k x[k]h[m-k]=$$
$$\sum_k\sum_m h[m-k]x\left[k\right]=\sum_k x[k]\sum_m h[m-k]=A_h\sum_k x[k]=A_hA_x$$

(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 $A_y = A_xA_h$ and use the <u>subplot</u> function plot x[n] and y[n] on the same graph.

```
x =sin(2*pi*0.01*(0:100)) + 0.05*randn(1,101);
h = ones(1, 5);
y = conv(h, x);
Ay = sum(y);
Ax = sum(x);
Ah = sum(h);
disp(['Ah*Ax = ',num2str(Ah*Ax)])
```

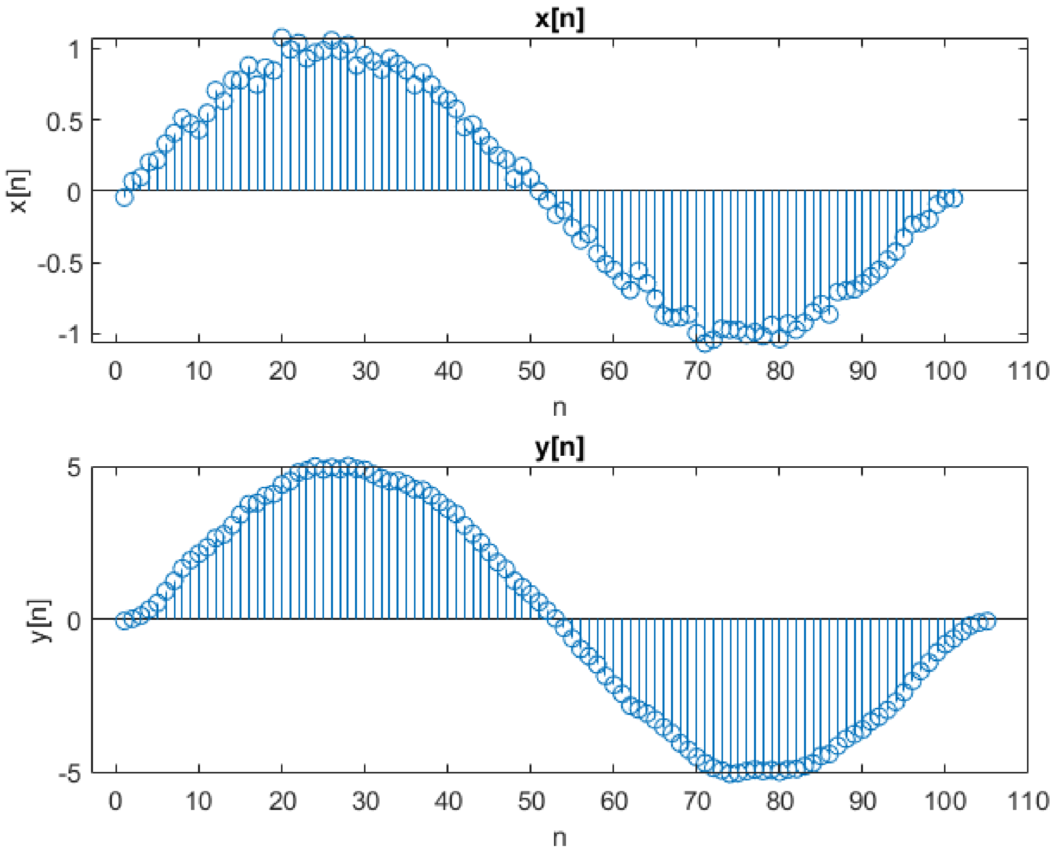
```
Ah*Ax = -2.2388
```

```
Ay
```

```
Ay = -2.2388
```

```
%areEssentiallyEqual = abs(Ay-Ah*Ax) < 200*eps(Ay)


subplot(2,1,1);
stem(x);
xlim([-3 110])
title('x[n]');
xlabel('n');
ylabel('x[n]');
subplot(2,1,2);
stem(y);
xlim([-3 110])
title('y[n]');
xlabel('n');
ylabel('y[n]');
```



According ot ouptut result, $A_y = A_x A_h$

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

```
h = 0.2*ones(1, 5);
y = conv(h, x);
Ay = sum(y);
Ax = sum(x);
Ah = sum(h);
disp(['Ah*Ax = ',num2str(Ah*Ax)])
```

```
Ah*Ax = -0.44775
```

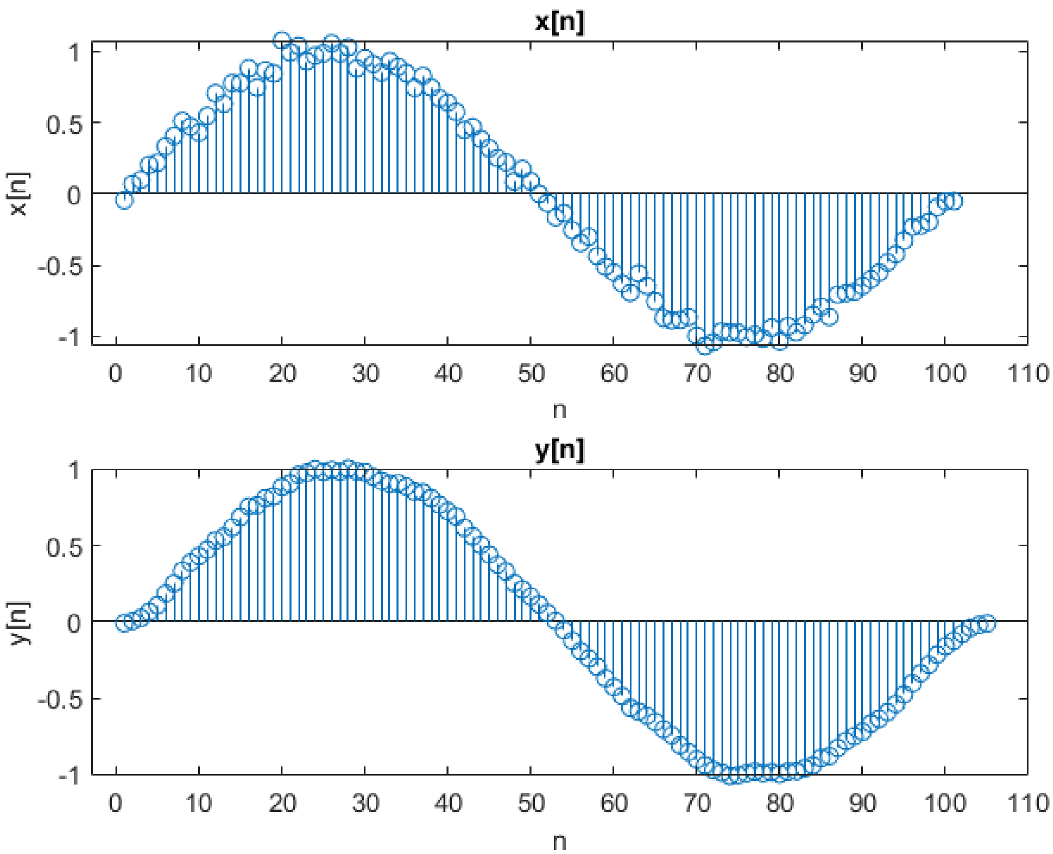
```
Ay
```

```
Ay = -0.4478
```

```
%areEssentiallyEqual = abs(Ay-Ah*Ax) < 200*eps(Ay)


subplot(2,1,1);
stem(x);
xlim([-3 110])
```

```
title('x[n]');
xlabel('n');
ylabel('x[n]');
subplot(2,1,2);
stem(y);
xlim([-3 110])
title('y[n]');
xlabel('n');
ylabel('y[n]');
```



(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).

Ans: Amplitude of y[n] in part (b) is about 5 and that of y[n] in part (c) is about 1. By the dedinition of convolution, firstable, h[n] becomes h[-n], then h[-n] starts shift **right.** Then when it moves to the peak of x[n], whose value is about 1, y[n] is about 1*1+1*1+1*1+1*1+1*1=5 (when h[n] is normalized in part (b)) and about 0.2*1+0.2*1+0.2*1+0.2*1+0.2*1 = 1 (when h[n] is normalized in part (c)).

Also since h[n] is five consecutive same values, it wil make x[n] more smooth especially all five same data point multiplied by x[n]. Because x[n] is a sine funciton with random numbers added at each data point and doing convolution would add 5 consecutive x[n] points together, which could cancel out the random number influence and make the curve smooth.

3. The response of a LTI system to the input x[n] = u[n] is y[n] = 2 $\cdot \left(\frac{1}{3}\right)^n \cdot u[n].$

(a) Find the impulse response h[n] of the system, and check the results using the function <u>filter</u>.

$$u[n] = \sum_{k=0}^{\infty} \delta[n-k]$$

$$y[n] = \sum_{k=0}^{\infty} h[n-k] = 2\left(\tfrac{1}{3}\right)^n u[n]$$

$$\Rightarrow (1+z^{-1}+z^{-2}+\ldots)H(z) = \frac{2}{1-\frac{1}{3}z^{-1}}, \quad |z|>\frac{1}{3}$$

$$\Rightarrow H(z)\cdot\frac{1}{1-z^{-1}} = \frac{2}{1-\frac{1}{3}z^{-1}}, \quad |z|>\frac{1}{3}$$

$$\Rightarrow H(z) = 2\,\frac{(1-z^{-1})}{1-\frac{1}{3}z^{-1}}, \quad |z|>\frac{1}{3}$$

$$= 2\left(\frac{1}{1-\frac{1}{3}z^{-1}} - \frac{z^{-1}}{1-\frac{1}{3}z^{-1}}\right), \quad |z|>\frac{1}{3}$$

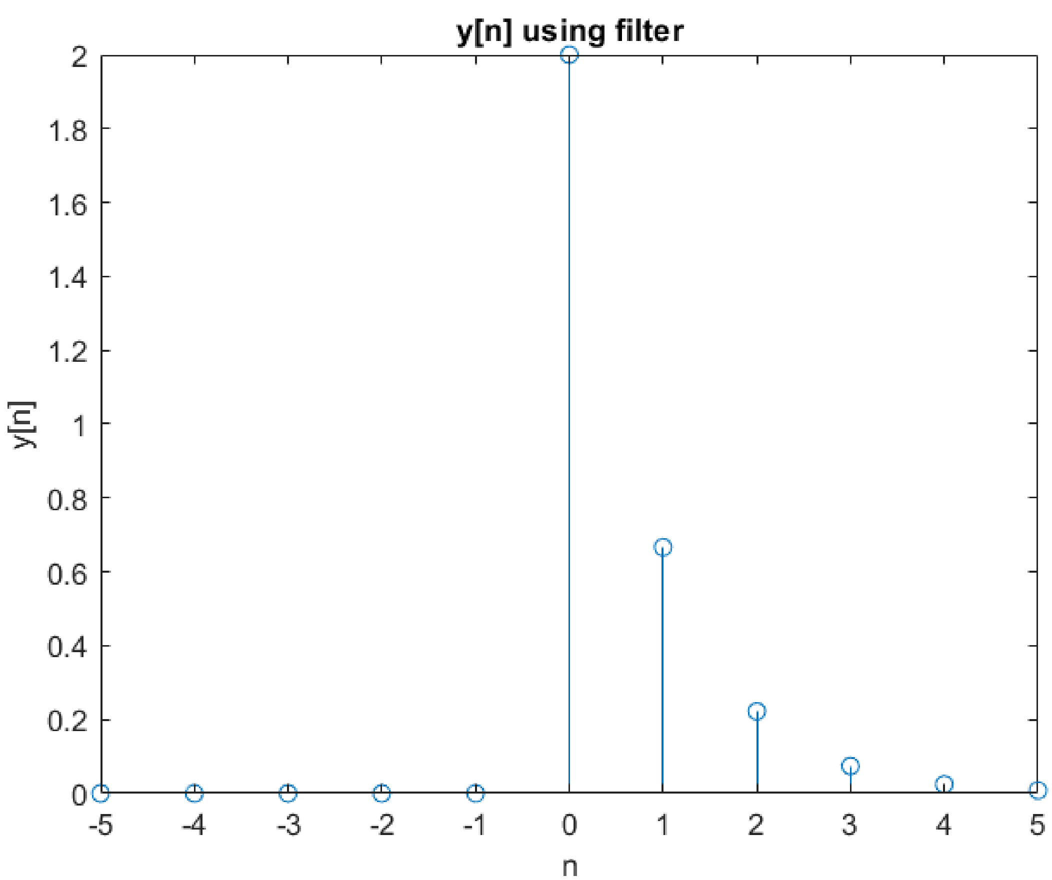
$$\Rightarrow h[n] = 2\left[\left(\tfrac{1}{3}\right)^n u[n] - \left(\tfrac{1}{3}\right)^{n-1} u[n-1]\right]$$

#

```
b = [2, -2];
a = [1, -1/3];
t = (-5:1:5)';
x = t >= 0;
y = filter(b, a, x)
```

```
y = 11×1
        0
        0
        0
        0
        0
   2.0000
   0.6667
   0.2222
   0.0741
   0.0247
     ⋮
```

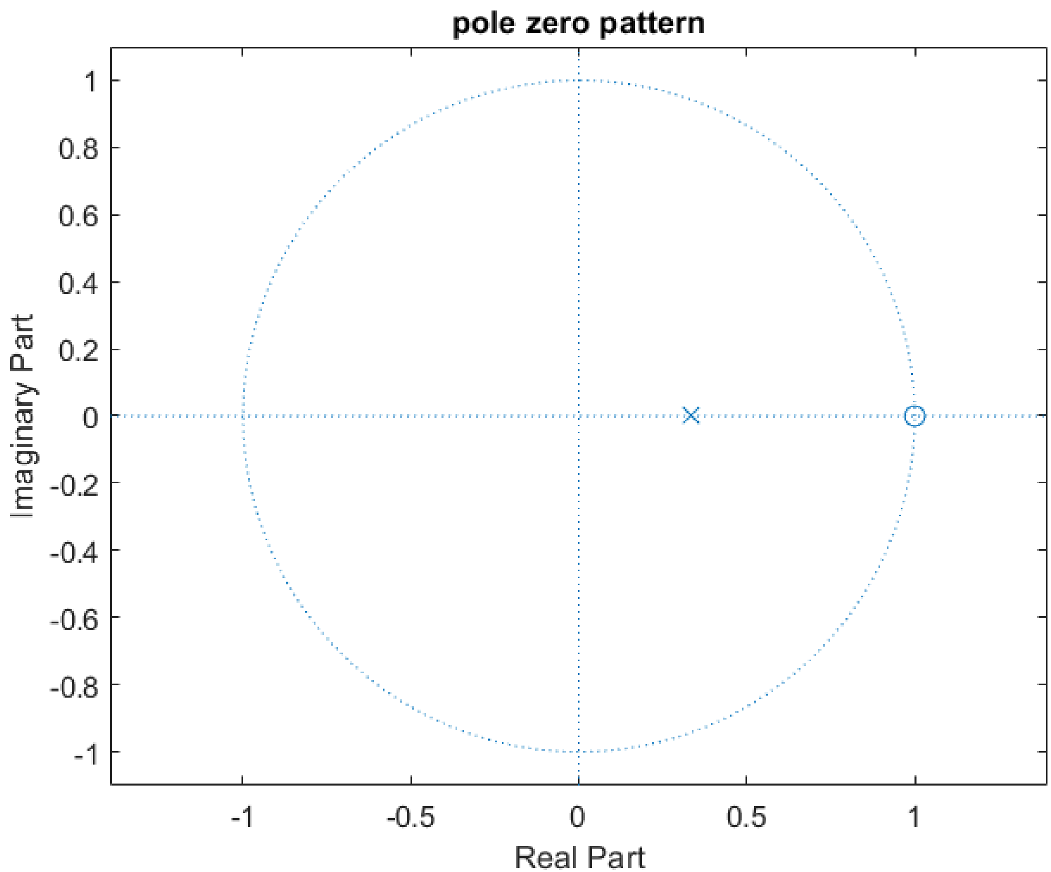
```
figure;
stem(-5:5, y);
title('y[n] using filter');
xlabel('n');
ylabel('y[n]');
```



As shown in the output sequence y and its plot, y[n] is equal to $2 \times \left(\frac{1}{3}\right)^n u[n]$

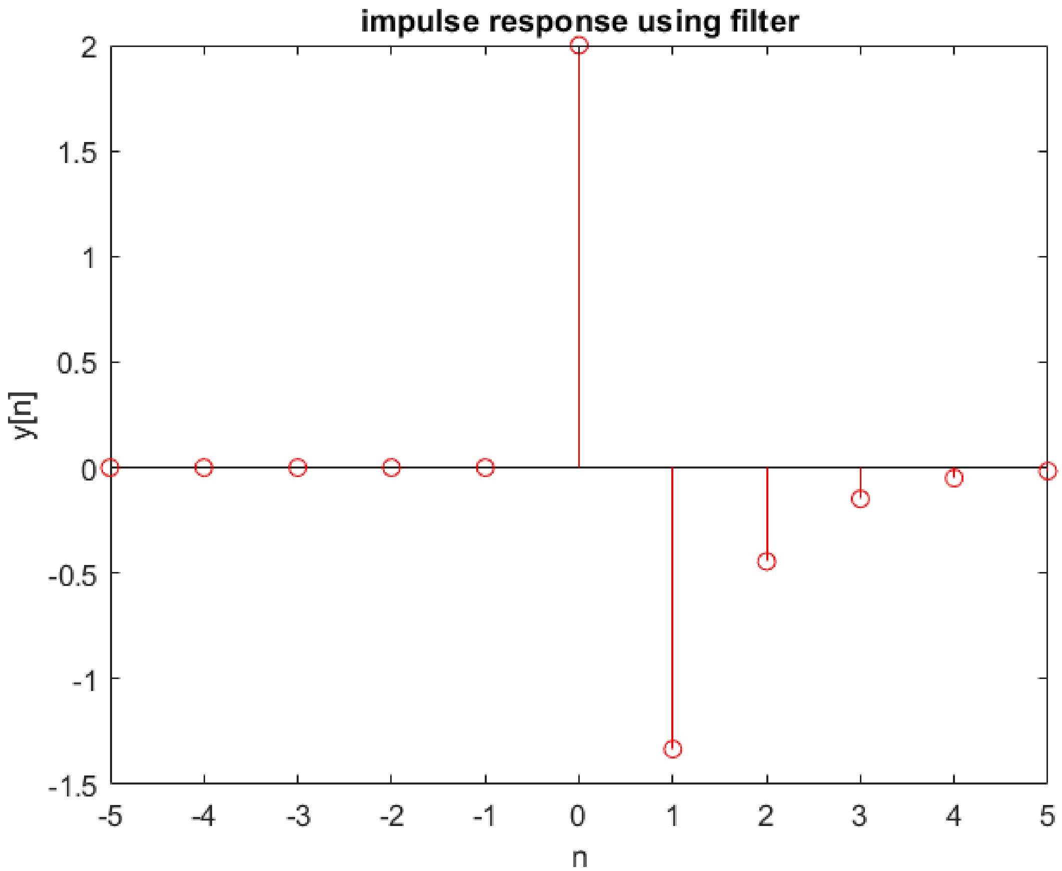
(b)Plot the pole-zero pattern using the function zplane(b, a)

```
subplot(1, 1, 1);
zplane(b, a)
title('pole zero pattern');
```

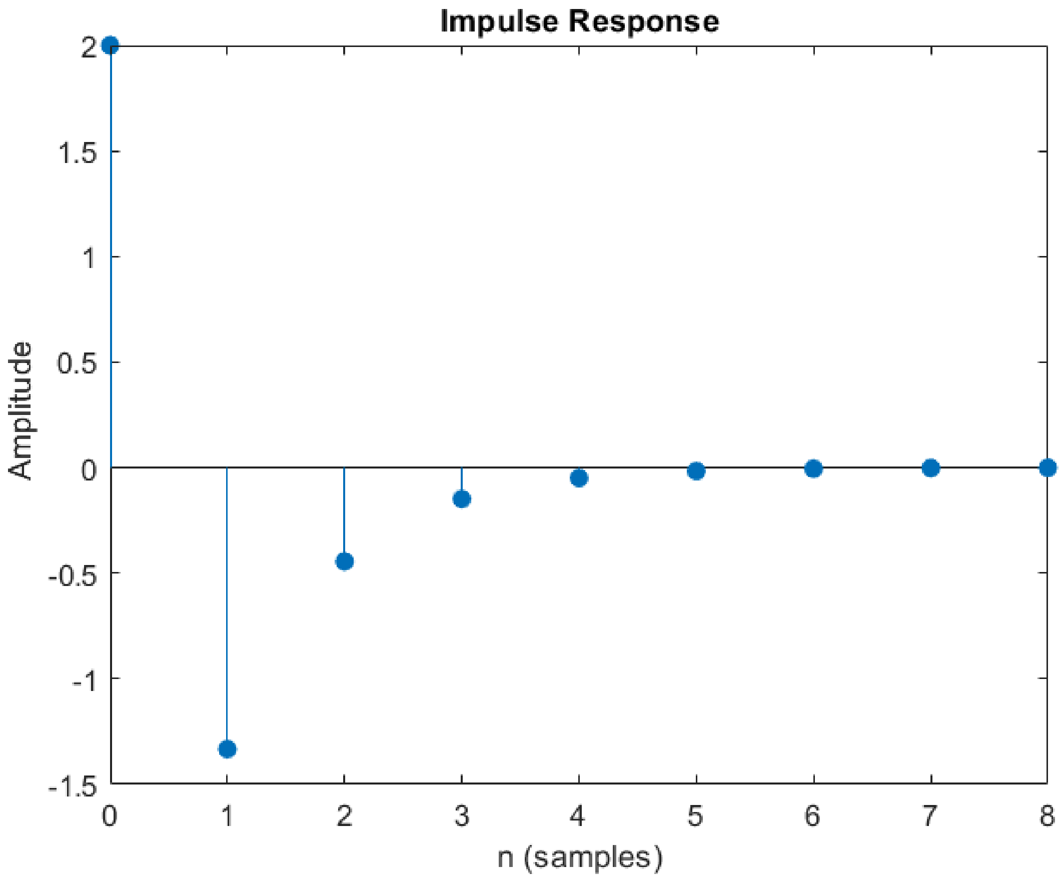


(c)Compute and plot the impulse response using the functions filter and stem. Compare with the plot obtained using the function impz

```
t = (-5:1:5)';
x = t == 0;   %delta function
y = filter(b, a, x);
stem(-5:1:5, y, 'r')
title("impulse response using filter");
xlabel('n');
ylabel('y[n]');
```


impulse response using filter
y[n]
n
2
1.5
1
0.5
0
-0.5
-1
-1.5
-5
-4
-3
-2
-1
0
1
2
3
4
5

```
impz(b, a);
```



Plot from filter and impz are the same as shown at the ouptut graph.

(d)Use the function residuez and the z-transform pairs in Table 3.1 to find an analytical expression for the impulse response h[n].

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

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

$$X(z) = \sum_{0}^{M-N} C_k z^{-k} + \sum_{0}^{N} \frac{A_k}{1 - P_k z^{-1}}$$

$$H(z) = 6 + \frac{-4}{1 - \frac{1}{3}z^{-1}}, \left|z\right| > \frac{1}{3} \left(\text{according to 3. } (a), \text{ ROC of } H(z) \text{ is } \left|z\right| > \frac{1}{3}\right)$$

$$\Longrightarrow h[n] = 6\delta[n] - 4\left(\frac{1}{3}\right)^n u[n]$$

4. 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

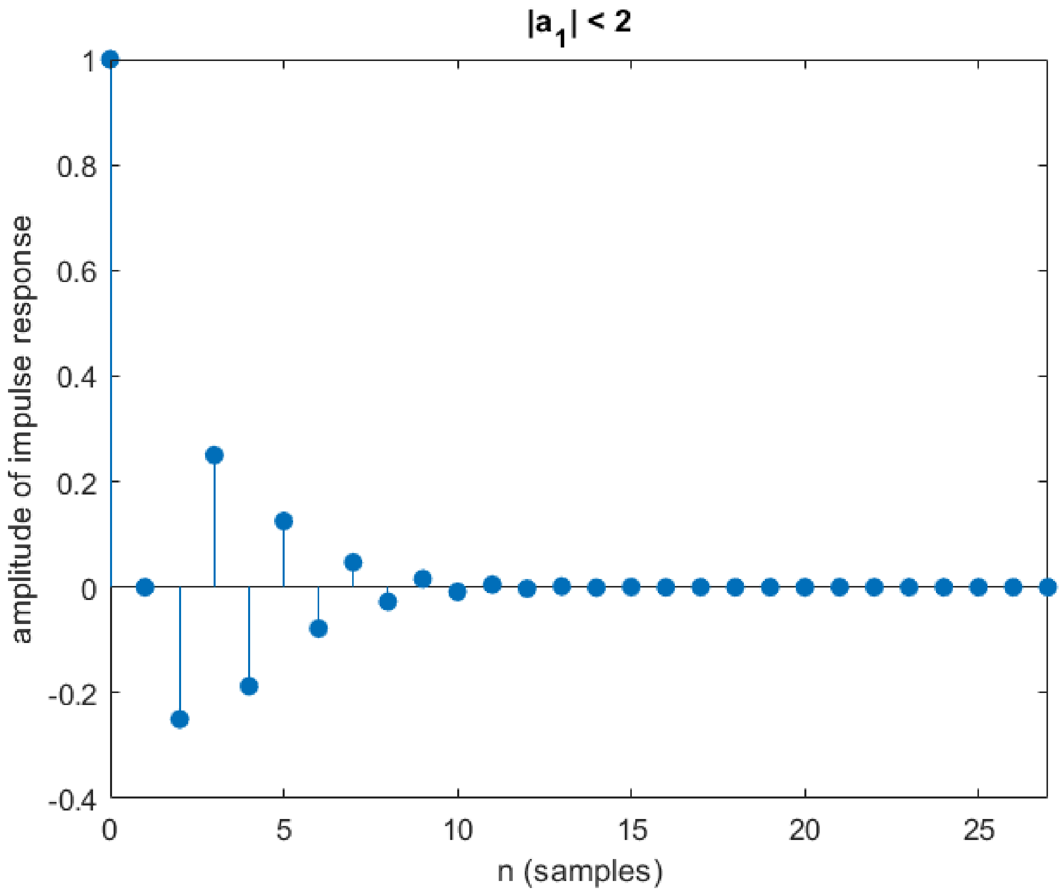
(a) the stability of the system

H(z) = $\frac{(b_0 + b_1z^{-1})}{1 + a_1z^{-1} + a_2z^{-2}} = \frac{z(b_0z + b_1)}{z^2 + a_1z + a_2}$. (poles = $\frac{(-a_1 \pm \sqrt{a_1^2 - 4a_2})}{2}$) To have equal and real poles, $a_1^2 = 4a_2$, then pole is z = $-\frac{a_1}{2}$. If the system stable, ROC has to include unit circle (i.e. |z| = 1). If pole is on unit circle $|a_1|$ = 2.

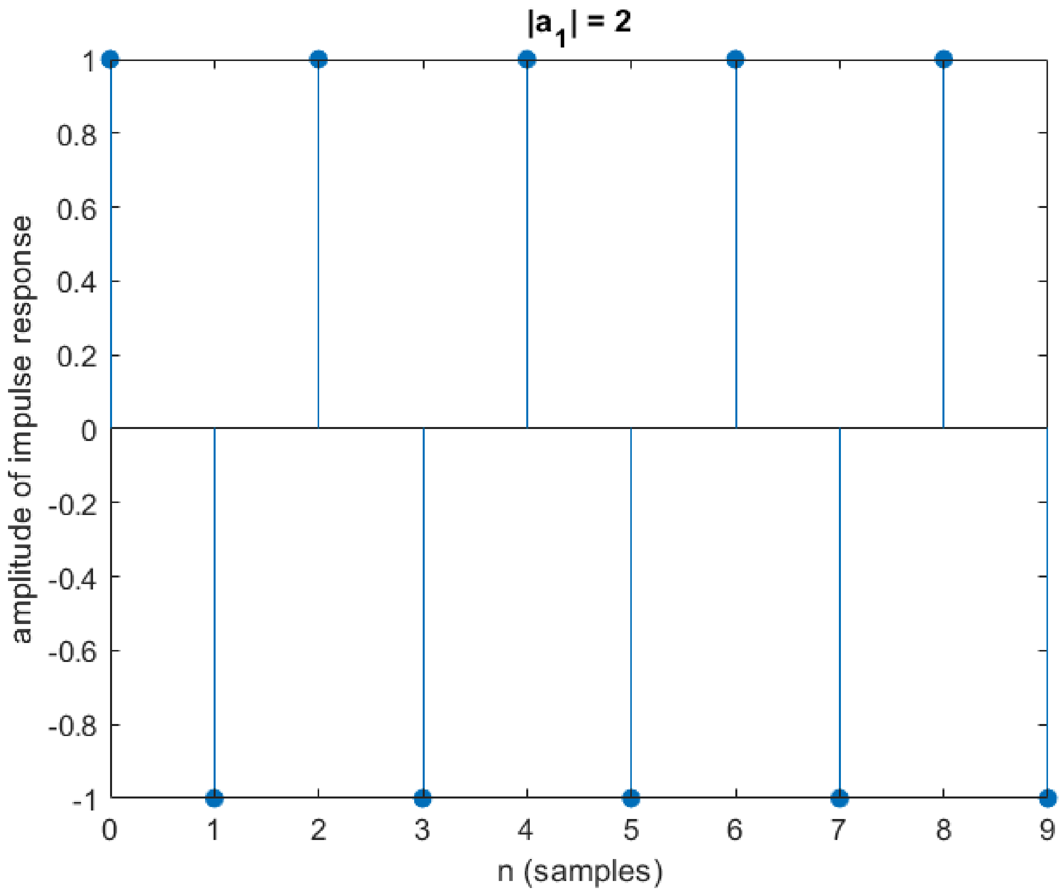
(b) the shape of the impulse response. (Hint: Use MATLAB to replicate Figure 3.10 for a double pole, find C and discuss stability of the three cases |a_1|<C, |a_1|= C, |a_1| > C)

If pole is on unit circle, $|a_1|$ = 2. Hence, we set **C = 2** to discuss the shape of impulse response when $|a_1| < 2, |a_1| = 2$ and $|a_1| > 2$ respectively. Also since position of zero will not affect the stability of the system, I can assume $b_0 = b_1 = 1$

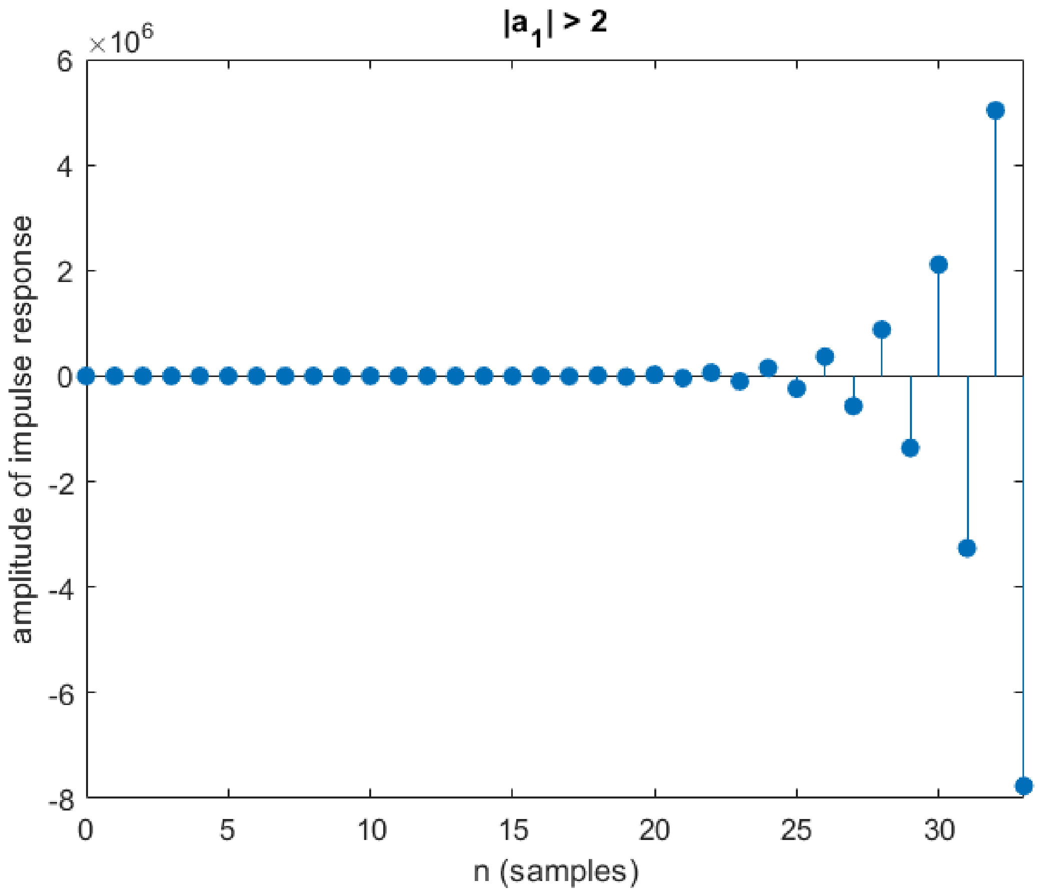
```
b = [1 1];
a1_case1 = [1 1 1/4];  % |a1| < 2
a1_case2 = [1 2 1];  % |a1| = 2
a1_case3 = [1 3 9/4]; % |a1| > 2
impz(b, a1_case1);
title('|a_{1}| < 2');
ylabel('amplitude of impulse response');
```




```
impz(b, a1_case2);
title('|a_{1}| = 2');
ylabel('amplitude of impulse response');
```




```
impz(b, a1_case3);
title('|a_{1}| > 2');
ylabel('amplitude of impulse response');
```



So accordign three different figures under different value of a_1, we can find only whne a_1 < 2, the system has the stable impulse response. (i.e. both poles has to be located inside the unit circle, the system would be stable.)

5. (10%) Consider the following LCCDE:

$$y[n] = 2cos(\omega_0)y[n-1] - y[n-2],$$

with no input but with initial conditions $y[-1] = 0$ and $y[-2] = -Asin(\omega_0)$.

(a) Show that the solution of the above LCCDE is given by the sequence

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

This system is known as a *digital oscillator* (derive in the live script).

$$y[n] = b_0 x[n] + b_1 x[n-1] + b_2 x[n-2] - a_1 y[n-1] - a_2 y[n-2]$$

$(a_1 = -2\cos w_0, \quad a_2 = 1)$

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

$\therefore$ In this problem, there is no $x[n]$, $\therefore b_0 = b_1 = b_2 = 0$, $x[-1] = x[-2] = 0$

$$Y^+(z) = \frac{A\sin(w_0)}{1 - 2\cos w_0 z^{-1} + z^{-2}} = \frac{z^2 A \sin w_0}{z^2 - 2\cos w_0 z + 1} = \frac{A \sin w_0}{(1 - z^{-1}e^{jw_0})(1 - z^{-1}e^{-jw_0})}$$

$$= \frac{m}{1 - z^{-1}e^{jw_0}} + \frac{n}{1 - z^{-1}e^{-jw_0}} \Rightarrow \begin{cases} m = \frac{-j}{2} A e^{jw_0} \\ n = \frac{j}{2} A e^{-jw_0} \end{cases}$$

$$\Rightarrow y[n] = \frac{-j}{2} A u[n] \left(e^{jw_0(n+1)} - e^{-jw_0(n+1)}\right)$$

$$= \frac{-j}{2} A u[n] \, 2j \sin[w_0(n+1)]$$

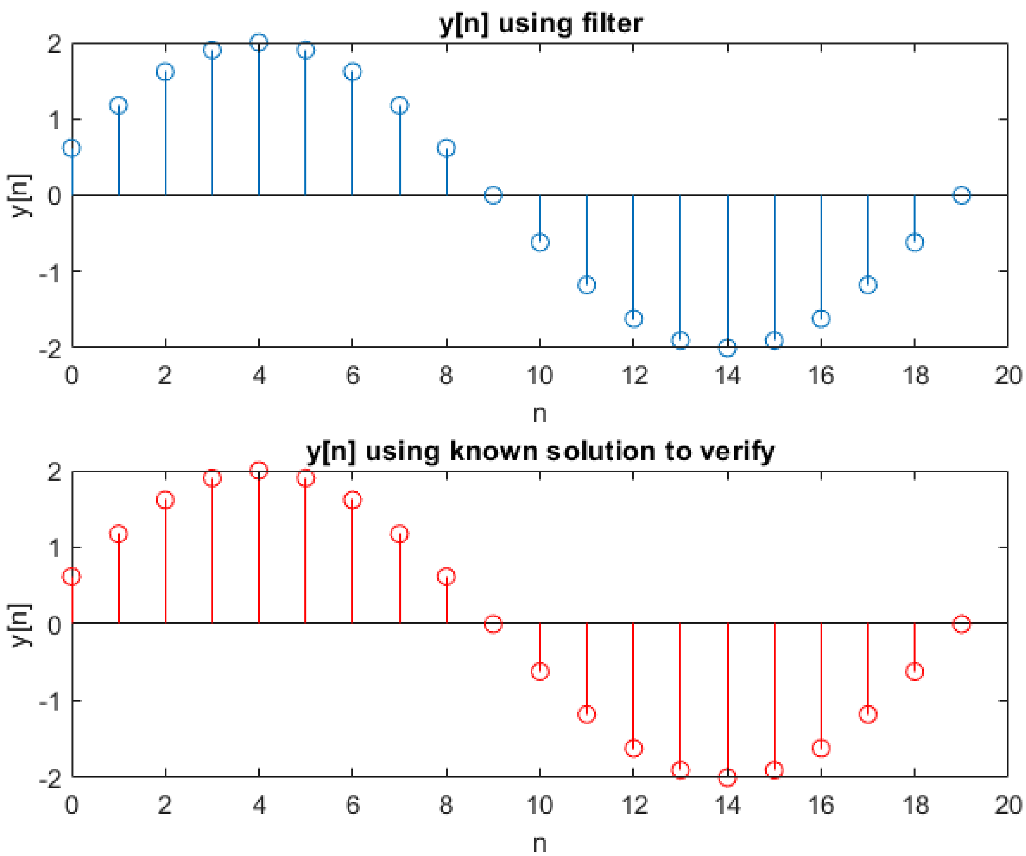
$$= A \sin[(n+1)w_0] u[n]$$

(b) For A = 2 and ω = 0.1π, verify the operation of the above digital oscillator using the filtc and the filter function. (Hint: check one-sized z-transform in supplement)

According to (a), y[n] = 2× $\sin[(n+1) \times 0.1 \times \pi]u[n] \Longrightarrow$ period $= 20$ $i.e.$ $N = 20$.
Therefore I input zeros function with length $= 20$ (zeros(1, 20)) to see 20 complete period of $y[n]$ as filter function will produce the output with the length of input signal.

```
A = 2;
w = 0.1*pi;
b = [0 0 0];
a1 = -2*cos(0.1*pi);
a2 = 1;
a = [1 a1 a2];
yic =  [0 -A*sin(0.1*pi)];
xic = [0 0];
zic = filtic(b, a, yic, xic);
```

```
y = filter(b, a, zeros(1, 20), zic);
subplot(2, 1, 1);
stem(0:19, y);
title('y[n] using filter');
xlabel('n');
ylabel('y[n]');
subplot(2, 1, 2);
n = 0:19;
y_ans = 2*sin(w*(n+1));
stem(n, y_ans, 'r');
title('y[n] using known solution to verify');
xlabel('n');
ylabel('y[n]');
```



According to the output grpah, signal using filter and signal using known solution are the same. Hence, the operation of part (a) is correct.

6. 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 and verify the result with dtfs function.

```
x = [5, 13, 1, -9, 32];
```

```
dtfs0(x)
```

```
ans = 1×5 complex
   8.4000 + 0.0000i   5.0756 + 2.4384i  -6.7756 + 4.1357i  -6.7756 - 4.1357i ⋯
```

```
dtfs(x)
```

```
ans = 1×5 complex
   8.4000 + 0.0000i   5.0756 + 2.4384i  -6.7756 + 4.1357i  -6.7756 - 4.1357i ⋯
```

```
round(dtfs0(x), 5) == round(dtfs(x), 5)
```

```
ans = 1×5 logical array
   1   1   1   1   1
```

```
% subplot(2, 1, 1);
% stem(dtfs0(x));
% title('dtfs0');
% xlabel('k');
% ylabel('C_{k}')
% subplot(2, 1, 2);
% stem(dtfs(x));
% title('dtfs');
% xlabel('k');
% ylabel('C_{k}')
```

觀察兩者四捨五入至小數點後第5位的結果，dtfs & dtfs0 皆產生相同的結果 (Values of logical array are all equal to 1)。

(b) Write a function x=idtfs0(c) which computes the inverse DTFS (4.63) and verify the result with idtfs function

```
x = [5, 13, 1, -9, 32];
c = [1, 2, 3, 4, 5];
round(idtfs0(c), 5) == round(idtfs(c), 5)
```

```
ans = 1×5 logical array
   1   1   1   1   1
```

觀察兩者四捨五入至小數點後第5位的結果，idtfs & idtfs0 皆產生相同的結果 (Values of logical array are all equal to 1)。

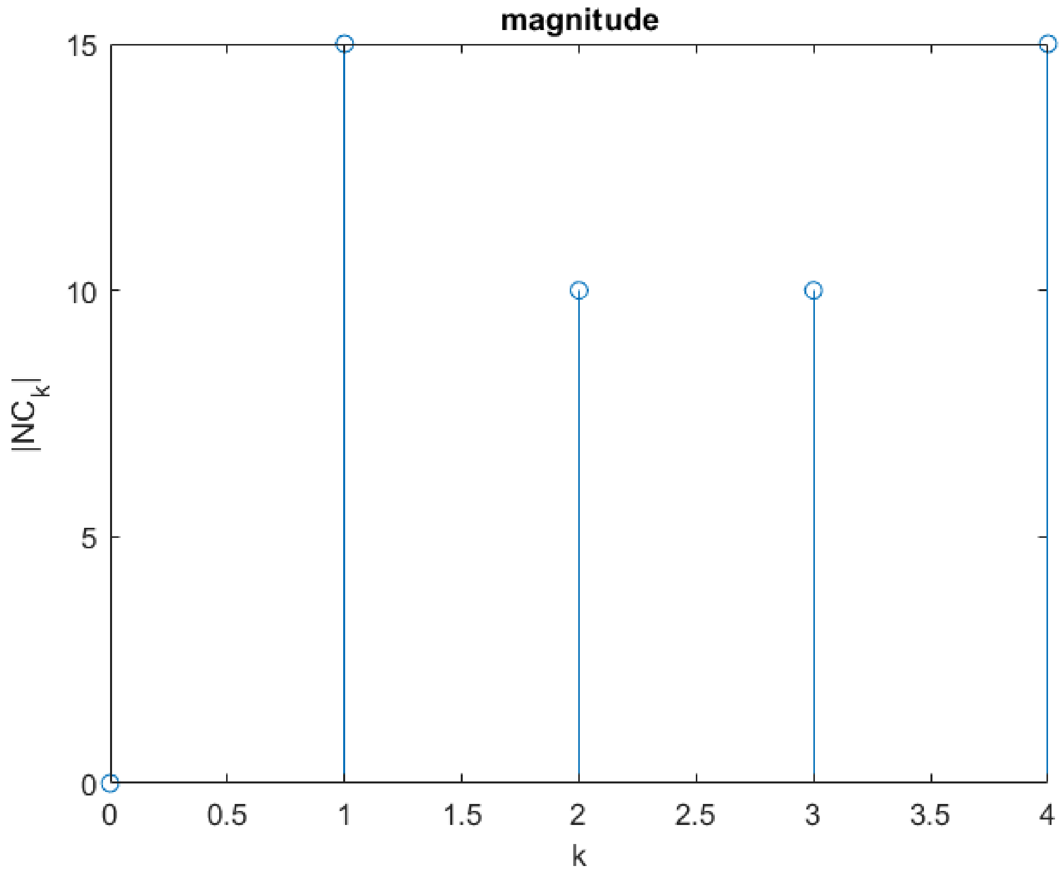
7. Determine and plot the magnitude and phase spectra of the following periodic sequences:

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

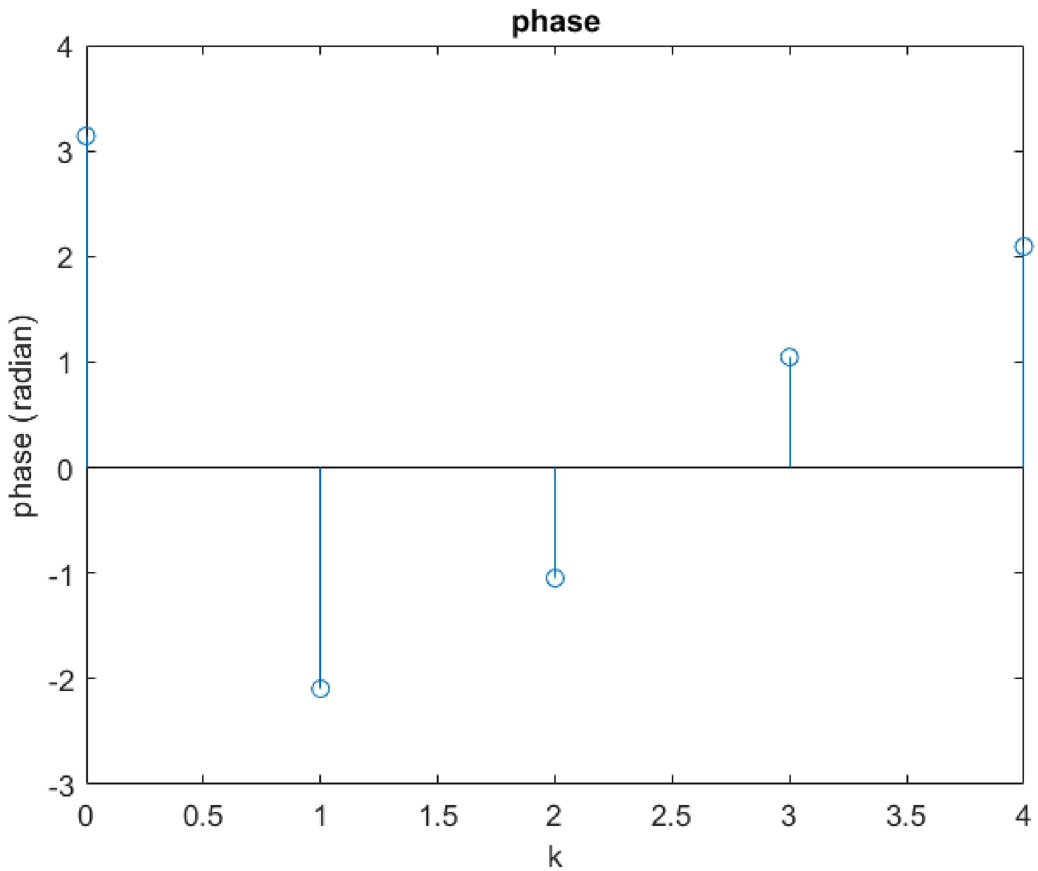
Ans: period of cos(1.2πn+$\frac{\pi}{3}$) and sin(0.4πn$-\frac{\pi}{6}$) are both equal to 5. So period of $x_1[n]$ is 5, i.e. N = 5.

$NC_k = \sum_0^{N-1} x[n]e^{-j2\frac{\pi}{N}kn}$ I got Fourier series coefficient C_k by calling **dfts** function. And x-axis is k, y-axis is $|NC_k|$. Also, $\omega_k \left(= 2\frac{\pi}{N}k\right)$ is the corresponded angular frequency at index $= k$, so $x-$axis can also be represented by ω_k as well. I use the same notation through 7 (a), (b), (c).

```
n = 0:4;
x1 = 4*cos(1.2*pi*n+pi/3)+6*sin(0.4*pi*n-pi/6);
c = dtfs(x1);
N = 5;
mag = N*abs(c);
figure;
stem(n, mag);
title('magnitude');
xlabel('k');
ylabel('|NC_{k}|');
```



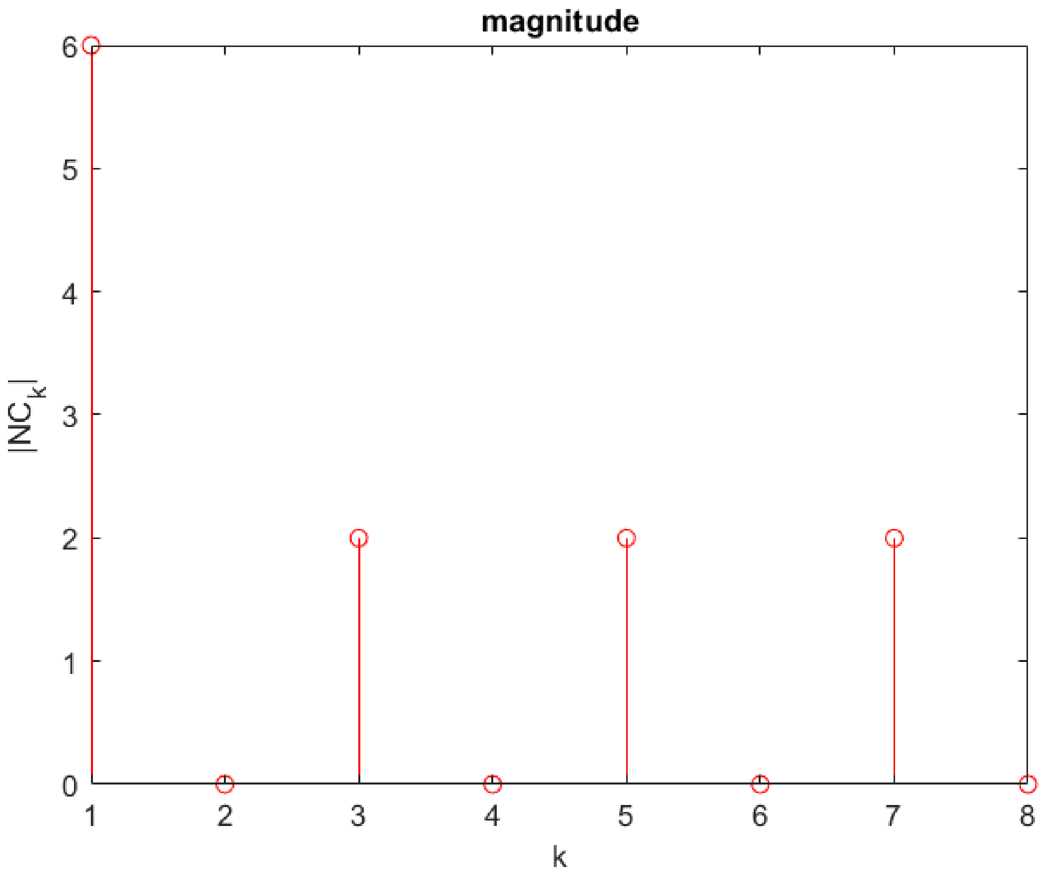

```
phase = angle(c);
stem(n, phase);
title('phase');
xlabel('k');
ylabel('phase (radian)');
```



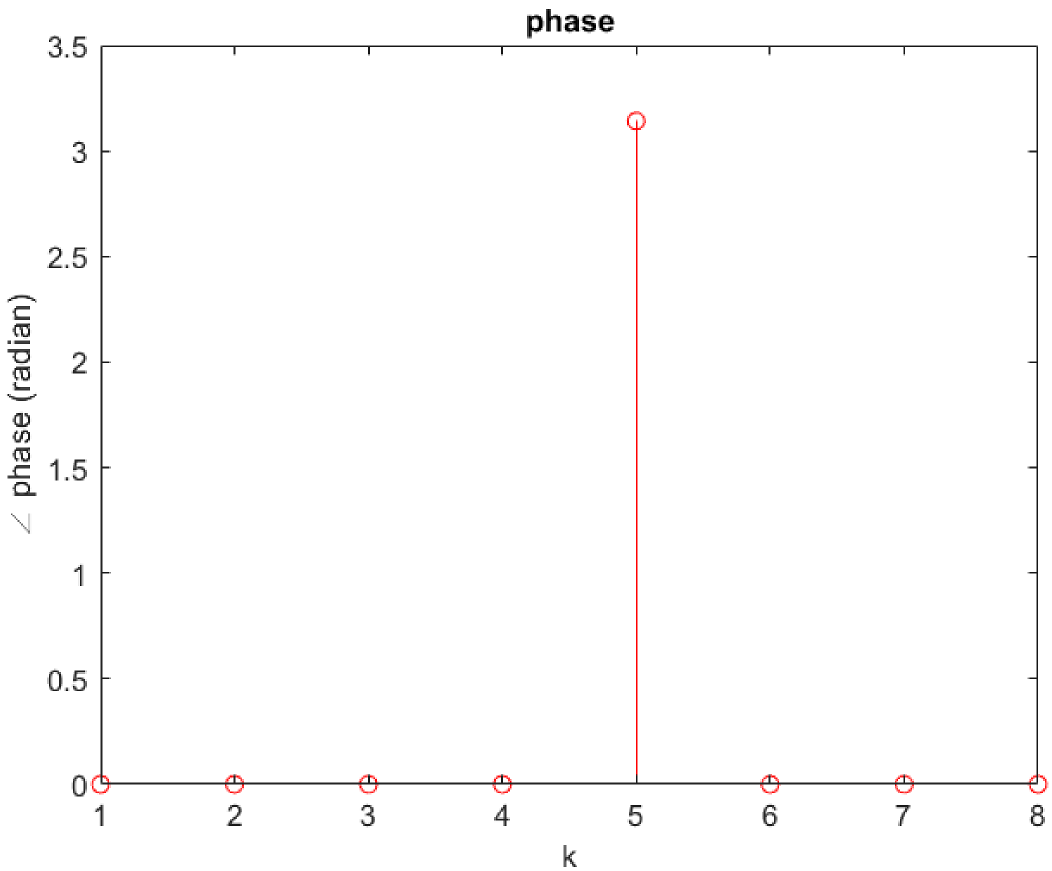
(b) x_2**[n]={1,1,0,1,1,1,0,1}, (one period)**

period = N = 8

```
N = 8;
x2 = [1, 1, 0, 1, 1, 1, 0, 1];
c =dtfs(x2);
mag = N*abs(c);
figure;
stem(mag, 'r');
title('magnitude');
xlabel('k');
ylabel('|NC_{k}|');
```



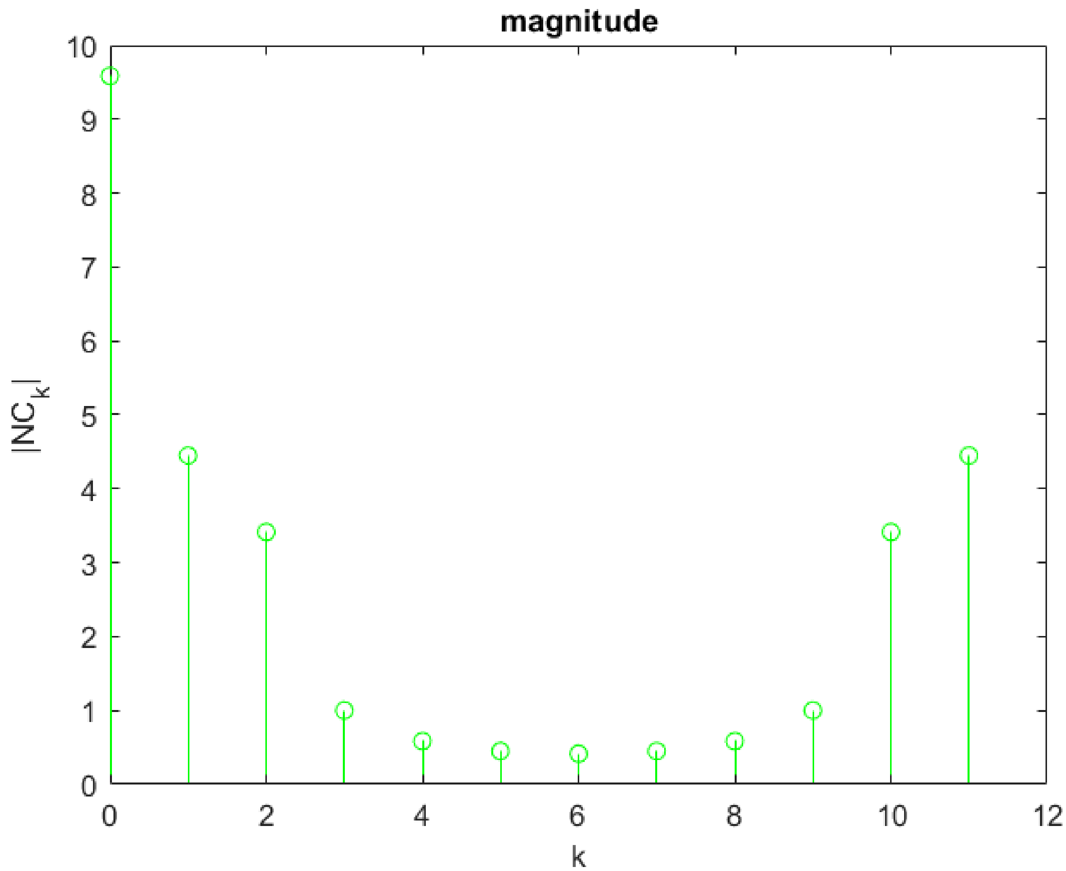

```
phase = angle(c);
stem(phase, 'r');
title('phase');
xlabel('k');
ylabel('\angle phase (radian)');
```



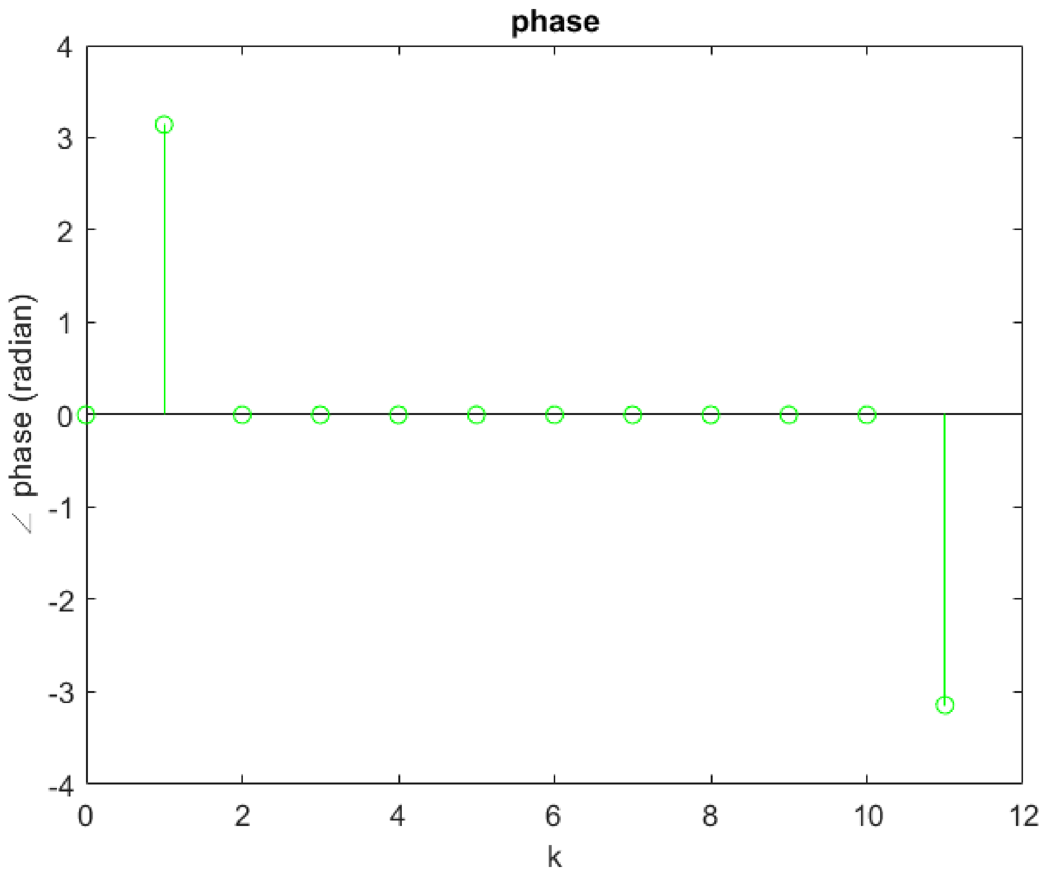
(c) $x_3[n]=1-\sin(\frac{\pi}{4}n)$, $0\leq n\leq 11$ **(one period)**

period = N = 8.

```
n = 0:11;
N = 12;
x3 = 1-sin(pi/4*n);
c =dtfs(x3);
mag = N*abs(c);
figure;
stem(n, mag, 'g');
title('magnitude');
xlabel('k');
ylabel('|NC_{k}|');
```




```
stem(n, angle(c), 'g');
title('phase');
xlabel('k');
ylabel('\angle phase (radian)');
```



8. (8%) Consider a noncausal finite length sequence $x[n] = \delta[n+1] + \delta[n] + \delta[n-1]$, we can compute the DTFT in MATLAB using following scripts:

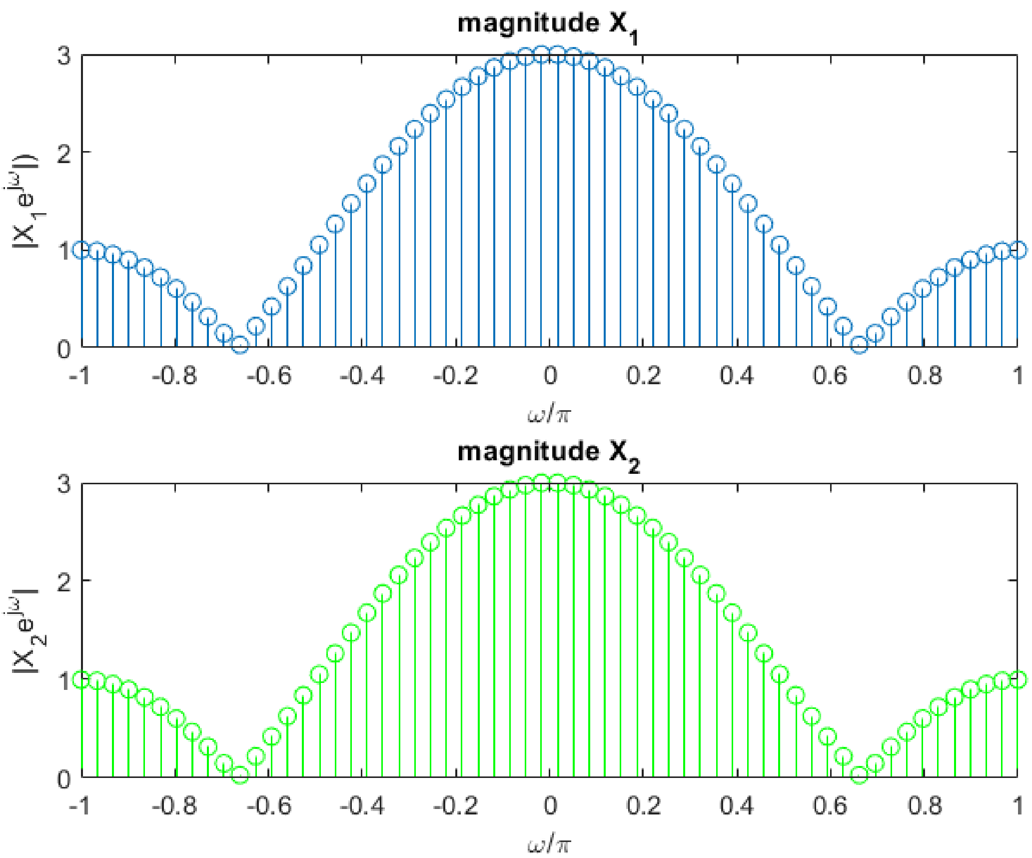
x=[1 1 1]; % n=-1,0,1

om=linspace(-pi, pi, 60);

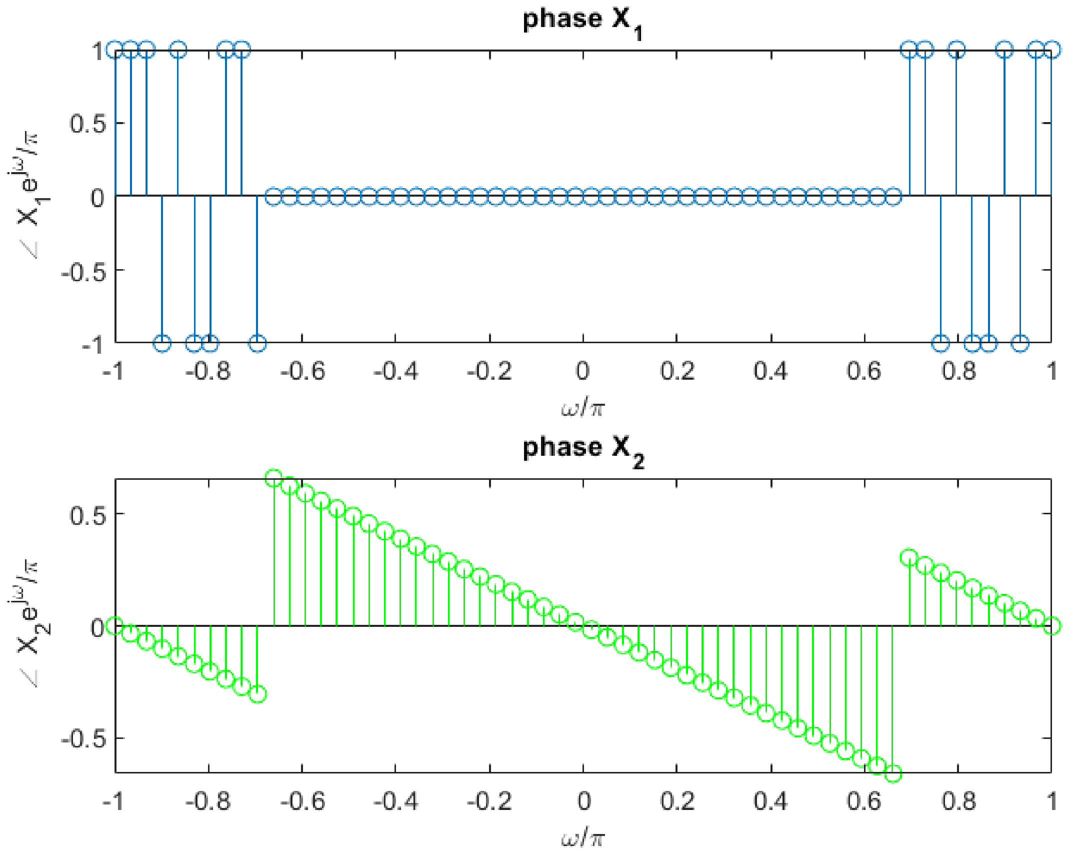
X1=dtft12(x, -1, om); X2=freqz(x, 1, om);

(a) Use subplot to plot the magnitude $|X1|, |X2|$ and phase $\angle X1, \angle X2$ in one graph respectively.

```
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);
stem(om/pi, abs(X1));
title('magnitude X_{1}');
xlabel('\omega/\pi');
ylabel('|X_{1}e^{j\omega}|)');
subplot(2,1,2);
stem(om/pi, abs(X2), 'g');
title('magnitude X_{2}');
xlabel('\omega/\pi');
ylabel('|X_{2}e^{j\omega}|');
```




```
figure;
subplot(2,1,1);
stem(om/pi, angle(X1)/pi);
title('phase X_{1}');
xlabel('\omega/\pi');
ylabel('\angle X_{1}e^{j\omega}/\pi');
subplot(2,1,2);
stem(om/pi, angle(X2)/pi, 'g');
title('phase X_{2}');
xlabel('\omega/\pi');
ylabel('\angle X_{2}e^{j\omega}/\pi');
```



(b) Is the magnitude |X1|=|X2| ? Is the phase $\angle X1 = \angle X2$? If not, Explain.

Ans: the magnitude is the same, but the phase it different. Fourier transform is

X $\left(e^{jw_k}\right) = \sum_{N_1}^{N_2} x[n] e^{-jw_k n} = e^{-jw_k N_1} \sum_{0}^{N_2-N_1} x[n+N_1] e^{-jw_k n}$ **freqz** always assumes $N_1 = 0$. Therefore,

when $N_1 \neq 0$, for each frequency component ω_k, its phase will be affected by $e^{-jw_k N_1}$, but as $| e^{-jw_k N_1} |$ = 1, magnitude is not affected (both of them have the same magnitude response).

9. (12%) In this problem use the image file “DSP.png” which has 100×300 pixels (unsigned 8 bits per pixel). You can use the following MATLAB script to load, show and store the image files:

```
img = imread('DSP.png');
imshow(img);
imwrite(img, 'DSP0.png');
```

(a) Consider the 5×5 impulse response $h[m, n]$ given as follow:

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

filter the “DSP.png” using (2.78) and display the resulting image, comment on the result. (Hint: You can directly use conv2 function, and make sure the data format is double before filtering).

```
img = imread('DSP.png');
figure;
disp('Original');
```

```
Original
```

```
imshow(img);
```


```
h = (1/25)*ones(5, 5);
img = double(img);
img = conv2(h, img);
%imwrite(img, 'DSP_a.png');
img = uint8(img);
disp('After filtering');
```

```
After filtering
```

```
imshow(img);
```

Comment: h[m, n] is a average filter, it will average 25 pixels value around each pixel of the original photo and create the filterd photo. After filtering, there appears black border. it's because when doing filtering on the pixels near the borders, h[m, n] covers the black word (DSP), after averaging, there appears black border around the border. Also the word (DSP becomes more soft), it's because h[m, n] covers white part of the photo, after averaging, the word beomes less black and a little blur as well.

(b) Repeat part (a) and try two different kernels $h_1[m,n]$ and $h_2[m,n]$: (known as Sobel filter)

$$h_1[m,n] = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad h_2[m,n] = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},$$

what is the difference between the two results? (Hint: You should do abs and uint8 before imshow)

```
h1 = [1 0 -1; 2 0 -2; 1 0 -1];
h2 = [1 2 1; 0 0 0; -1 -2 -1];
img = imread('DSP.png');
img_double = double(img);
img_h1 = conv2(h1, img_double);
img_h1 = abs(img_h1);
img_h1 = uint8(img_h1);
%imwrite(img_h1, 'DSP_h1.png');
figure;
disp('h1 filter');
```

```
h1 filter
```

```
imshow(img_h1);
```

```
img_h2 = conv2(h2, img_double);
img_h2 = abs(img_h2);
img_h2 = uint8(img_h2);
%imwrite(img_h2, 'DSP_h2.png');
disp('h2 filter');
```

```
h2 filter
```

```
imshow(img_h2);
```

Difference between the two results: Sobel filter is used for edge detection, where the pixel value change sharply. In this problem, there are only two colors in the picture. (black and white) Hence, the edges are located at the pixels between black and white parts of image. Sobel filter has two directions, horizontal and vertical. h_1 is the horizontal one. and h_2 is the vertical one. Horizontal sobel filter will detect the edge in the horizontal direction. As shown at the the result from h_1 filter, edge on the horizontal has been highlighted in white. No sharp change of pixel value in the horizontal direction is black, such as upper and lower part of word (DSP) and white all black and white part of image except edge. As the result from h_2 filter shows, the edge on the vertical part has been hightlight in white. No sharp change of pixel

value in vertical direction is black, such as left part of 'D' and 'p'. I noted that as 'S' has sharp pixel value change in both direction except upper and lower part of 'S'. Therefore, the border of 'S' is almost all highlighted at two pictures.

(c) Repeat (b) by using filter2 function, what is the difference between the two functions? (Hint: Check the pixel values before abs)

```
img_h1 = filter2(h1, img_double);
img_h1 = abs(img_h1);
img_h1 = uint8(img_h1);
%imwrite(img_h1, 'DSP_h1.png');
figure;
imshow(img_h1);
```

```
img_h2 = filter2(h2, img_double);
img_h2 = abs(img_h2);
img_h2 = uint8(img_h2);
%imwrite(img_h2, 'DSP_h2.png');
imshow(img_h2);
```

Difference between the two functions: `conv2(image, h) is the same as filter2(rot90(h,2), image). rot90(h, 2) means rotate h (kernel) 180° counterclockwise. In this two problem,` h_1 `is still horizontal edge detection filter and` h_2 is still vertical edge detection filter. The only difference is that before taking absolute value, (result from conv2) = -(result from filter2). So after taking absolute value, the iamge matrix value is the same, that's why photos from (b) and (c) are the same.

10. (12%) This problem uses the sound file "handel.wav" available in MATLAB. This sound is sampled at Fs = 8192 samples per second using 8-bits per sample. You can use the following MATLAB script to load, play and store the audio files:

[y,Fs] = audioread('handel.wav');
playerObj = audioplayer(y, Fs);
play(playerObj);
audiowrite('handel0.wav', y, Fs);

(a) Select every other sample in audio signal y 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. (Hint: You can use the logical array to index array elements)

```
%Fs = 8192;
[y, Fs] = audioread('handel.wav');
playerObj = audioplayer(y, Fs);
play(playerObj);
```

```
% x = [1 0; 0 1];
% n = floor(length(y)/4); % [1 0] repeat times
% B = repmat(x,n,1);
% B = logical(B);
% y_a = y(B);
y_a = y(1:2:length(y));
playerObj = audioplayer(y_a, Fs/2);
play(playerObj);
% audiowrite('handel.wav', y, Fs);
```

(b) Select every fourth sample in audio signal y 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.

```
% a = 1;
% for n=1:length(y)
```

```
%     if mod(n, 4) == 0
%         y_b(a) = y(n);
%         a = a + 1;
%     end
% end
y_b = y(1:4:length(y));
playerObj = audioplayer(y_b, Fs/4);
play(playerObj);
```

(c) From the results of (a) and (b), what do you discover?

Ans: They are all low-pitched compared with original sound and the pitch from (b) is lower than (a). That's because their frequency has decreased throung downsampling. And (b)'s sampling rate is down by a factor of four. (a)'s sampling rate is down by a factor of two. Hence, (b) sounds more low-pitched.