DIGITAL SIGNAL PROCESSING

Venkata Karthikeya

June 27, 2020

Contents

1 To Plot the magnitude by varing r, w.

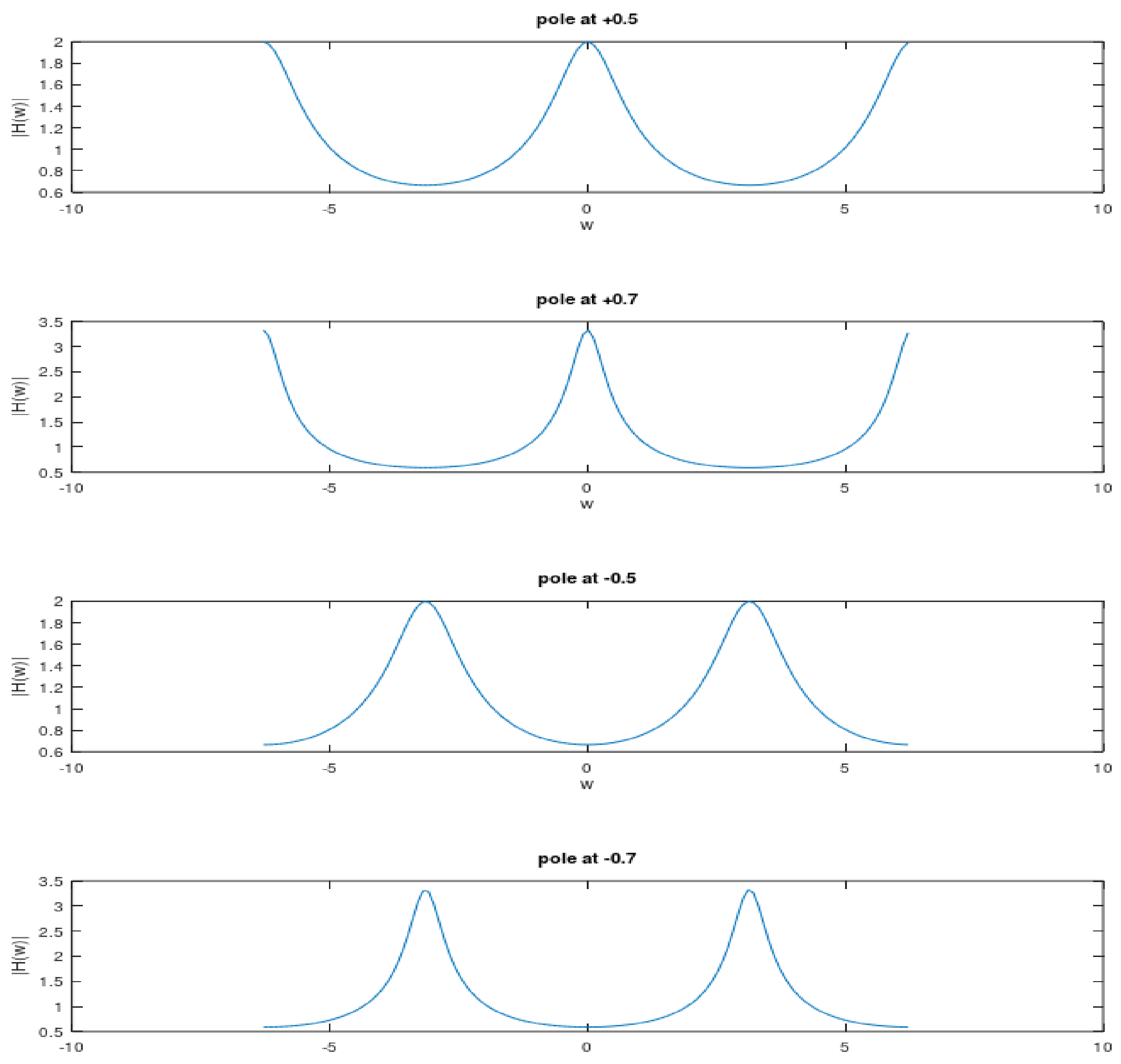
AIM: a)varing r keeping w constant. b)varing w keeping r constant.

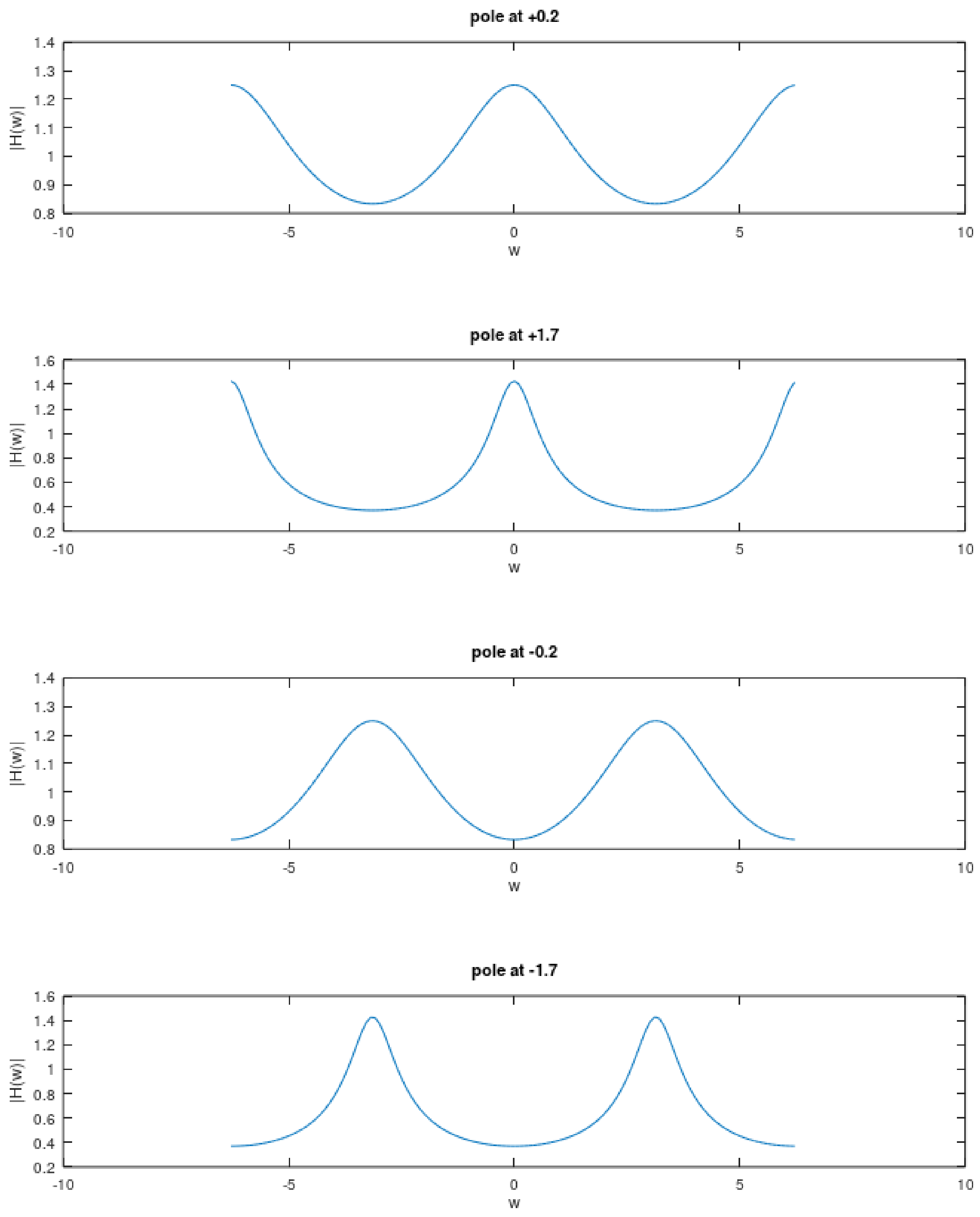
APPARATUS: octave software.

CODE AND GRAPHS:

```
clc;close;clear all
w=-2*pi:0.1:2*pi;
h1=1./(exp(j.*w)-0.5);
h2=1./(exp(j.*w)-0.2);
h3=1./(exp(j.*w)-0.7);
h4=1./(exp(j.*w)-1.7);
subplot(4,2,1);
plot(w,abs(h1))
title("pole at +0.5");
xlabel("w");
ylabel("|H(w)|");

subplot(4,2,2);
plot(w,abs(h2))
title("pole at +0.2");
xlabel("w");
ylabel("|H(w)|");

subplot(4,2,3);
plot(w,abs(h3))
title("pole at +0.7");
xlabel("w");
ylabel("|H(w)|");

subplot(4,2,4);
plot(w,abs(h4))
title("pole at +1.7");
xlabel("w");
ylabel("|H(w)|");

h5=1./(exp(j.*0)+0.5);
h6=1./(exp(j.*pi)+0.5);
h7=1./(exp(j.*w)+0.7);
h8=1./(exp(j.*w)+1.7);
subplot(4,2,5);
plot(w,abs(h5))
title("pole at -0.5");
xlabel("w");
ylabel("|H(w)|");

```

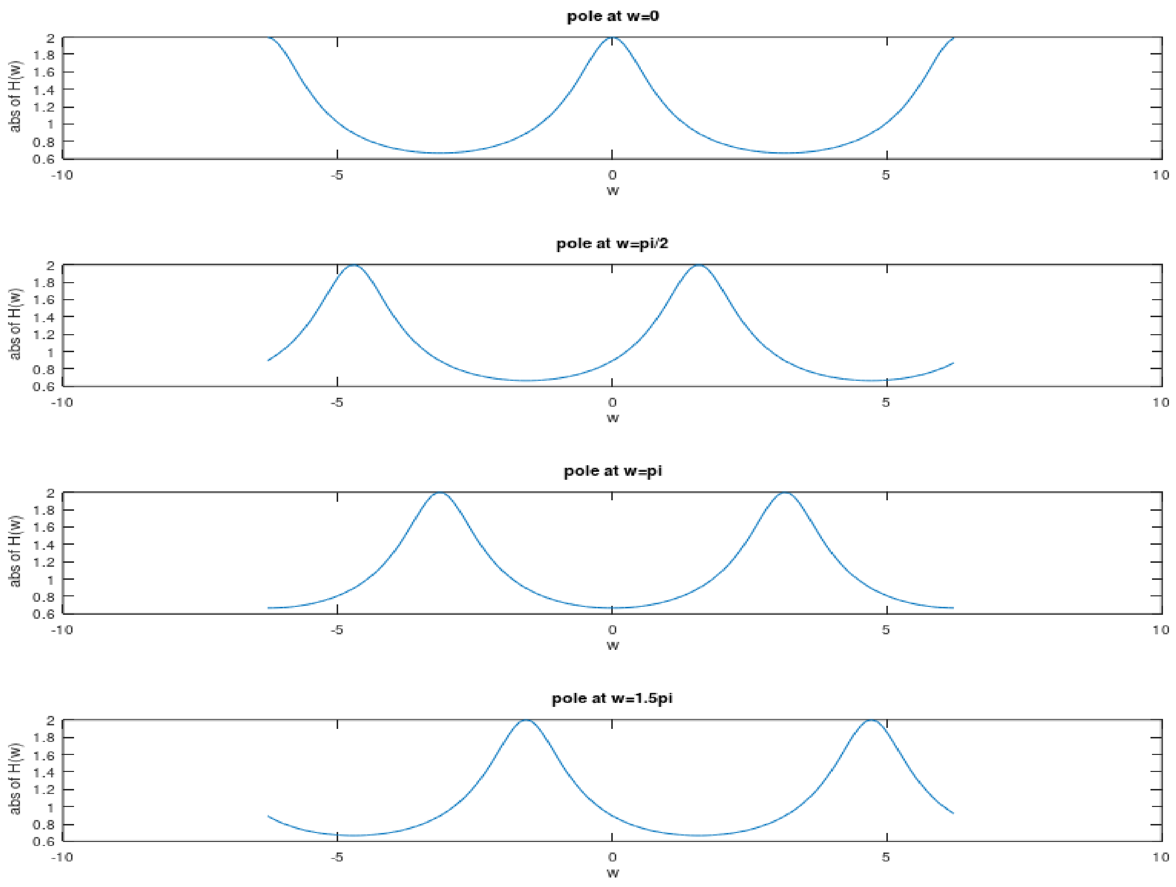
```
subplot(4,2,6);
plot(w,abs(h6))
title("pole at -0.2");
xlabel("w");
ylabel("|H(w)|");

subplot(4,2,7);
plot(w,abs(h7))
title("pole at -0.7");
xlabel("w");
ylabel("|H(w)|");

subplot(4,2,8);
plot(w,abs(h8))
title("pole at -1.7");
xlabel("w");
ylabel("|H(w)|");
```





```
clc;close;clear all
w=-2*pi:0.1:2*pi;

z=exp(j*w);

h1=1./(z-0.5*exp(j*0));
h2=1./(z-0.5*exp(j*(pi/2)));
h3=1./(z-0.5*exp(j*pi));
h4=1./(z-0.5*exp(j*(1.5*pi)));
```

```

subplot(4,1,1);
plot(w,abs(h1))
title("pole at w=0");
xlabel("w");
ylabel(" abs of H(w)");

subplot(4,1,2);
plot(w,abs(h2))
title("pole at w=pi/2");
xlabel("w");
ylabel(" abs of H(w)");

subplot(4,1,3);
plot(w,abs(h3))
title("pole at w=pi");
xlabel("w");
ylabel(" abs of H(w)");
subplot(4,1,4);
plot(w,abs(h4))
title("pole at w=1.5pi");
xlabel("w");
ylabel(" abs of H(w)");
```



CONCLUSION:

w is constant, varying r:

- The magnitude of the curve increases while moving towards unit circle and at unit circle the magnitude shots to infinite.
- The width of the curve is more for pole at less than 1,width of the curve is less for pole at more than 1.
- For poles which lie on + ve $Re(z)$ we get peak at $w = 0$;for poles lie on -ve $Re(z)$ we get peak at $w = -pi$ and pi.

r is constant, varying w:

- we get the magnitude same but the position of peak is present at corresponding value of w
- These peaks repeat at a peroid of $2pi$.

2 To make the comb filter , and plot the magnitude ,pole-zero plot.

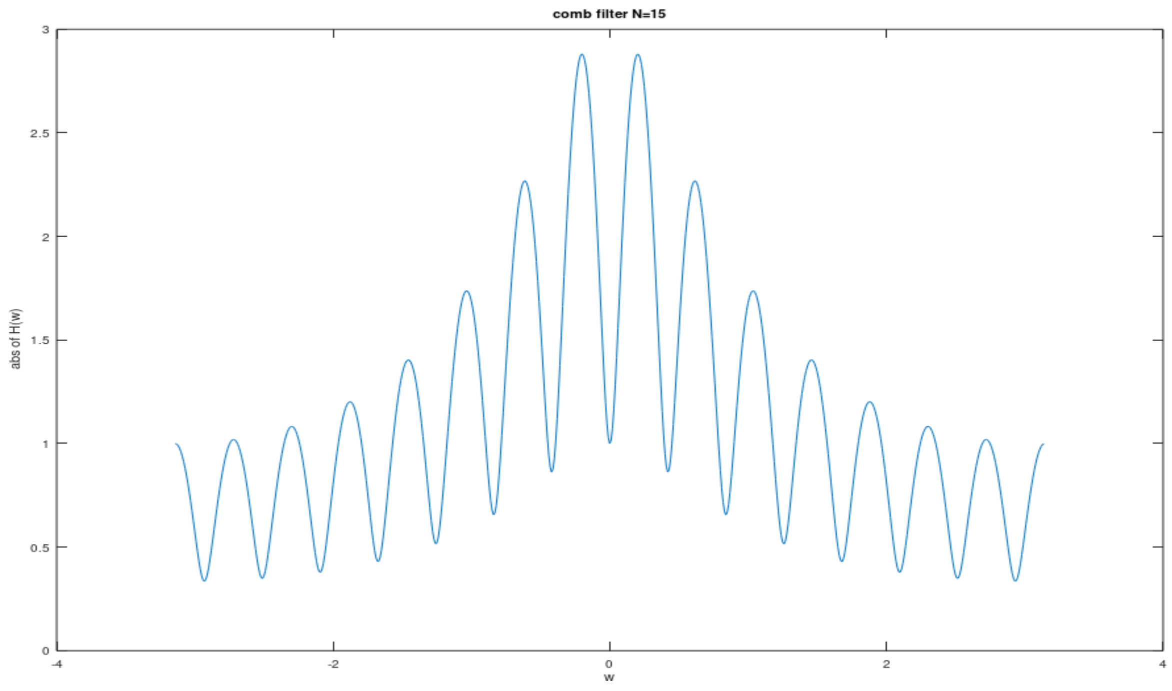
AIM: To make the comb filter , and plot the magnitude ,pole-zero plot.

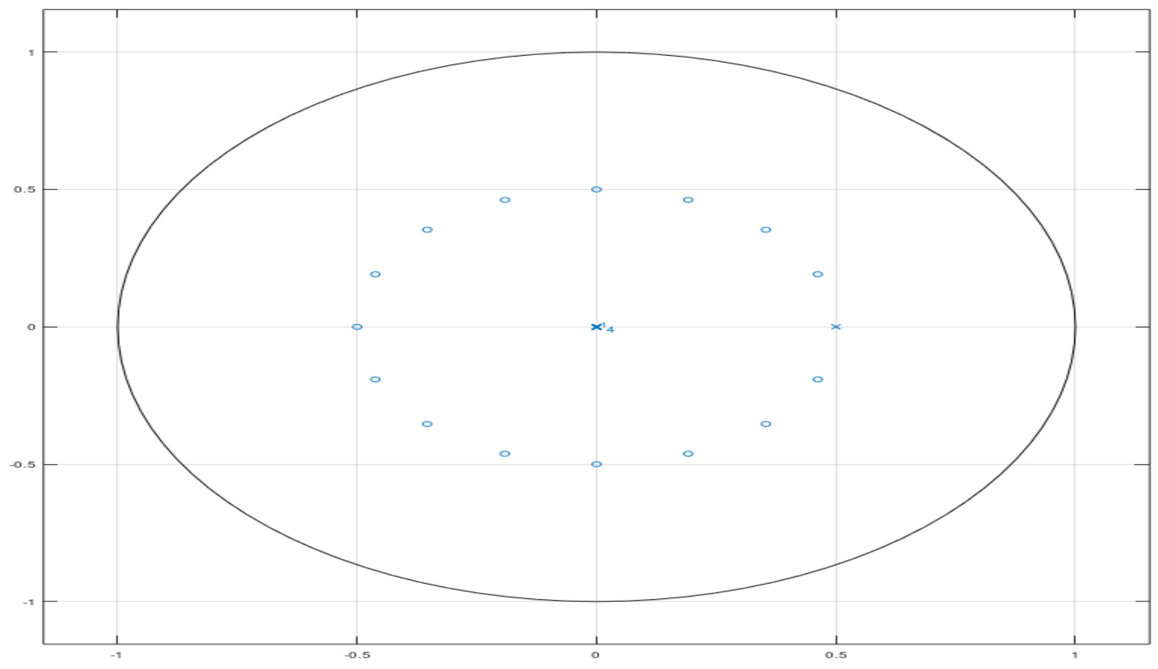
APPARATUS: octave software.

CODE:

```
clc;close all;clear all;
w=-pi:0.01:pi;
a=0.5;
N=15;
h=(1-a*exp(j.*w*N))./(1-a*exp(j.*w));
plot(w,abs(h))
title("comb filter N=15");
xlabel("w");
ylabel(" abs of H(w)");
z=exp(j*w);
z=1./z;
H=(1-power(a*z,N))./(1-a*z);
m=1:1:N;
zplane([1,power(a,m)],[1,-a])
```

GRAPH:





CONCLUSION:

- we get the peaks of magnitude graph (like a comb) goes on decreasing and whatever the peak value on the right side is present at the left side at the corresponding value of w.

- In the pole-zero plot we are able to see N-1 zeros because one zero and pole occur at the same point.

3 To plot the magnitude and phase response for the poles at inverse locations.

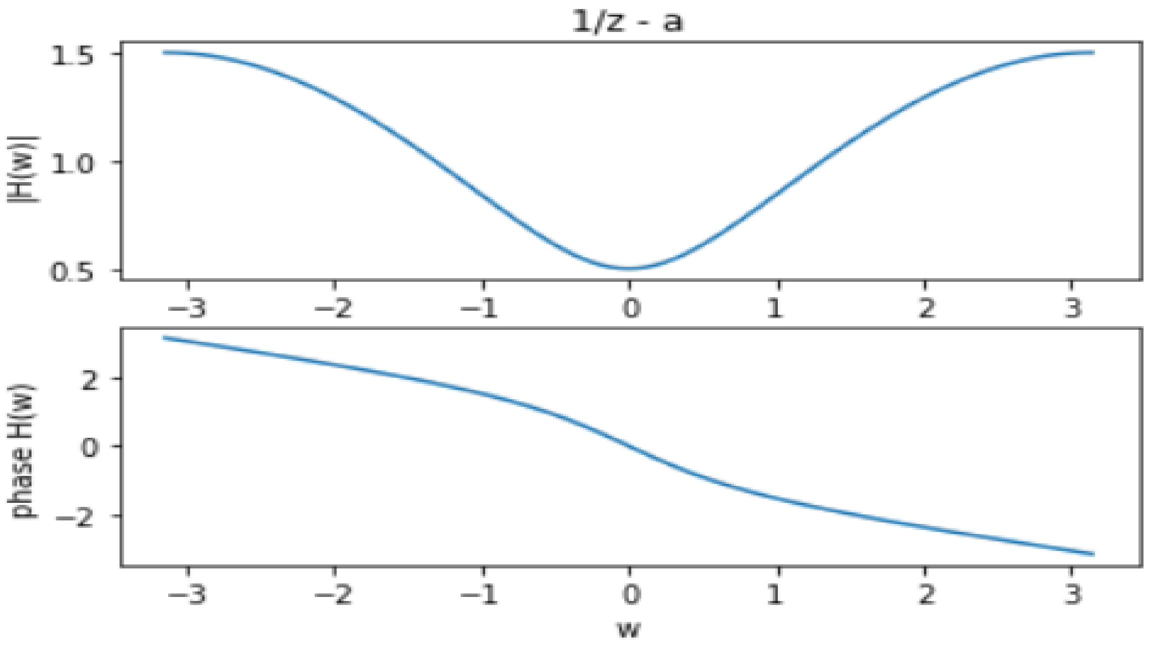
AIM: To plot the magnitude and phase response for the poles at inverse locations.

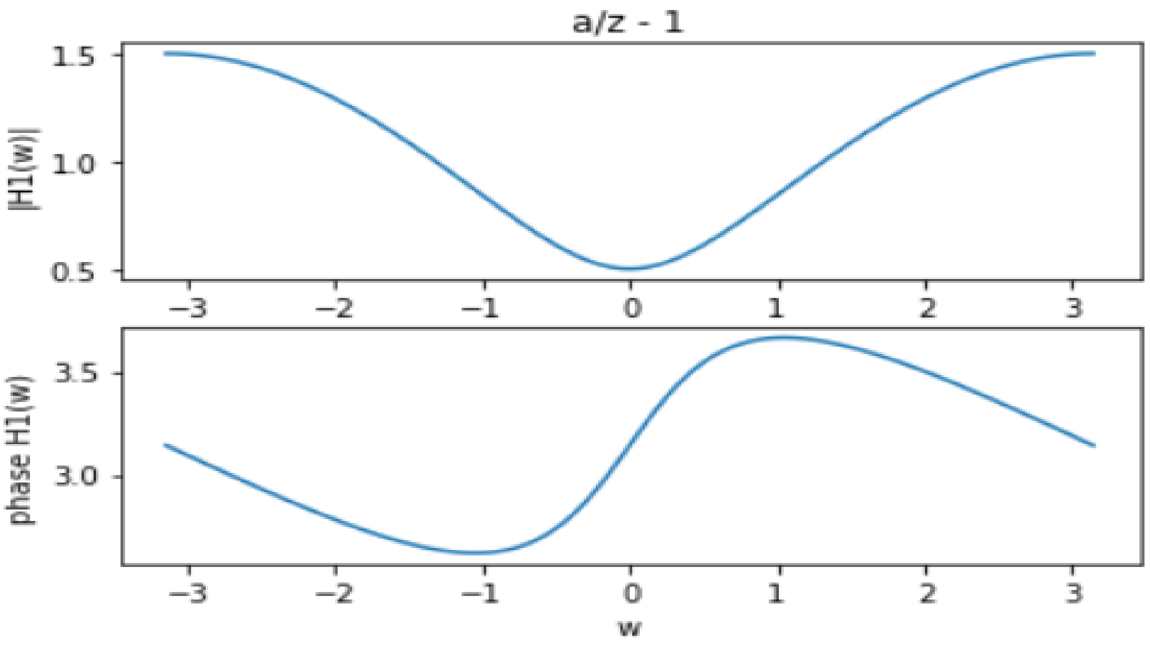
APPARATUS: Python software.

CODE:

```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 22 15:50:06 2020

@author: KARTHIKEYA
"""

import matplotlib.pyplot as plt
import numpy as np

w=np.linspace(-np.pi,np.pi,101)
z=np.exp(1j*w)
z=1/z
a=0.5
H=z-a
H1=a*z-1
plt.figure(1)
plt.subplot(2,1,1)
plt.plot(w,abs(H))
plt.xlabel('w')
plt.title('1/z - a')
plt.ylabel('|H(w)|')
plt.subplot(2,1,2)
plt.plot(w,np.angle(H))
plt.xlabel('w')
plt.ylabel('phase H(w)')
plt.figure(2)
plt.subplot(2,1,1)
plt.plot(w,abs(H1))
plt.xlabel('w')
plt.title(a/z - 1)
plt.ylabel('|H1(w)|')
plt.subplot(2,1,2)
plt.plot(w,np.unwrap(np.angle(H1)))
plt.xlabel('w')
plt.ylabel('phase H1(w)')
```

GRAPH:





CONCLUSION:

- The magnitude of the two graphs is same, But the phase varies.
- In the pole-zero plot we are able to see N-1 zeros because one zero and pole occur at the same point.
- To avoid this additional phase shift we use $z^{-}1 - a$ as a standard form(causal system)

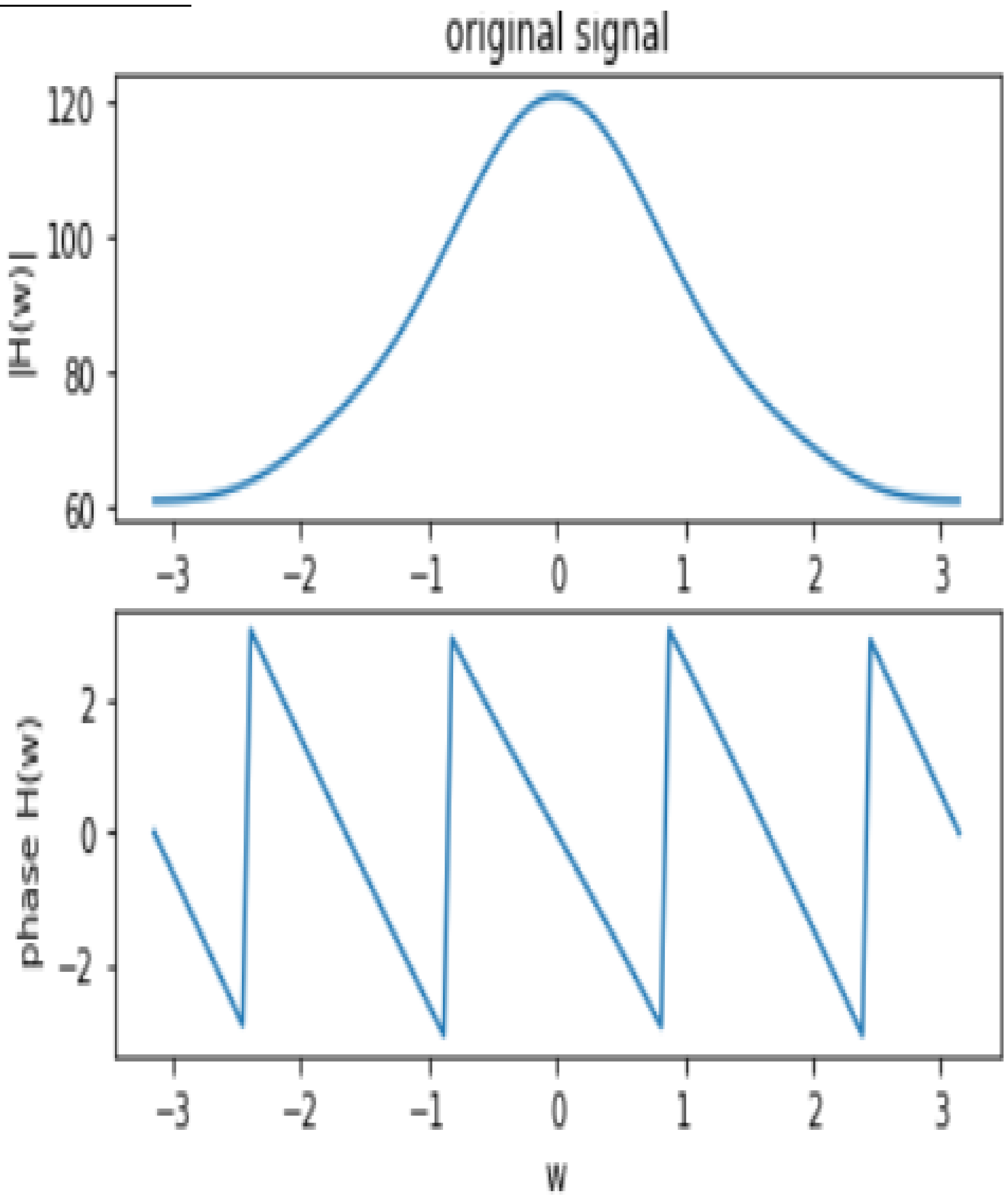
4 To show every signal can be represented as the product of all pass and min phase.

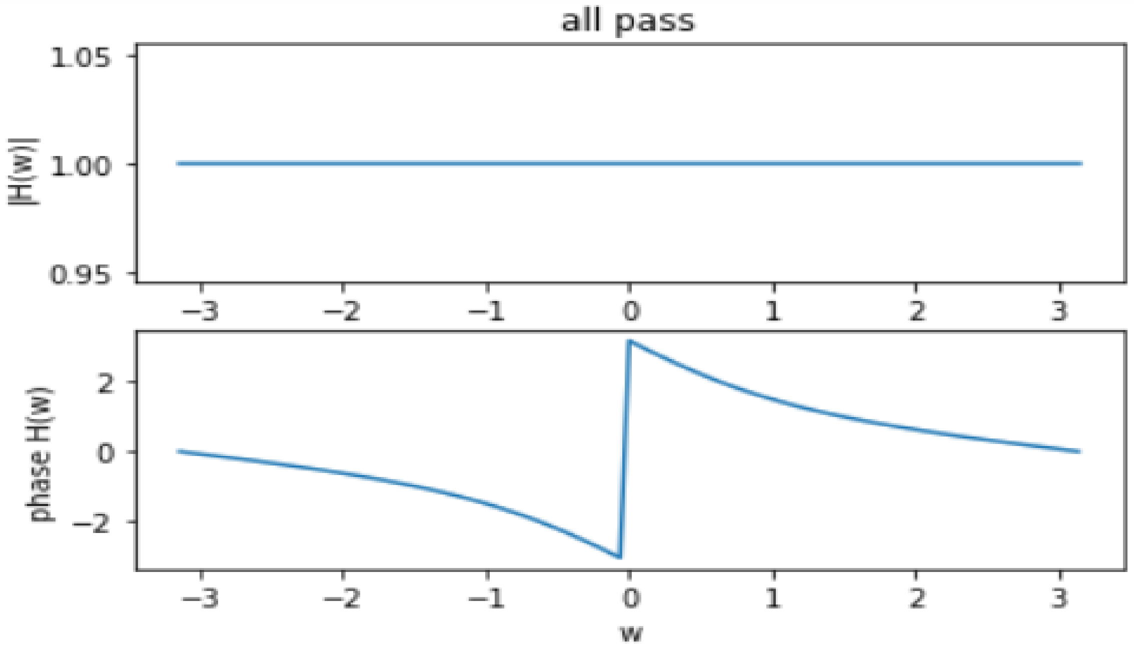
AIM: To show every signal can be represented as the product of all pass and min phase.
APPARATUS: Python software.

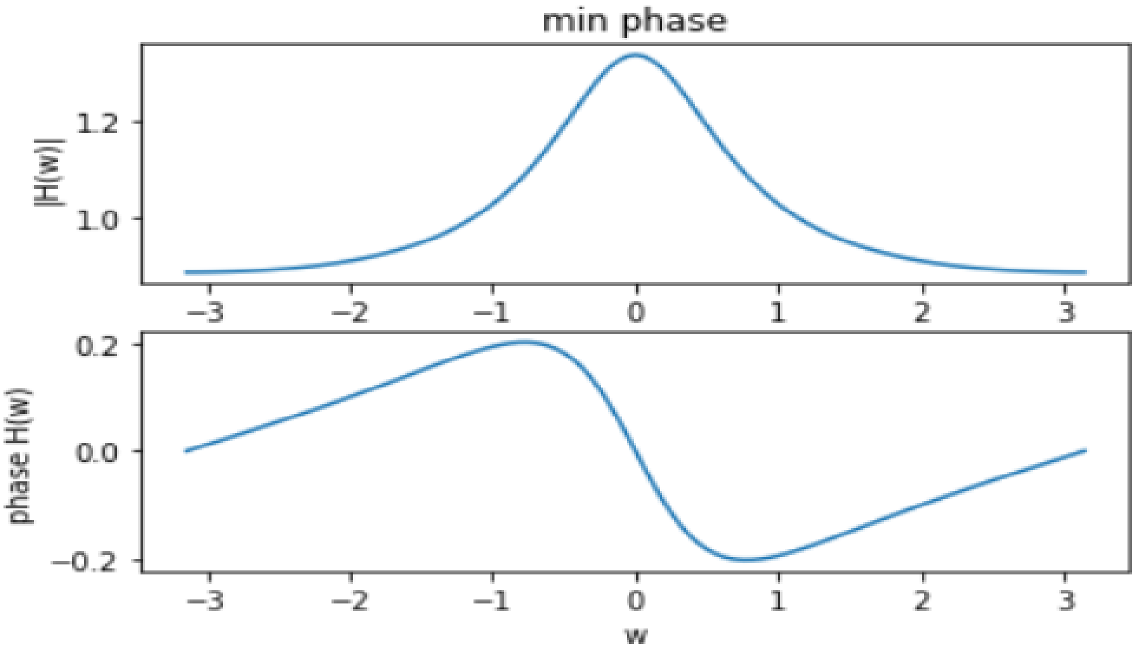
CODE:

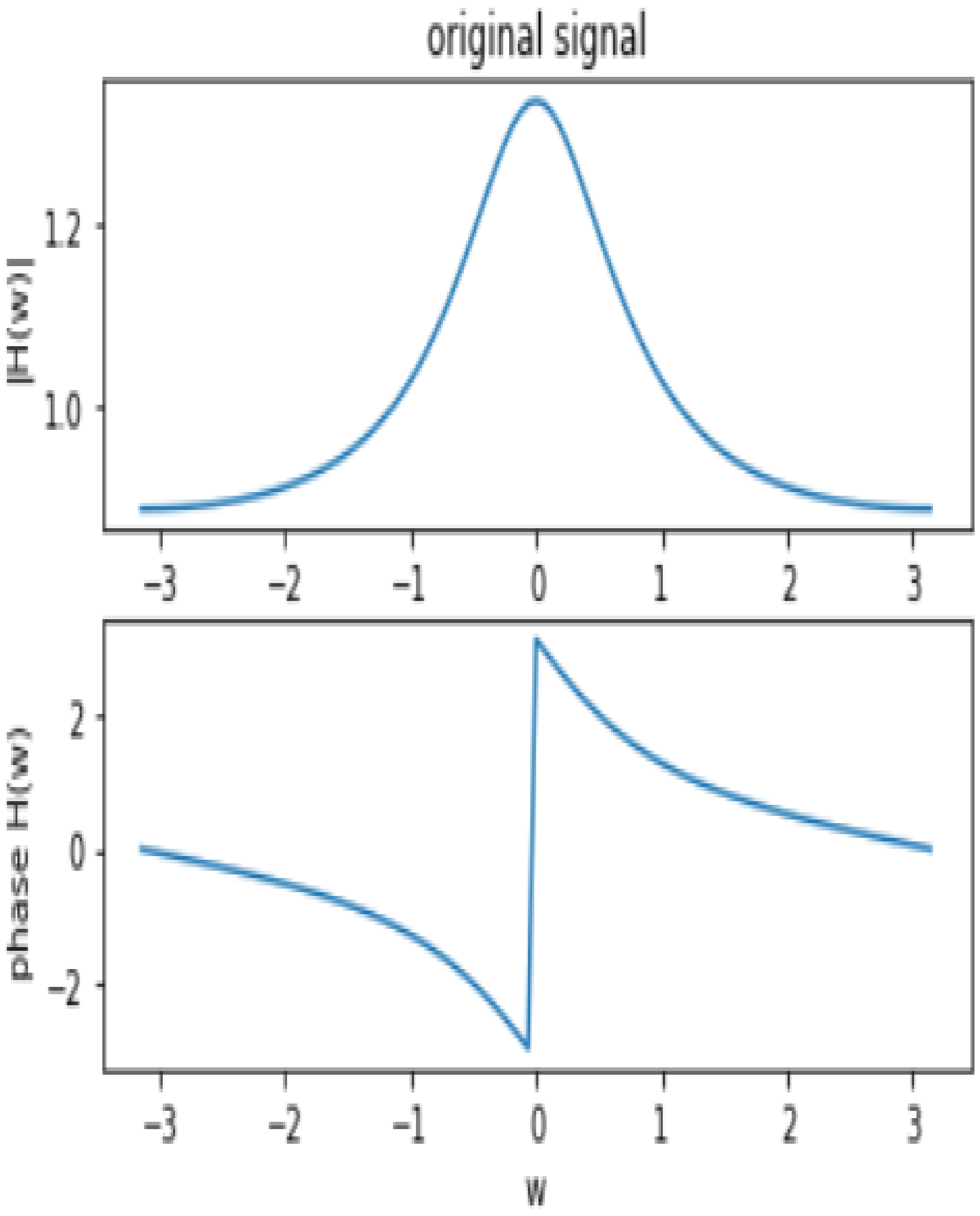
```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan 16 12:29:54 2020

@author: KARTHIKEYA
"""
import matplotlib.pyplot as plt
import numpy as np
plt.close("all")
w=np.linspace(-np.pi,np.pi,101)
z=np.exp(1j*w)
z=1/z
#h2=z+0.5
h1=((1-3*z)/(1-(1/3)*z))*(1/3) #all pass
h2=((1-(1/3)*z)/(1-0.5*z))#min phase
#h1=((1-3*z)/(1-0.5*z)) #H(z)=(a-z^-1)/(1-az^-1)
#h2=(1-(1/3)*z)/(1-(1/3)*z)
H=h1*h2
plt.figure(1)
plt.subplot(4,1,1)
plt.plot(w,abs(h1))
plt.subplot(4,1,2)
plt.plot(w,np.angle(h1))
plt.subplot(4,1,3)
plt.plot(w,abs(h2))
plt.subplot(4,1,4)
plt.plot(w,np.unwrap(np.angle(h2)))
plt.figure(2)
plt.subplot(2,1,1)
plt.plot(w,abs(H))
plt.subplot(2,1,2)
plt.plot(w,np.angle(H))
'''a=3
N=5
H1=(1-(a*exp(1j*w*N))/(1-(a*z))
plt.figure(3)
plt.subplot(2,1,1)
plt.plot(w,abs(H1))
plt.subplot(2,1,2)
plt.plot(w,np.angle(H1))'''
```

GRAPHS:


original signal
|H(w)|
120
100
80
60
-3
-2
-1
0
1
2
3
phase H(w)
2
0
-2
-3
-2
-1
0
1
2
3
w


all pass
|H(w)|
phase H(w)
w


min phase
|H(w)|
phase H(w)
w



CONCLUSION:

- Every signal can be represented as the product of all pass and min phase ,if any signal passes through all pass the magnitude is remain same and the phase of the signal changes;(pole's and zero's present at complex conjugate).
- For min phase the phase change is zero the magnitude remain to be same.

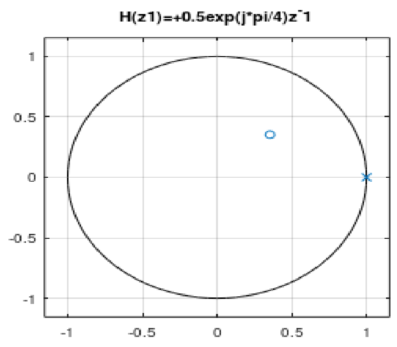
5 To plot the conjugate pole zero plots with their magnitude.

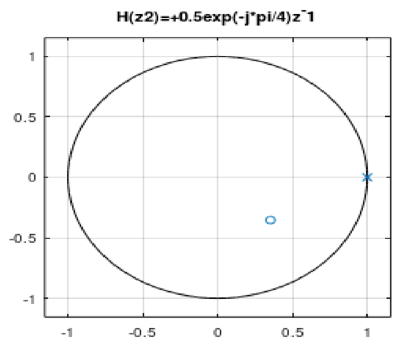
AIM: To plot the conjugate pole zero plots with their magnitude.
APPARATUS: Octave software.

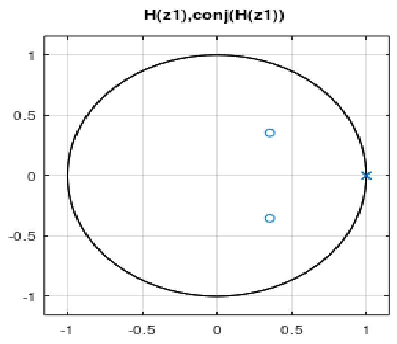
CODE AND GRAPHS:

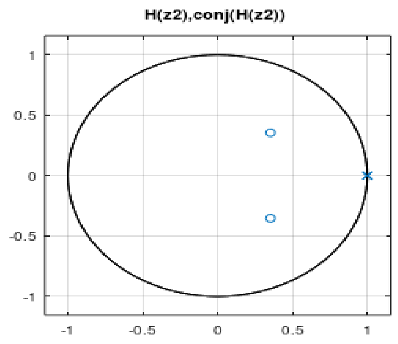
```
 clc;close all;clear all;
 w=-pi:0.1:pi;
 z=exp(j*w);
 z=1./z;
 z1=0.5*exp(j*pi/4);
 z2=0.5*exp(-j*pi/4);
z3=conj(z1);
z4=conj(z2);
figure(1);
subplot(2,2,1);
zplane([z1],[1]);
title("H(z1)=+0.5exp(j*pi/4)z?1");
 subplot(2,2,2);
 zplane([z2],[1]);
 title("H(z2)=+0.5exp(-j*pi/4)z?1 ");
 subplot(2,2,3);
 zplane([z1],[1]);
hold on;
zplane([z3],[1]);
title("H(z1),conj(H(z1)) ");
subplot(2,2,4); zplane([z2],[1]);
hold on;zplane([z4],[1]);
title("H(z2),conj(H(z2)) ");
figure(2);
subplot(2,2,1);
plot(w,abs(z*z1));
title("H(z1)=+0.5exp(j*pi/4)z?1 ");
xlabel("w");
 ylabel("|H(w)|");
 subplot(2,2,2);
 plot(w,abs(z*z2));
 title("H(z2)=+0.5exp(-j*pi/4)z?1 ");
 xlabel("w");
 ylabel("|H(w)|");
 subplot(2,2,3);
 plot(w,abs(z*z3));
 title("conj(H(z1)) ");
 xlabel("w");
 ylabel("|H(w)|");
 subplot(2,2,4);
 plot(w,abs(z*z4));
 title("conj(H(z2)) ");
```

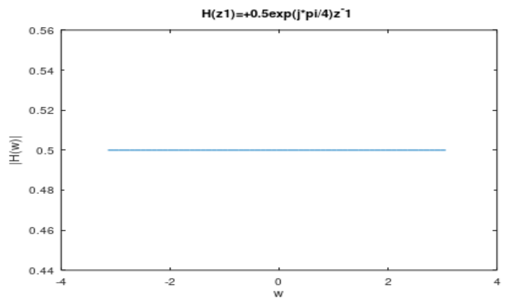
```
xlabel("w");
ylabel("|H(w)|");
```

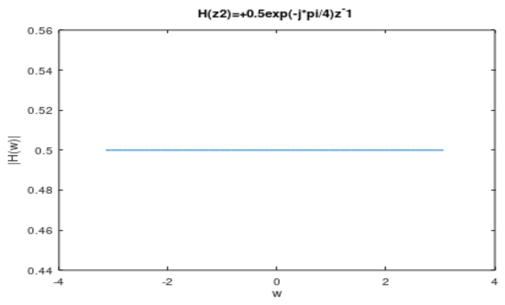


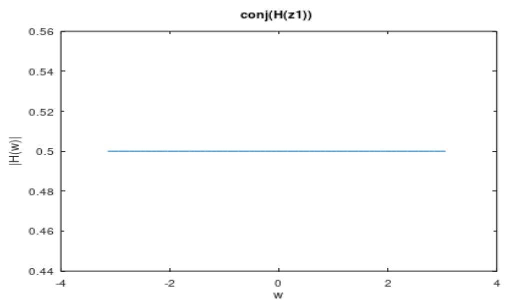


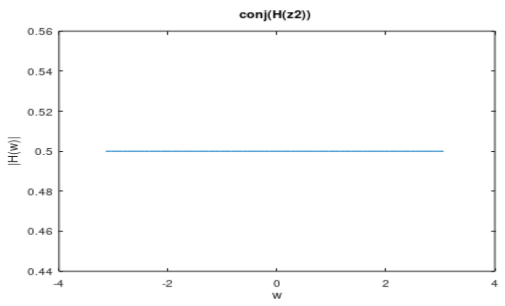












CONCLUSION:

- The magnitude of all of them is same.
- Hence we can conclude that the zeros at complex conjugate location should have the same magnitude.

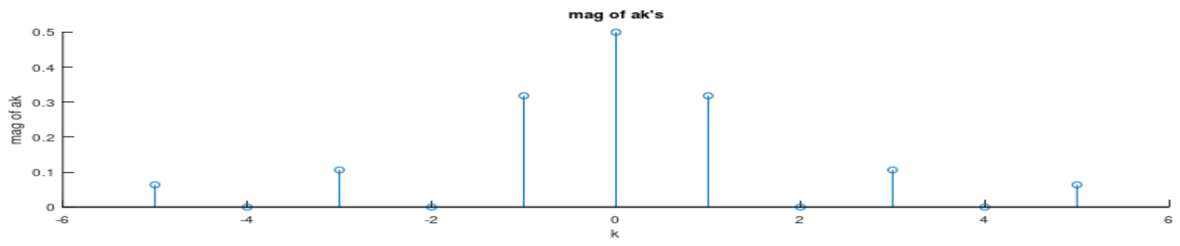
6 To plot the inverse dtft.

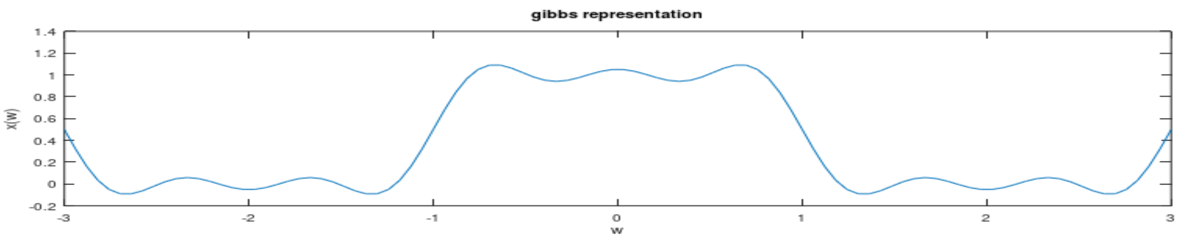
AIM: To plot the inverse dtft.
APPARATUS: Octave software.

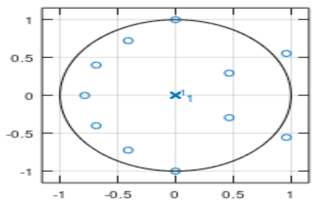
CODE:

```
clc;clear;close all;
k=-5:5;
#taken T =1
a=sin(k.*pi*1./2)./(pi.*k);#obtained ak's by formula
a(k==0)=0.5;
subplot(3,1,1)
stem(k,a);
title("mag of ak's")
xlabel("k")
ylabel('mag of ak')
t=linspace(-3,3,100);
x=zeros(1,100);#original signal
x(t>-1 & t<1)=1;
s=zeros(1,length(t));
i=1;
for m=1:length(k)
 y=a(m).*exp((j.*k(m)*t*pi)./2);#by analysis formula
 s=s+y;
  i++;
endfor
subplot(3,1,2)
plot(t,s)
xlabel("w")
ylabel('x(w)')
title("gibbs representation")
hold on
plot(t,x)
subplot(3,1,3)
zplane([0.5,a],[1])
```

GRAPH:







CONCLUSION:

- *If we increase the range of k the complex exponentials are merge in the square wave except at the edges(discontinuous points).

7 To create a filter with window method for an audio.

AIM: To create a filter with window method for an audio.

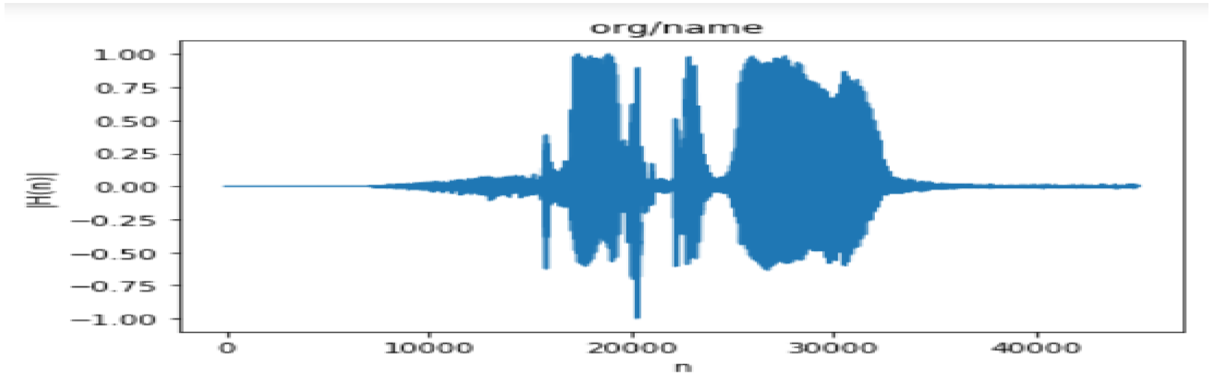
APPARATUS: Python software.

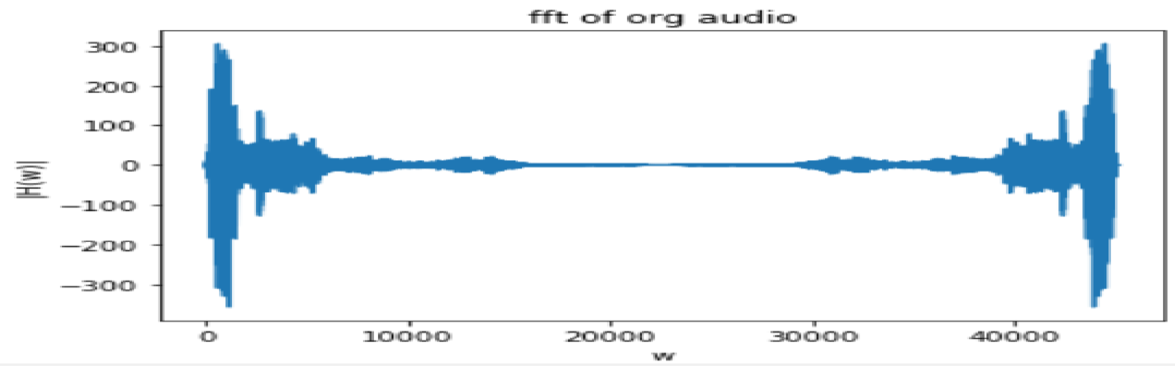
CODE:

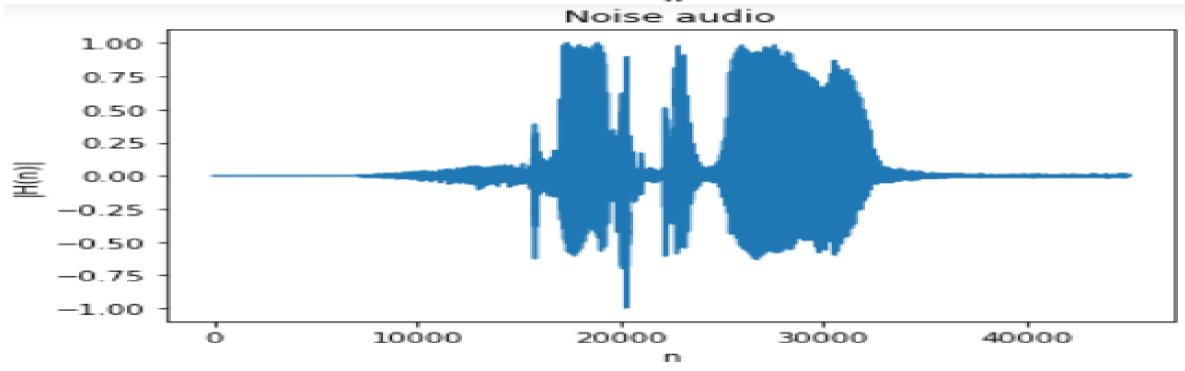
```
import librosa
import numpy as np
import matplotlib.pyplot as plt
import librosa.display#used to load
    the song
import IPython.display as ipd
w, sr = librosa.load(r'F:\book\
    bt18ece055_karthikeya\lab oct 55\
    karthik.wav')
#%loading wav file
print(np.shape(w),sr)
print(w)
ipd.Audio(w,rate=sr)
plt.figure(1)
plt.plot(w)
w_f=np.fft.fft(w)
plt.xlabel('n')
plt.title('org/name ')
plt.ylabel('|H(n)|')
plt.figure(2)
plt.plot(w_f)
plt.xlabel('w')
plt.title('fft of org audio')
plt.ylabel('|H(w)|')
#s, sr1 = librosa.load(r'F:\book\
    bt18ece055_karthikeya\lab oct 55\
    sinew.wav')
#s=s(:,4502)
#print(np.shape(s))
#print(s)
si=np.sin(2*np.pi*(200/sr))
w_s=w+si #% adding noise
plt.figure(3)
plt.plot(w)
w_sf=np.fft.fft(w_s)
plt.xlabel('n')
plt.title('Noise audio')
plt.ylabel('|H(n)|')
plt.figure(4)
plt.plot(w_sf)
ipd.Audio(w_sf,rate=sr)
plt.xlabel('w')
plt.title('fft of org with noise
    audio')
plt.ylabel('|H(w)|')
ipd.Audio(w_sf,rate=sr)
n=np.array(range(-20000,20000))
```

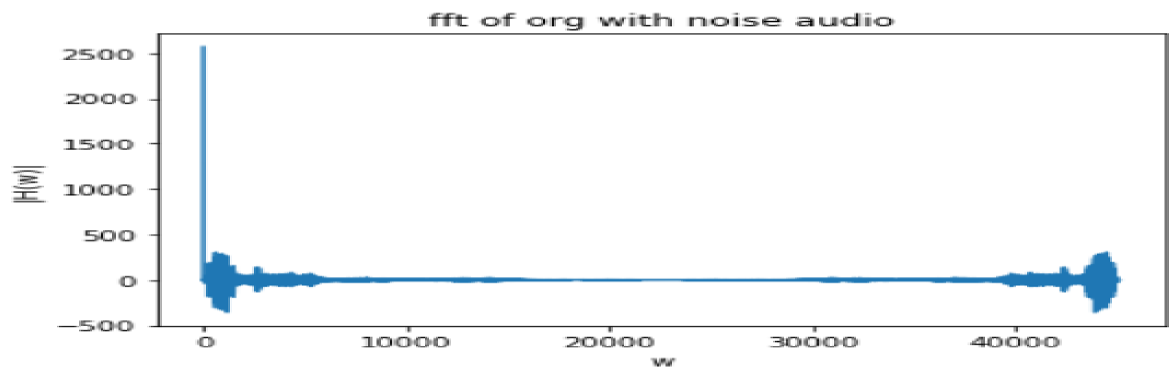
```
sin_c=np.sin(2*np.pi*(1/sr)*n)/np.pi
    *n
plt.figure(5)
plt.plot(sin_c)
plt.xlabel('n')
plt.title('sinc')
plt.ylabel('|H(n)|')
sin_cf=np.fft.fft(sin_c)
sin_cf=(sin_cf/10000000)+10
plt.figure(6)
plt.plot(n,sin_cf)
plt.xlabel('w')
plt.title('fft of sinc')
plt.ylabel('|H(w)|')
diff=len(w_sf)-len(sin_cf)
print(diff/2)
x=np.zeros(2526)
y=np.concatenate((x,sin_cf))
z=np.concatenate((y,x))
op=w_sf*z
plt.figure(1)
plt.plot(op)
plt.xlabel('n')
plt.title('fft of filtered wave')
plt.ylabel('|H(n)|')
opf=np.fft.ifft(op)
plt.figure(2)
plt.plot(opf)
plt.xlabel('w')
plt.title('filtered wave')
plt.ylabel('|H(w)|')
ipd.Audio(opf,rate=sr)
```

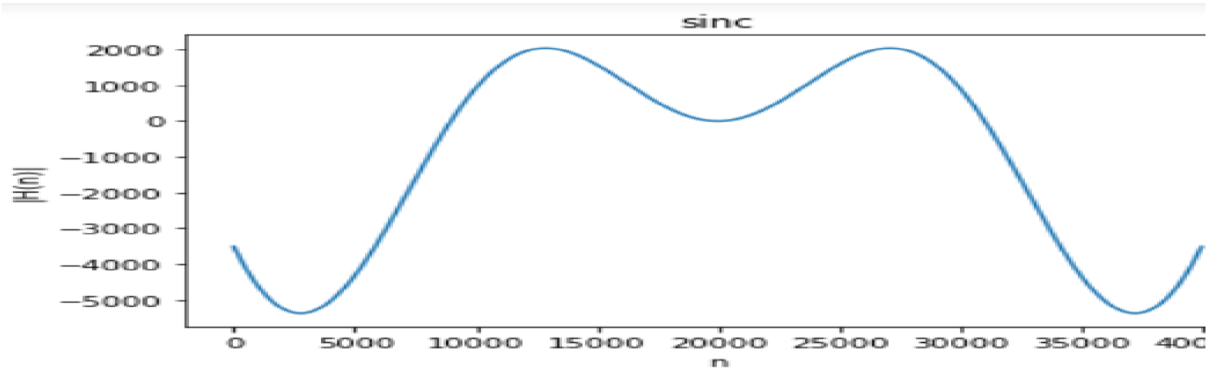
GRAPH:

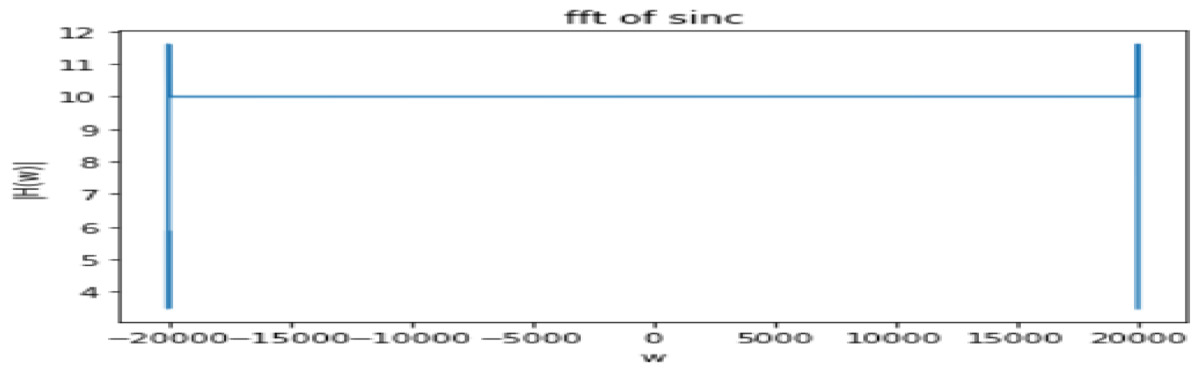


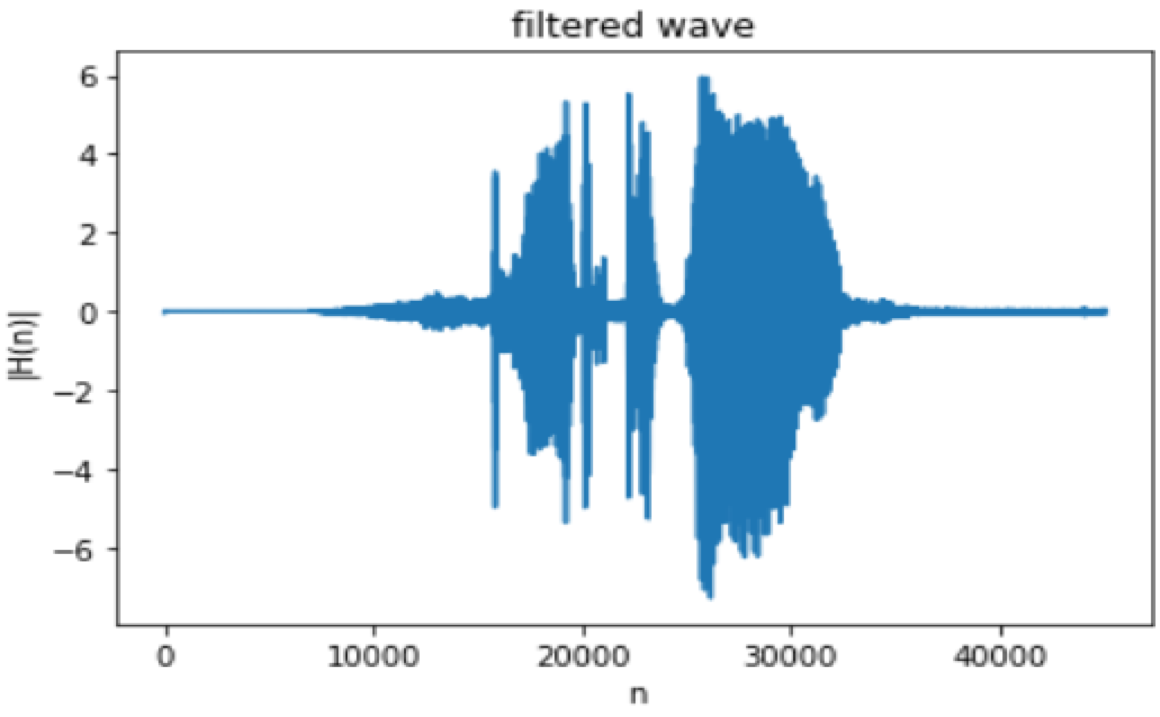


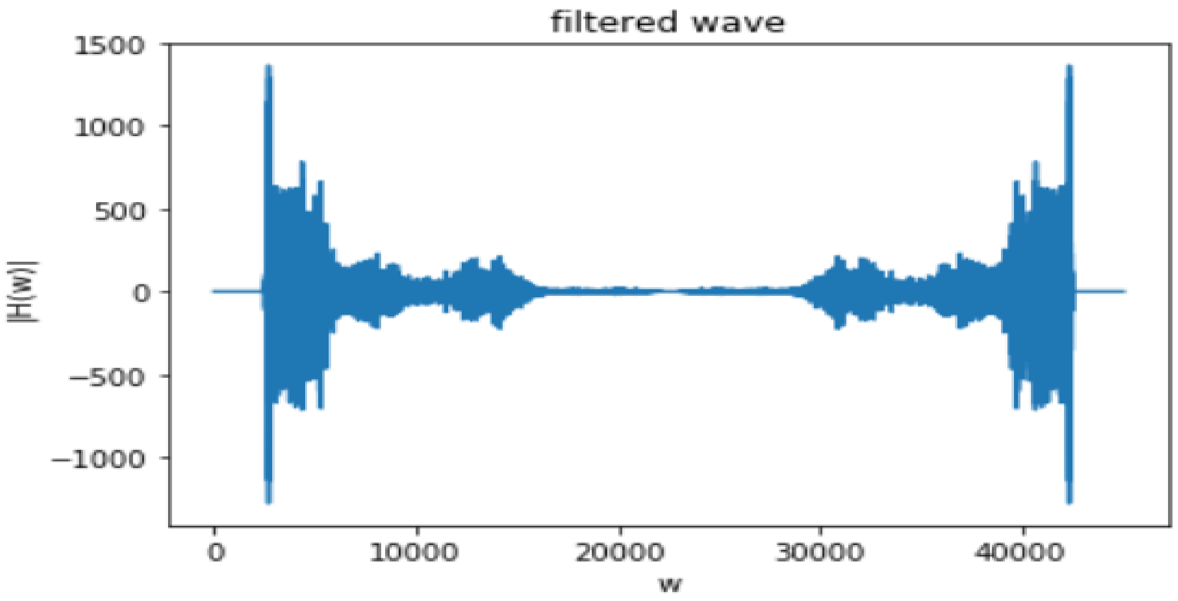





sinc
|H(n)|
n


fft of sinc
|H(w)|
w


filtered wave
|H(n)|
n


filtered wave
1500
1000
500
0
−500
−1000
|H(w)|
0
10000
20000
30000
40000
w

CONCLUSION:

- We have taken the audio and added noise to it. Mutiply fft of audio with window and then do inverse fft to get audio without the noise.
- The problem faced during the exp is to build the rectangle wave (using trial and error method i.e I keep changing the values of n, w_c to get desired rectangle and adding zeros before and after the rect to get in same size of audio given).
- We used a rectangular filter,the disadvantage of rectangular filter is we get ripples at the edges.
- we are able to get the original audio after filtering.

8 To plot the rectangular,hanning,hamming window.

AIM: To plot the rectangular,hanning,hamming window.
APPARATUS: octave software.

THEORY:
*For rectangle window we get more ripples,so we go for hanning,hamming window's.

$$hanningwindow = 0.5 * (1 + cos((2 * pi. * n)/M))$$

$$hammingwindow = 0.54 + 0.46 * cos((2 * pi. * n)/M)$$

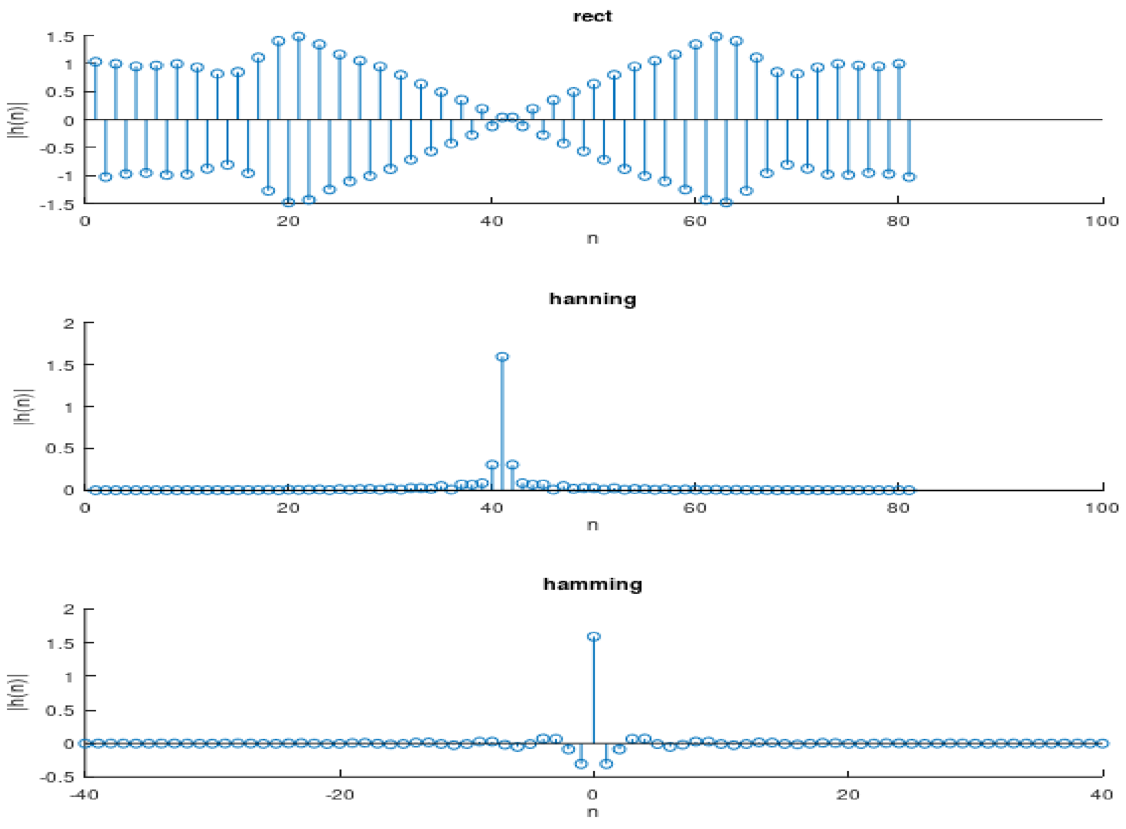
where M is the order of the filter.
CODE:

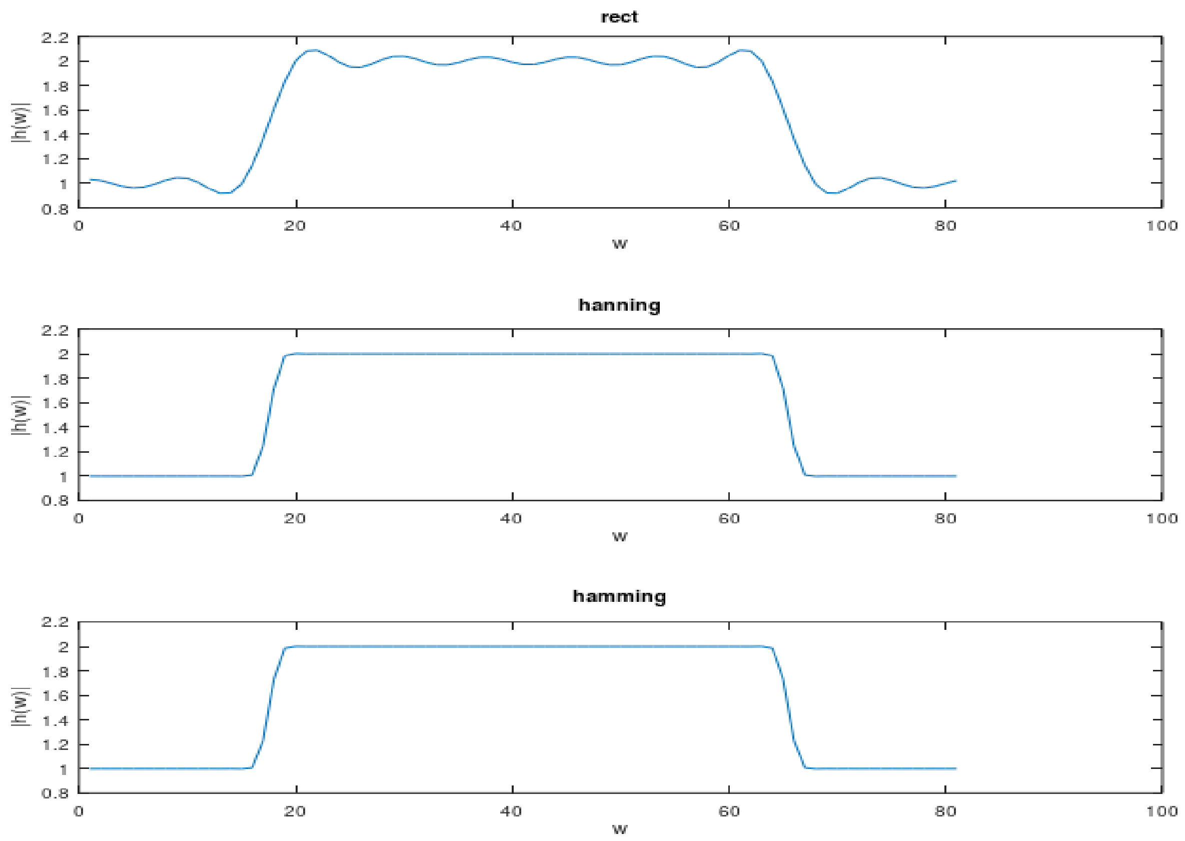
```
clc;clear;close all;
M=80;
n=-(M/2):1:(M/2);
w=5;
h=sin(w.*n)./(pi.*n);#sinc function
h(41)=w/pi;
re=[zeros(1,30),ones(1,21),zeros
    (1,30)];#rectangular window
rect=h.*re;#h_ideal*window
H=fft(rect);
w_han=0.5*(1+cos((2*pi.*n)/M));
w_ham=0.54+0.46*cos((2*pi.*n)/M);

h_han=h.*w_han;#h_ideal*window
```

```
h_ham=h.*w_ham;#h_ideal*window

figure(1)
subplot(3,1,1)
stem(H)
xlabel('n')
ylabel('|h(n)|')
title('rect')
subplot(3,1,2);
stem(abs(h_han))
xlabel('n')
ylabel('|h(n)|')
title('hanning')
subplot(3,1,3);
stem(n,h_ham)
xlabel('n')
ylabel('|h(n)|')
title('hamming')
figure(2)
subplot(3,1,1)
plot(abs(H))
xlabel('w')
ylabel('|h(w)|')
title('rect')
subplot(3,1,2)
plot(abs(fft(h_han)))
xlabel('w')
ylabel('|h(w)|')
```

```
title('hanning')
subplot(3,1,3)
plot(abs(fft(h_ham)))
xlabel('w')
ylabel('|h(w)|')
title('hamming')
```

GRAPH:





CONCLUSION:

- we have taken a sinc function and multiplied with different windows and converted them into frequency domain.
- Each window has its own pros and cons. For example:rect- less transition width. Hanning/Hammin transition width more than rectangular.
- The reason to go for other windows is get good filter for desired features.For eg: the rectangle window has more ripples(side lobes) where the others does not posses them.

9 To plot the low pass,high pass,band pass,band reject filters and their zeros in z plane.

AIM: To plot the low pass,high pass,band pass,band reject filters and their zeros in z plane.

APPARATUS: octave software.

THEORY:

- Lowpass filter with allows the frequencies less than w_c and reject which are higher than that.

$$lp(n! = 0) = \frac{sin(w * n)}{pi * n} \tag{1}$$

$$lp(0) = \frac{w}{pi} \tag{2}$$

- Highpass filter with allows the frequencies more than w_c and reject which are lower than that.

$$hp(n! = 0) = \frac{sin(pi * n) - sin(w * n)}{(pi * n)} \tag{3}$$

$$hp(0) = \frac{w}{pi} \tag{4}$$

- Bandpass filter with allows the frequencies in between $w1, w2$ and reject all other.

$$B_p(n! = 0) = \frac{sin(w_2 * n) - sin(w_1 * n)}{(2 * pi * n)} \tag{5}$$

$$B_p(0) = \frac{w_2 - w_1}{p} \tag{6}$$

- Band reject filter which allows the frequencies in which are below w_1 and above w_2 and reject all other.

$$Rp(n! = 0) = \frac{sin(w_1 * n) - sin(w_2 * n)}{(2 * pi * n)} \tag{7}$$

$$Rp(0) = \frac{(pi - w2 + w1)}{pi} \tag{8}$$

CODE and GRAPHS:

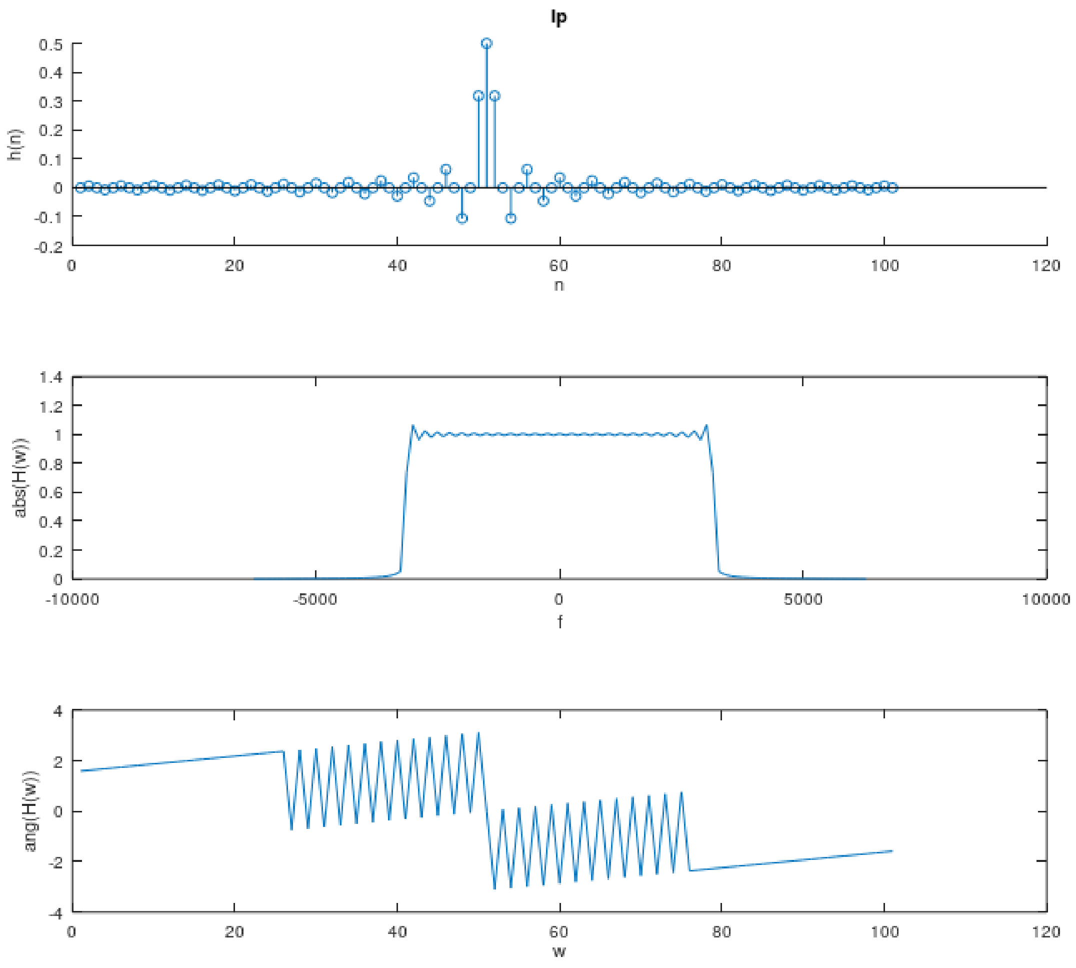
```
clc;clear;close all;
L=101;
M=L-1;
n=-(M/2):1:(M/2);
a=51;
F=20;
fs=2000;
#w=(2*pi*F)/fs;
w=pi/2;
x1=linspace(-pi,pi,L);
x=x1*fs;
h=sin(w.*n)./(pi.*n);
h(a)=w/pi;

y=fft(h);


figure(1)
subplot(3,1,1)
stem(h)
```

```
xlabel('n')
ylabel('h(n)')
title("lp")

subplot(3,1,2)
plot(x,abs(fftshift(y)))
xlabel('f')
ylabel('abs(H(w))')

subplot(3,1,3)
plot(angle(fftshift(y)))
xlabel('w')
ylabel('ang(H(w))')
#zplane([h(1),h(2),h(3)],[1])


H=-sin(w.*n)./(pi.*n);
H(a)=1-(w/pi);

figure(2)
subplot(3,1,1)
stem(H)
xlabel('n')
ylabel('h(n)')
title("hp")
subplot(3,1,2)
b=fft(H);
plot(x,abs(fftshift(b)))
```

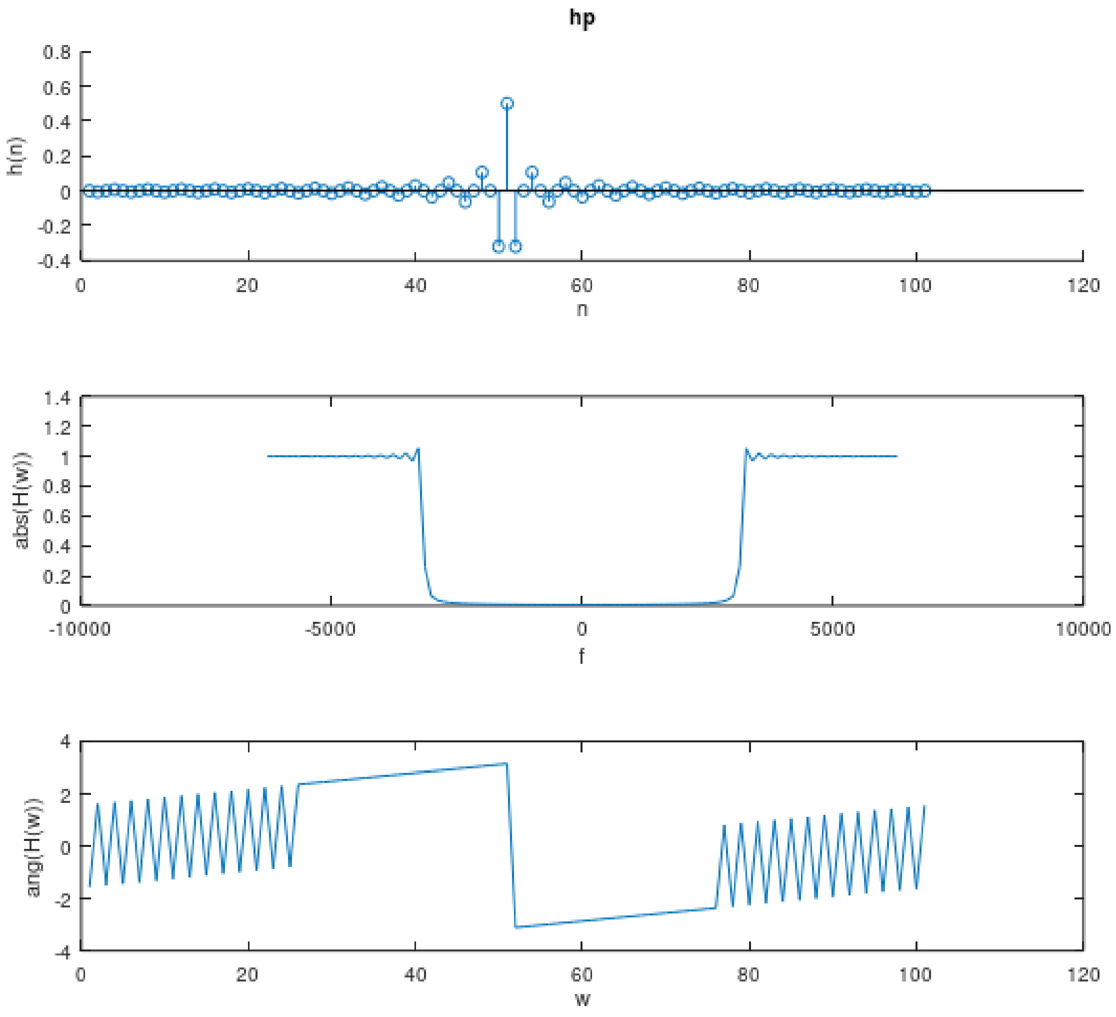
```
xlabel('f')
ylabel('abs(H(w))')
subplot(3,1,3)
plot(angle(fftshift(b)))
xlabel('w')
ylabel('ang(H(w))')
#zplane([H(1),H(2),H(3)],[1])

w1=w;
w2=1.5*w;
h_bp=(sin(w2.*n)-sin(w1.*n))./(pi.*n
    );
h_bp(a)=(w2-w1)/pi;
figure(3)
subplot(3,1,1)
stem(h_bp)
xlabel('n')
ylabel('h(n)')
title("bp")

subplot(3,1,2)
bp=fft(h_bp);
plot(x,abs(fftshift(bp)))
xlabel('f')
ylabel('abs(H(w))')
subplot(3,1,3)
plot(angle(fftshift(bp)))
xlabel('w')
```

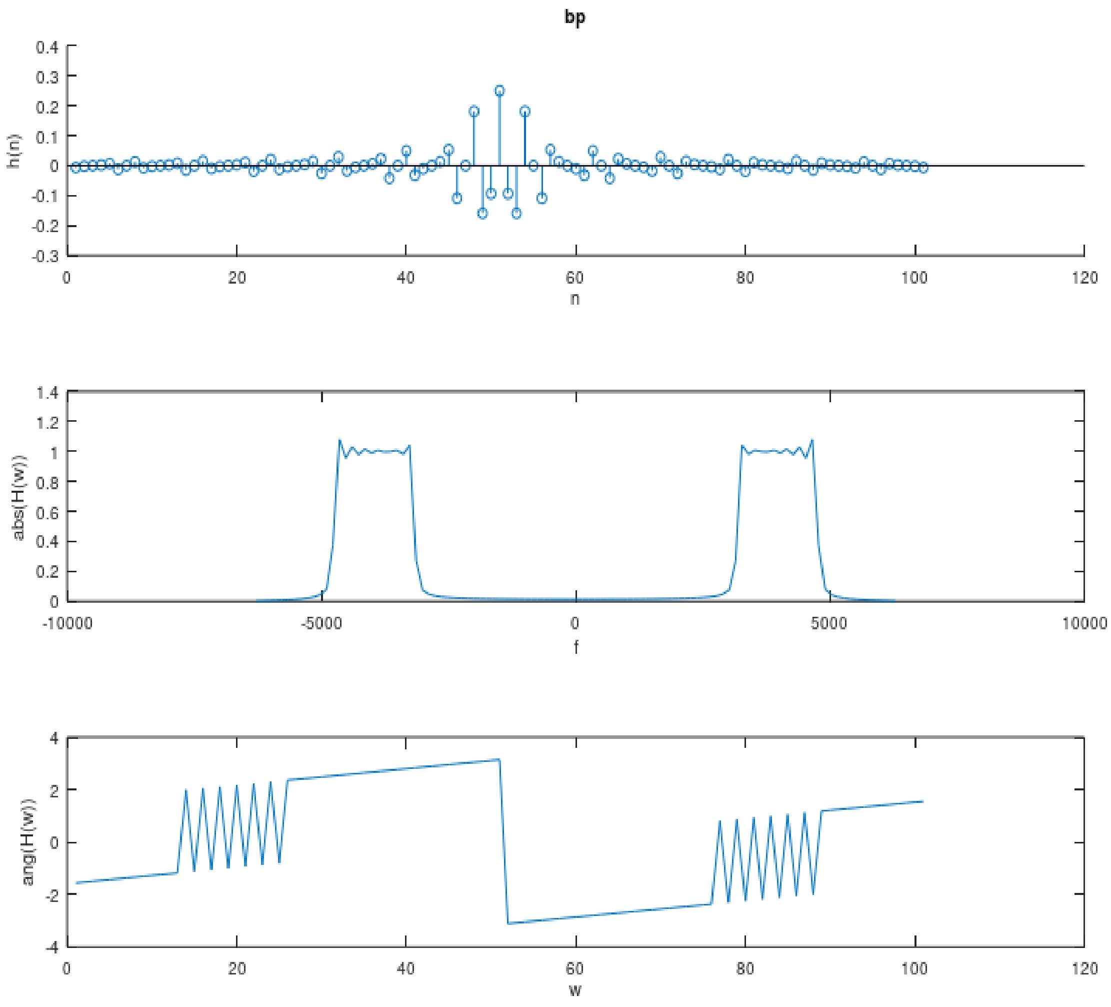
```
ylabel('ang(H(w))')
#zplane([h_bp(1),h_bp(2),h_bp(3)],[1])

h_rp=(sin(w1.*n)-sin(w2.*n))./(pi.*n);
h_rp(a)=(pi-w2+w1)/pi;
figure(4)
subplot(3,1,1)
stem(h_rp)
xlabel('n')
ylabel('h(n)')
title("br")

subplot(3,1,2)
rp=fft(h_rp);
plot(x,abs(fftshift(rp)))
xlabel('f')
ylabel('abs(H(w))')
subplot(3,1,3)
plot(angle(fftshift(rp)))
xlabel('w')
ylabel('ang(H(w))')
#zplane([rp(1),rp(2),rp(3)],[1])
##figure(5)
##h123=h(1:M);
##zplane(h123,[1])
##title("lp")
```

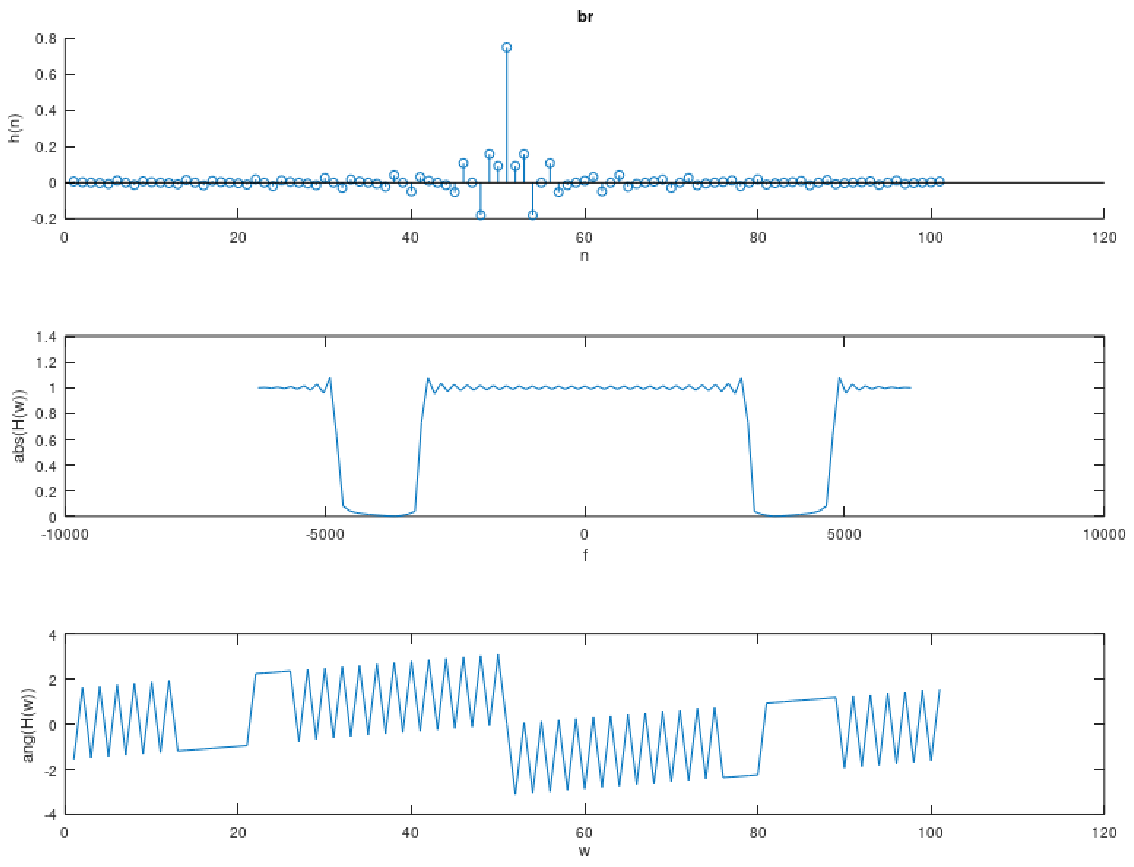
```
##H12=H(1:M);
##figure(6)
##zplane(H12,[1])
## title("hp")
##h_bp1=h_bp(1:M);
##figure(7)
##zplane(h_bp1,[1])
## title("bp")
##figure(8)
##H_rp1=h_rp(1:M);
##zplane(H_rp1,[1])
## title("br")
```



lp
h(n)
n
abs(H(w))
f
ang(H(w))
w

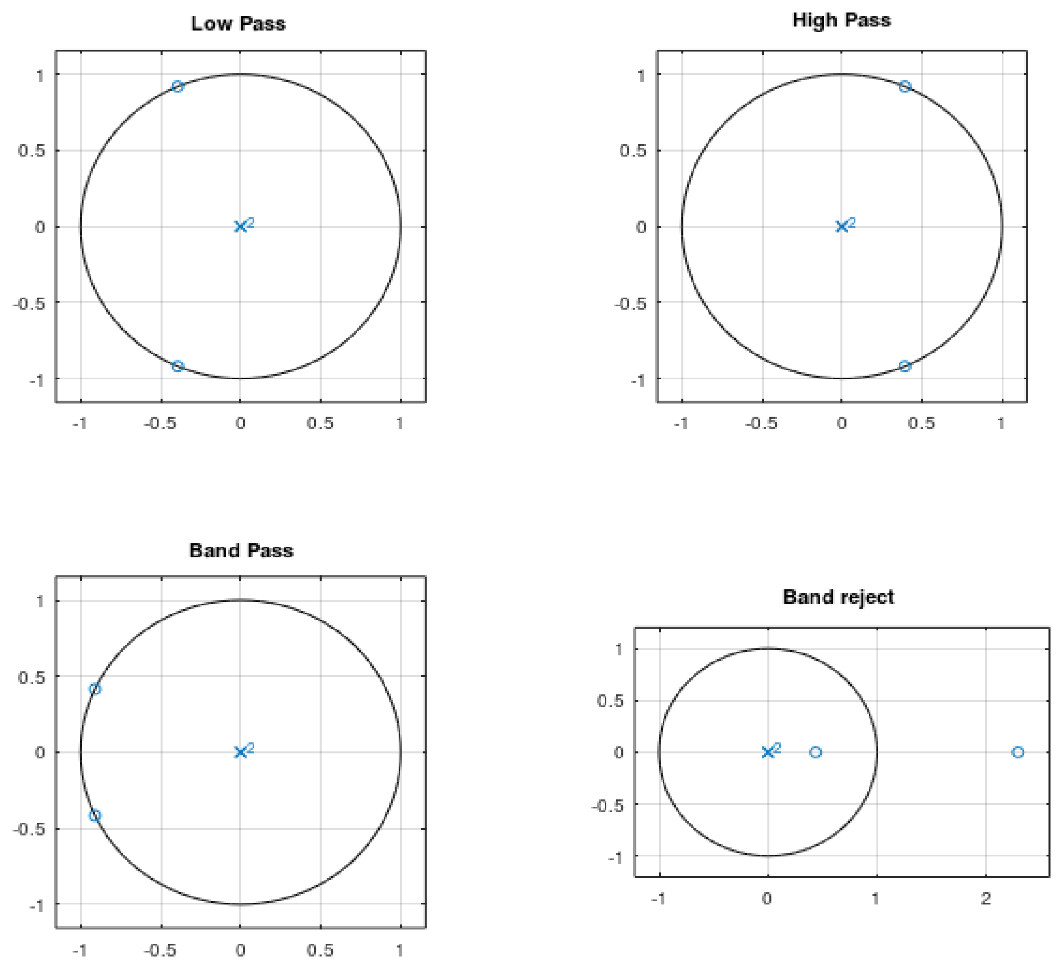

hp
h(n)
n
abs(H(w))
f
ang(H(w))
w


bp
h(n)
n
abs(H(w))
f
ang(H(w))
w



```
clc;clear;close all;
#order=2
#calculated manually,w=pi/2,wc1=pi
    /10;
subplot(2,2,1)
zplane
    ([0.63662,0.5,0.63662],[1,0,0]);
title("Low Pass ");
subplot(2,2,2)
zplane
    ([-0.63662,0.5,-0.63662],[1,0,0]);
title("High Pass");
subplot(2,2,3)
```

```
zplane([0.2199,0.4,0.2199],[1,0,0]);
title("Band Pass");
subplot(2,2,4)
zplane([-0.2199,0.6,-0.2199],[1,0,0]);
title("Band reject ");
```



CONCLUSION:

- We are able to plot the low pass,high pass,band pass ,band reject filters-also observe the pole zero plots.
- For low pass filter we have zeros at higher fre-

quencies,For high pass filter we get zeros at lower frequencies.(if zero are present then those frequencies are attenuated).

- We also observe that the phase is linear for all the filters.

10 To plot symmetric and antisymmetric plots.

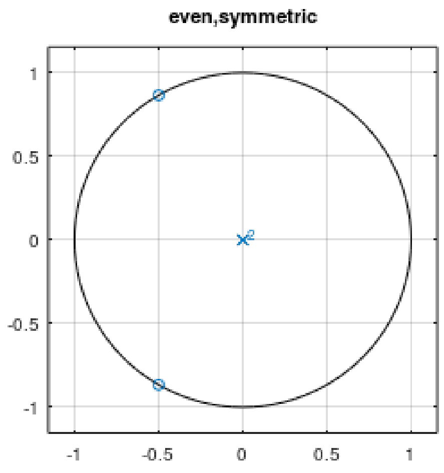
AIM: To plot symmetric and antisymmetric plots in z plane.

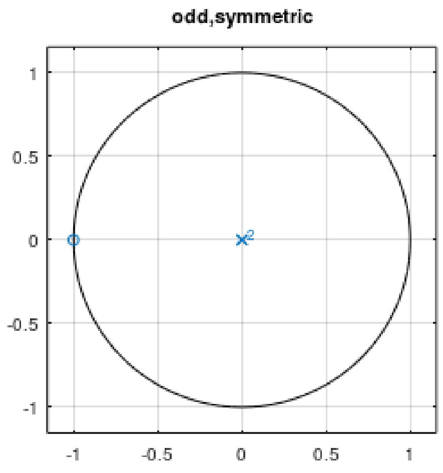
APPARATUS: octave software.

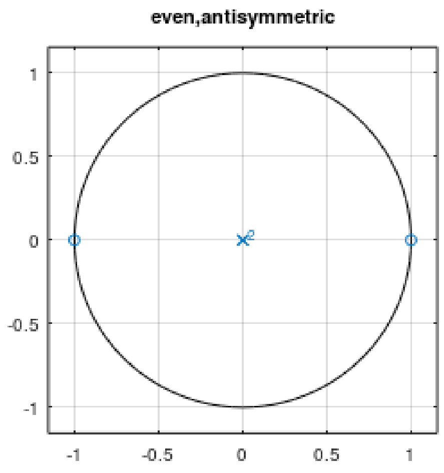
CODE:

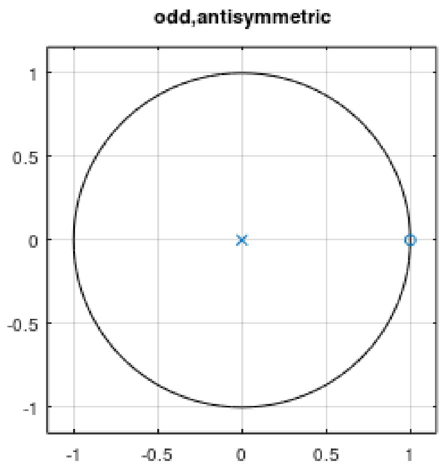
```
clc;
clear;
close all;
w=-pi:0.2:pi;
#m=even and symmetric
subplot(2,2,1)
zplane([0.5,0.5,0.5],[1]);
title('even,symmetric')
#m=odd and symmetric
subplot(2,2,2)
zplane([0,0.5,0.5],[1]);
title('odd,symmetric')
#m=even and antisymmetric
subplot(2,2,3)
zplane([-0.5,0,0.5],[1]);
title('even,antisymmetric')
#m=odd and antisymmetric
subplot(2,2,4)
zplane([-0.5,0.5],[1]);
title('odd,antisymmetric')
```

GRAPH:









CONCLUSION:

- we are able to plot the symmetric and anti symmetric pole zero plots.

11 To design IIR filter from butterworth analog equation.

AIM: To design IIR filter from butterworth analog equation.

APPARATUS:
octave software.
THEORY:
*IIR(infinite impluse response) filters have less order than fir for same set of specifications.
These IIR filters are well developed in analog theory.so by using butterworth equation we convert s domain filter to z domain(i.e analog to digital).

$$H(\Omega) = \frac{1}{\sqrt{1 + \frac{\Omega}{\Omega_c}^{2N}}}$$

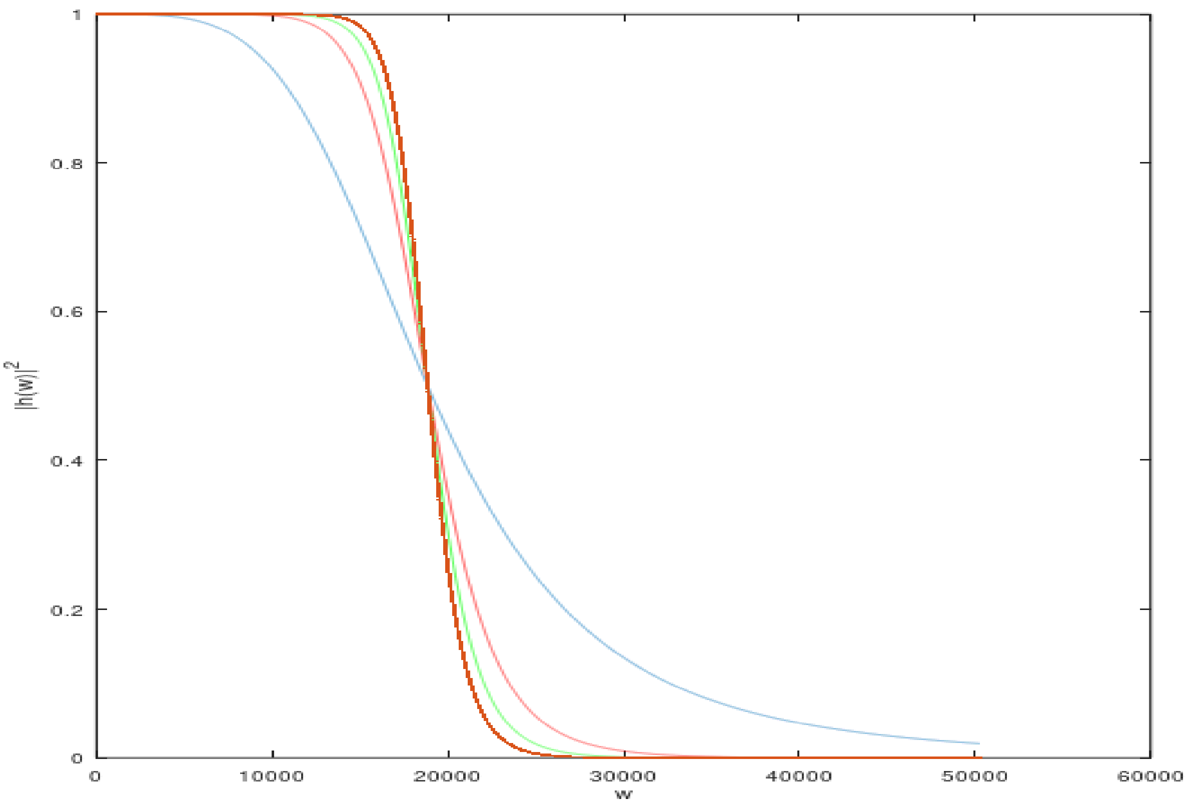
where N is the order;$\Omega = 2\pi * f$
(9)Ω_c *is cutoff frequency*

CODE:

```
clc;clear;close all;
f=0:8000;
w=2*pi*f;
fc1=3000;
wc1=2*pi*fc1;
n1=2;
n2=5;
n3=7;
n4=9;
h1=1./(1+power((w./wc1),2*n1));
h2=1./(1+power((w./wc1),2*n2));
h3=1./(1+power((w./wc1),2*n3));
h4=1./(1+power((w./wc1),2*n4));
plot(w,h1)
hold on;
plot(w,h2,'r')
hold on;
plot(w,h3,'g')
hold on;
```

```
plot(w,h4,'.')
xlabel('\Omega')
ylabel('|h(\Omega)|^2')
```

GRAPH:



CONCLUSION:

- As the value of the ‘n’ increases the graph becomes more as an ideal filter.

12 To plot the poles for butterworth analog equation.

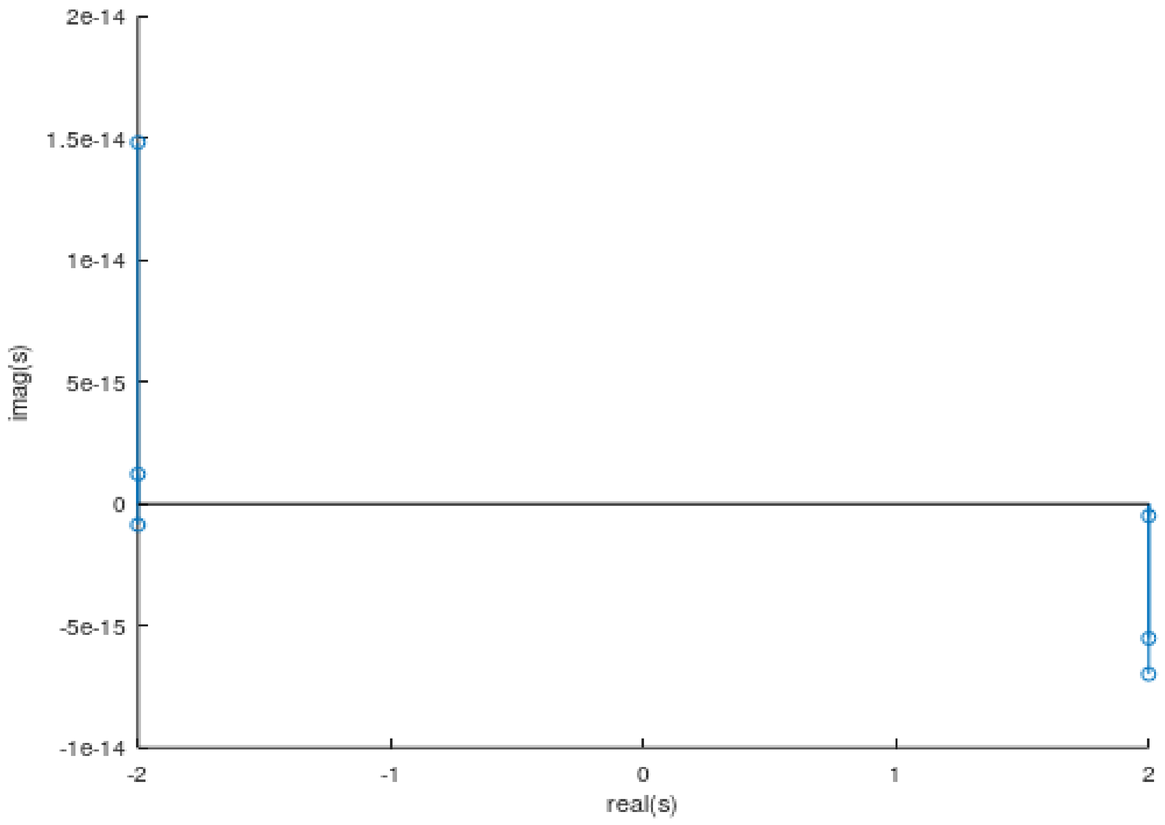
AIM: To plot the poles for butterworth analog equation.

APPARATUS: octave software.

CODE:

```
clc;clear;close all;
wc=1;
N=5;
k=0:2*N-1;
s=wc*exp(j*pi/2)*exp(((j*((2.*k)+1))*pi)/2*N);

figure(2)
stem(real(s),imag(s))
xlabel('real(s)')
ylabel('imag(s)')
```

GRAPH:



CONCLUSION:

- The butterworth equation has 2N roots , applying square root we get an equation with N roots,and those N must lie on the left side of the imaginary axis (because to make the system is to be causal and stable).

13 To Desgin IIR low pass and high pass filters

AIM: To Desgin IIR low pass and high pass filters.

APPARATUS: octave software.

SPECIFICATIONS:

Following specifications for low pass can be observed:
Passband edge: $\omega_p = 0.2\pi$.
Stopband edge: $\omega_s = 0.3\pi$.
Passband attenuation: $A_p = -1.25dB$.
Stopband attenuation: $A_s = -15dB$.
Following specifications for high pass can be observed:
Passband edge: $\omega_p = 0.3\pi$.
Stopband edge: $\omega_s = 0.2\pi$.
Passband attenuation: $A_p = -1dB$.
Stopband attenuation: $A_s = -15dB$.

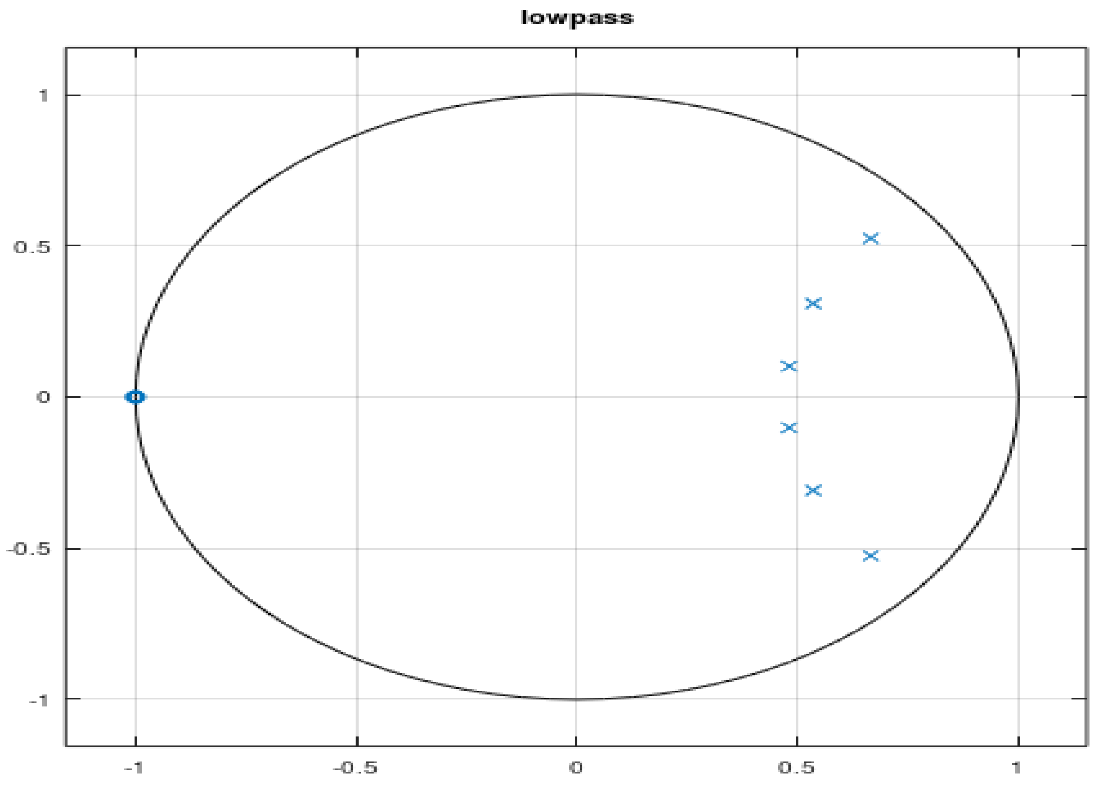
N = $\log\left(\frac{10^{|A_s|/10}-1}{10^{|A_p|/10}-1}\right)\frac{}{2\times\log\left(\frac{\Omega_s}{\Omega_p}\right)}$

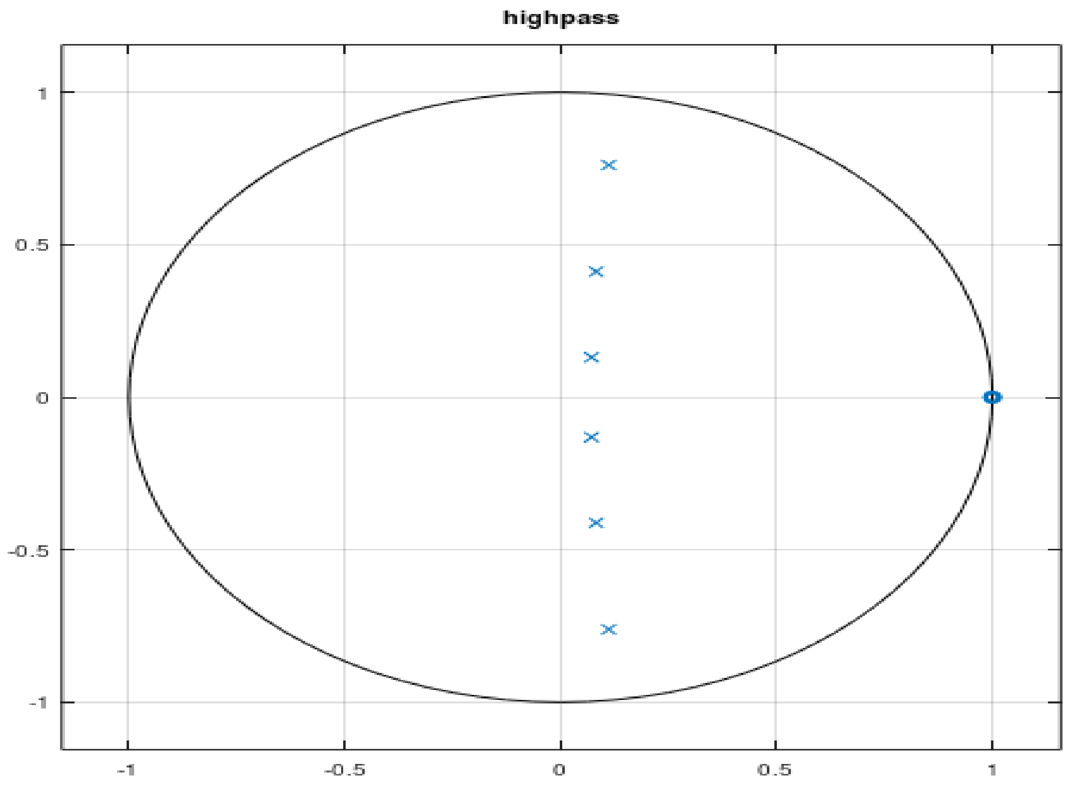
CODE:

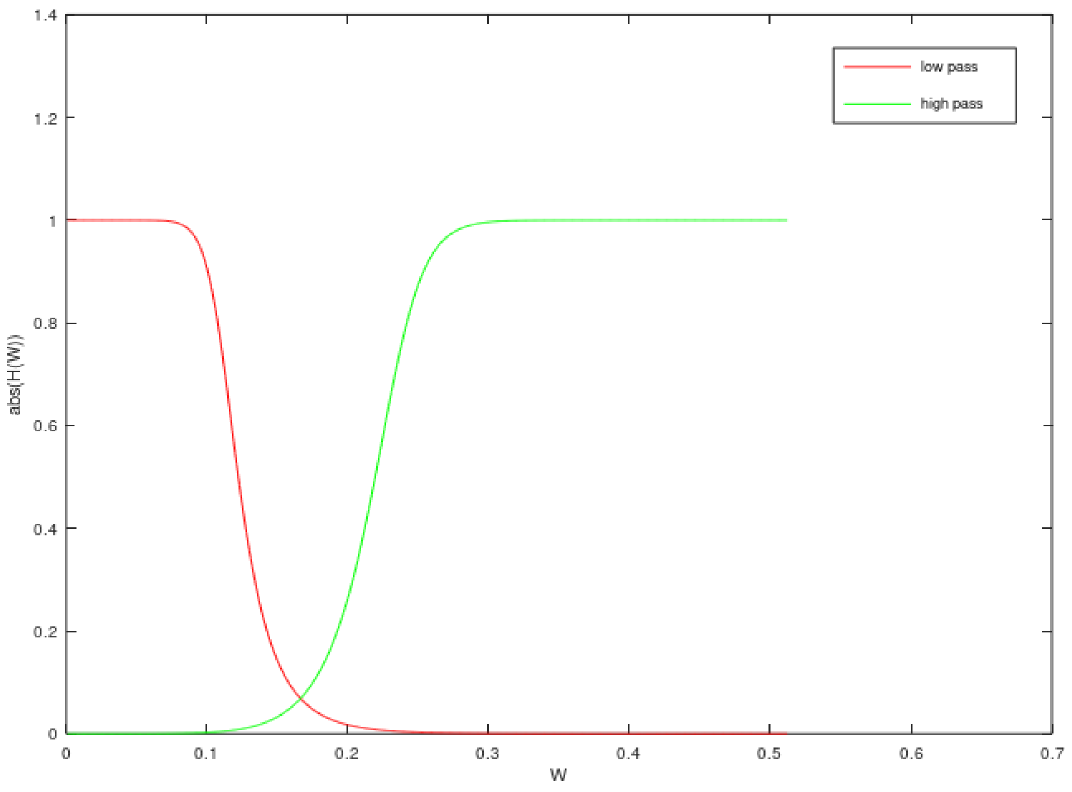
```
clc;clear;close all;
[N1,wc1]=buttord(0.2,0.3,1.25,15);
#wp,ws without pi,Ap,As without minus sign
Wc1=tan(wc1*pi/2);#captial Wc from wc.(wrap)
[b1,a1]=butter(N1,wc1);
figure(1)
zplane(b1,a1)
title('lowpass')
[N2,wc2]=buttord(0.3,0.2,1,15);
Wc2=tan(wc2*pi/2);
[b2,a2]=butter(N2,Wc2,"high");
figure(2)
zplane(b2,a2)
title('highpass')
h=freqz(b1,a1);
h2=abs(freqz(b2,a2));
figure(3)
fre=0.001:0.001:0.512;
plot(fre,abs(freqz(b1,a1)),'r');
hold on;
plot(fre,abs(freqz(b2,a2)),'g')
xlabel('W')
ylabel('abs(H(W))')
legend ('low pass','high pass');
figure(4)
plot(fre,20*log10(abs(h)),'r');
hold on;
plot(fre,20*log10(abs(h2)),'g')
xlabel('W')
```

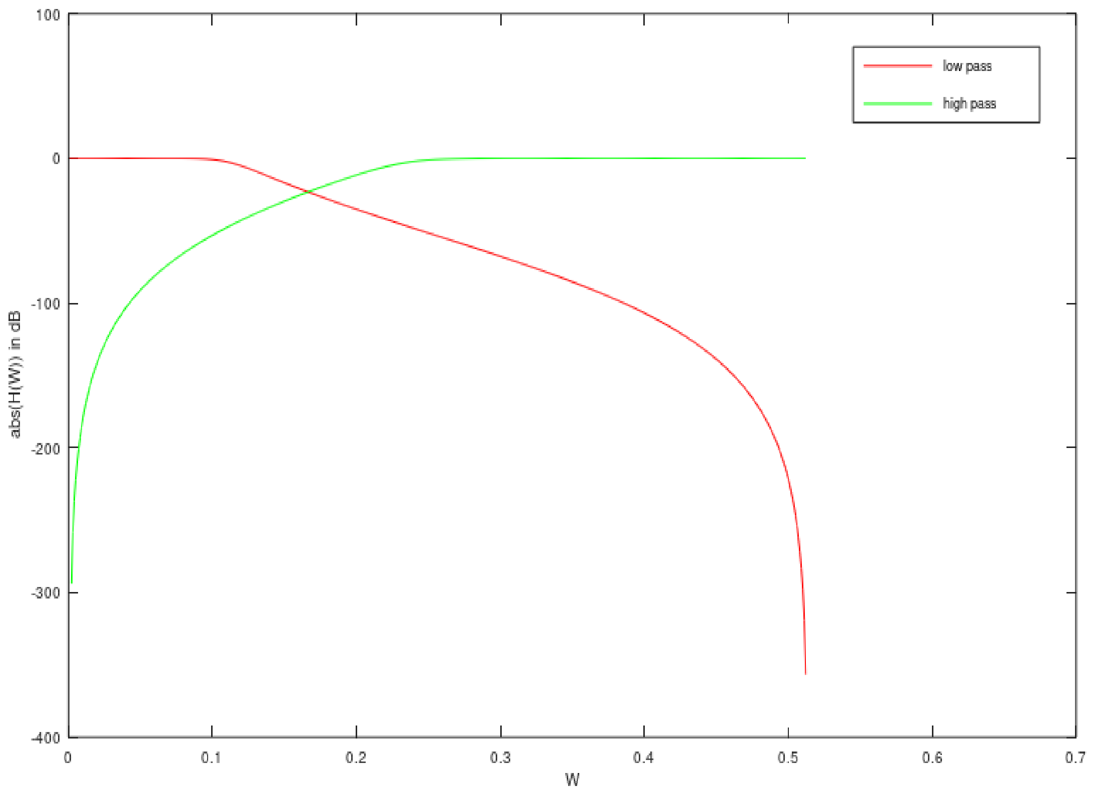
```
ylabel('abs(H(W)) in dB')
legend ('low pass','high pass');
```

GRAPH:






low pass
high pass
abs(H(W))
W


low pass
high pass
abs(H(W)) in dB
W

CONCLUSION:

- With the specifications given I follow steps such as prewrap,calculate N,denormalising,bilinear transformation calculated manually and use the in built commands such as buttord,butter,freqz for plotting graph.
- I verified the cutoff frequencies those are matched with calculated values.

14 To plot Circular Convolution

AIM: To plot Circular Convolution.

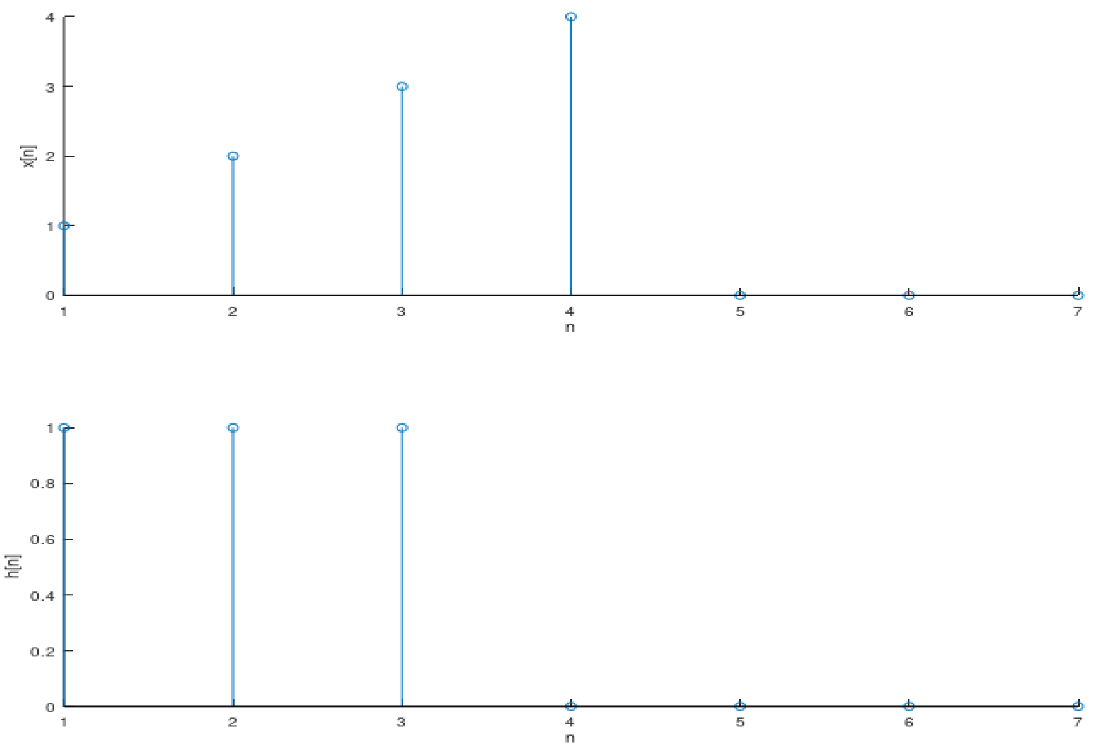
APPARATUS: octave software.

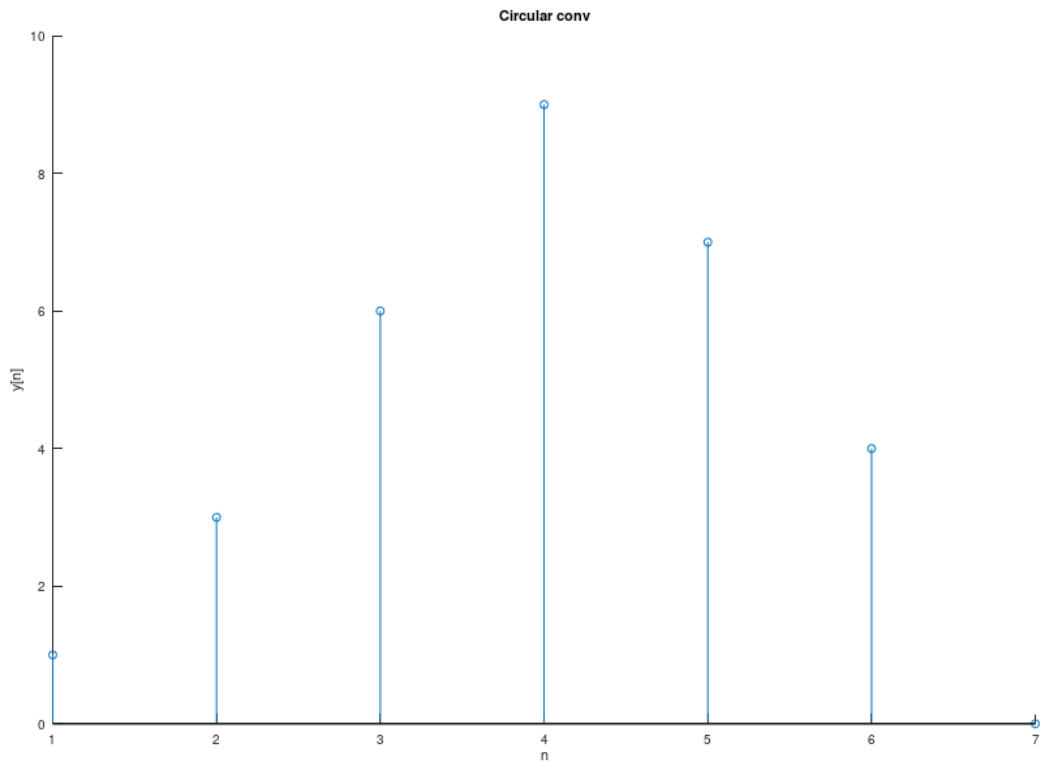
circular convolution: The N point circular convolution of two signals $x[n]$ and $h[n]$ is defined as,
y[n] = x[n] $\circledast h[n] = \sum_{k=0}^{N-1} x[k]h\left([n-k]\right)_N$

CODE:

```
clc;clear;close all;
x = [1, 2, 3, 4];
h = [1,1,1,0];
n1=length(x);
n2 = length(h);
N =n1+n2-1;% max(n1,n2)
a=1:N;
x = [x,zeros(1,N-n1)];
h = [h,zeros(1,N-n2)];
y = zeros(1,N);
for i =0:N-1
    for j = 0:N-1
        k = mod((i-j),N);%to get h index
        y(i+1) = y(i+1) + x(j+1)*h(k+1);%rotating h
    end
end
figure(1)
subplot(2,1,1)
stem(a,x)
ylabel('x[n]')
xlabel('n')
subplot(2,1,2)
stem(a,h)
ylabel('h[n]')
xlabel('n')
figure(2)
stem(a,y)
ylabel('y[n]')
xlabel('n')
title("Circular conv")
```

GRAPH:





CONCLUSION:

- I verified the output by manually also both are same.

15 To code Discrete Fourier Transform.

AIM: To code DFT.

APPARATUS: octave software.

Discrete Fourier Transform equations:

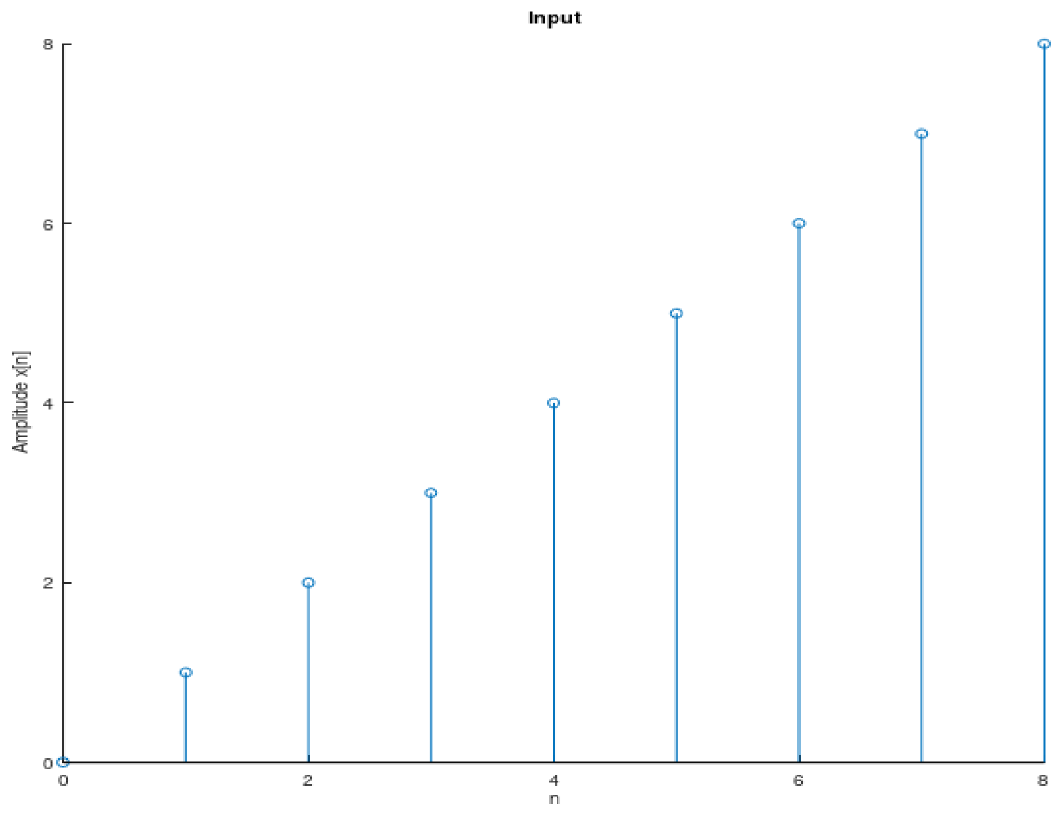
X(k) = $\sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi kn}{N}}$ $\{AnalysisEquation\}$

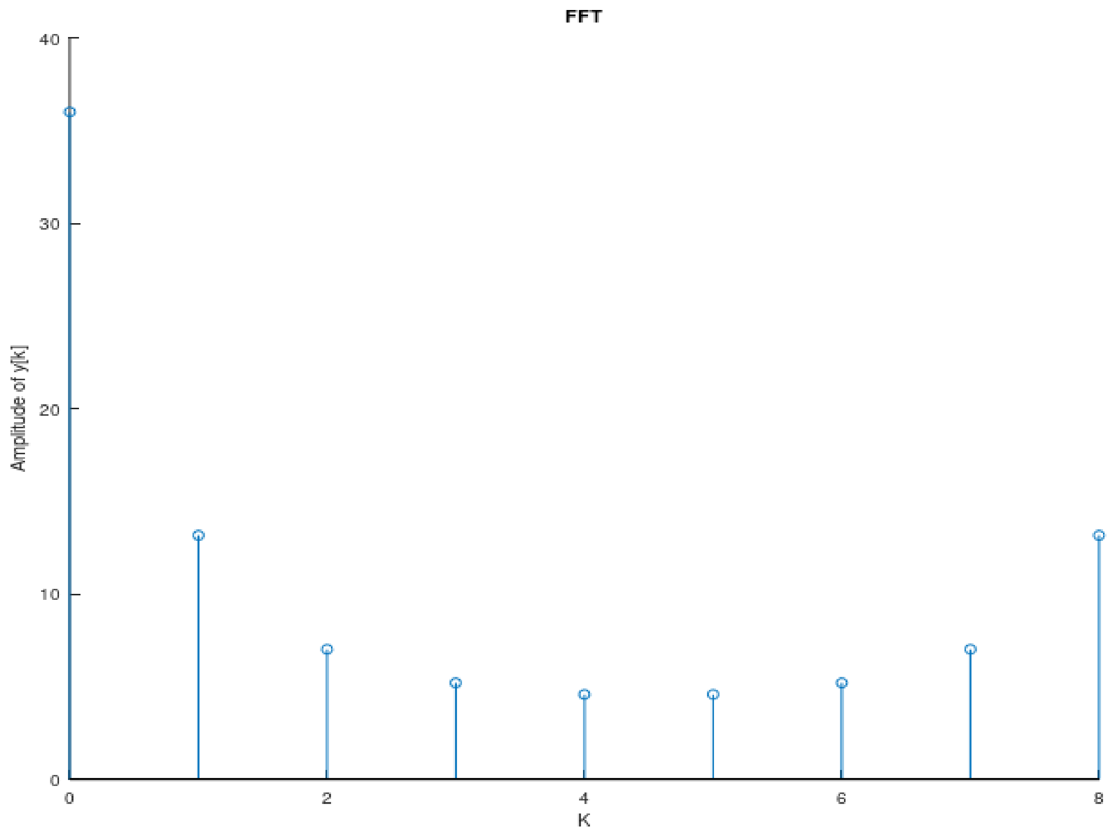
$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi kn}{N}}$ $\{SynthesisEquation\}$

CODE:

```
x=[1,2,3,4,5,6,7,8];
N=length(x);
y=zeros(1,N);
for k = 1:N %for every value of k
 y(k) = 0;
  for n = 1:N %for every value of n
     y(k) = y(k)+x(n)*exp(-1i*2*pi*(k-1)*(n-1)./N);%analysis eq
  end
end
t = 0:N-1;
figure(1)
stem(t,x); % for discrete plotting
%plotting input sequence
ylabel('Amplitude x[n]');
xlabel('n');
title('Input');

magnitude = abs(y);
figure(2)
t = 0:N-1;
stem(t,abs(y));
%plotting FFT sequence
ylabel('Amplitude of y[k]');
xlabel('K');
title('FFT');
```

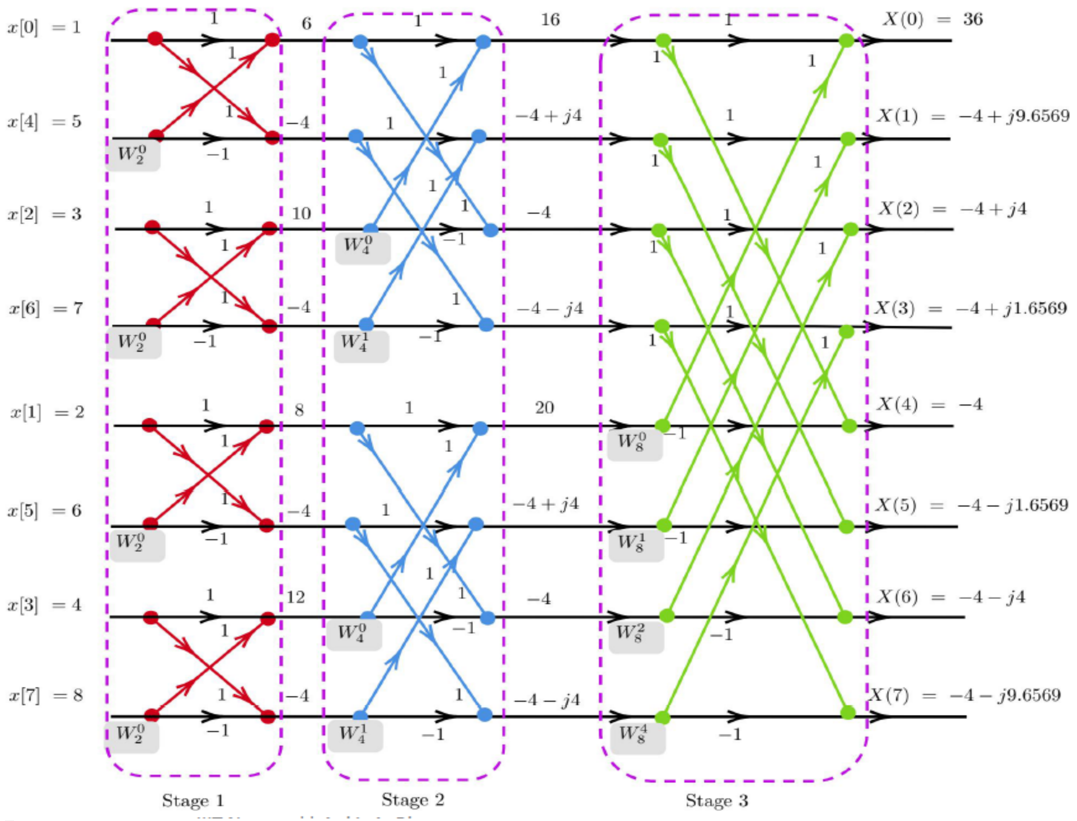
GRAPH:

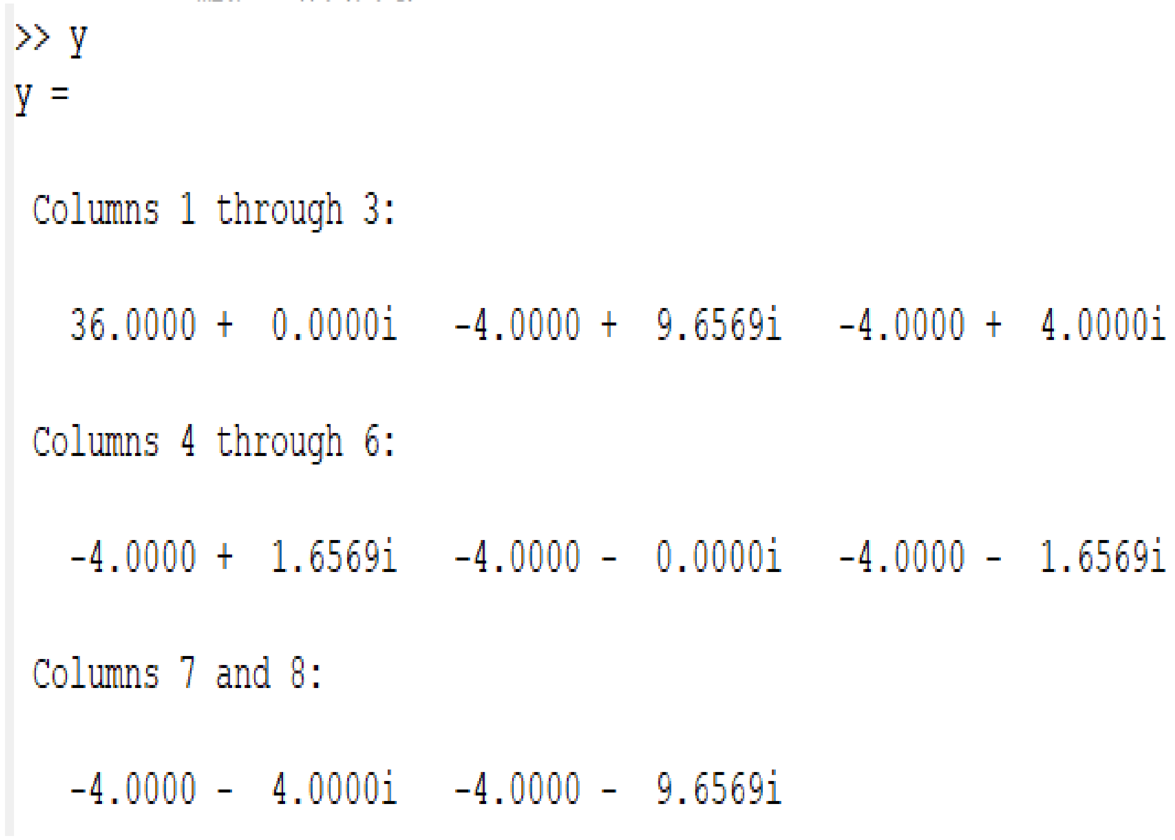




CONCLUSION:

- I implemented the code with same values taken during the class.





```
>> y
y =

 Columns 1 through 3:

   36.0000 +  0.0000i   -4.0000 +  9.6569i   -4.0000 +  4.0000i

 Columns 4 through 6:

   -4.0000 +  1.6569i   -4.0000 -  0.0000i   -4.0000 -  1.6569i

 Columns 7 and 8:

   -4.0000 -  4.0000i   -4.0000 -  9.6569i
```