Sarvajanik Education Society

Sarvajanik College of Engineering & Technology, Surat

Department of Electronics and Communication Engineering,

CERTIFICATE

This is to certify that the Lab experiments carried out by DHWANI SHAH, of B.E. 4th Sem 7th, of Electronics and Communication Engineering Department having Enrolment No: 180420111054 for the subject Digital Signal Processing, during the academic year, 2021 is found to be satisfactory.

(Prof. Maulin Joshi)	**(Ketki Pathak)**
ECC Department.	ECC Department

Date: 10/ 10/ 2019

Sarvajanik College of Engineering and Technology

Electronics and Communication Engineering Department

B.E. IV (Semester VII)

Academic Year 2021-22

Name: Dhwani Shah

Enrolment No.: 180420111054

Subject: Digital Signal Processing

Subject Code: 3171003

INDEX

Sr. No	Aim	Date	Sign
1	**Aim:** **a) Using the matlab functions plot sawtooth, square, sine, cosine, rectangular pulse, triangular pulse, sinc function and signum function. The time axis should span t = -20 to 20 and amplitude axis between -1.5 to 1.5.** **b) Also generate Digital impulse, step and ramp sequences.** **c) Implement a sinusoidal signal with 5 Hz frequency and 3 Vpp.** **d) Take one discrete signal X[n] with atleast 5 samples and perform basic operations like advancement of the signal by 3 samples, delay of the signal by 2 samples, compression of the signal by 3 factor, expansion of the signal by 2 factor and folding of signal.** **e) Perform FFT operation of given X[n] signal.**	Jun 23	
2	**Aim:** **a) Implement the convolution sum using function and without function.(x[n] ={1,2,3,4} and h[n] ={2,1,2,1} also take these two signals sine and cosine as mentioned in presentation)** **b) Perform convolution sum without inbuilt function.** **c) Check convolution properties like commutative, distributive and Associative for a given system. Homework Assignment.**	Jun 30	
3	**AIM :Write MATLAB program to check for linearity and Time Variance and stability of various system given below:** **a) Linearity: $y(n) = e^n$** **b)Time Variance: $y(n) = n^2 * \cos(\omega n)$**	Jul 5	

	c)Stability:System1 Z / (Z^2 + 0.5 Z+1)		
4	**AIM:** **(a) Determine the impulse response for the system described by second order difference equation -** **y(n)=x(n)+0.6y(n-1)-0.08y(n-2)** **(b)Also obtain the pole zero plot for the same and hence comment on the ROC and stability of the system.** **(c)Check the stability of the system analytically (hint: use equation of h(n) and comment on stability.**	Jul 14	
5	**Aim - Consider the system** $$H(z) = \frac{1-2z^{-1}+2z^{-2}-z^{-3}}{\left(1-z^{-1}\right)\left(1-0.5z^{-1}\right)\left(1-0.2z^{-1}\right)}$$ **ROC : 0.5 < |z| < 1** **a) Sketch Pole-Zero patterns. Find system is stable?** **b)Determine the impulse response of the system.** **c) Consider generalization of the system described H(z)= (1−a1* z^−1+ a2*z^−2− a3* z^−3) /(1−b1*z^−1)(1−b2* ^ z−1)(1−b3^ z−1):and think that it is all distinct pole causal system.** **You should write the code that reads coefficients b1,b2,b3 of the system. and then find the stability of the system..**	Oct 8	
6	**AIM:Consider Frequency analysis of the Amplitude Modulated Discrete-time Signal x(n) = Cos $2\pi f_1 n$ + Cos $2\pi f_2 n$** **Where f_1 = 1/128 and f_2 =5/128 modulates the Amplitude modulated signal:** **Xc(n) = Cos 2πfcn where fc = 50/128The resulting AM signal is 50/128 xam(n) = x(n)Cos 2πfcn. Using MATLAB Program:** **(a)Sketch the signals x(n), xc(n) and xam(n); 0 < n < 255** **(b)Compute and sketch the 128-point DFT of the signal** **x_{am}(n); 0 < n < 127** **(c)Compute and sketch the 128-point DFT of the signal x_{am}(n): 0 < n < 99**	Jul 28	

7	**AIM : Design Butterworth LPF for following specifications:** **Passband Attenuation : 0.4dB** **Stopband Attenuation: 35 dB** **Passband frequency : your Roll no + 400Hz** **Stopband frequency : your Roll no + 800 Hz** **Sampling frequency : 3000 Hz** **Write the Matlab program to accomplish the same.** **What changes are required to design a HPF for the same specifications? Hence design and program the same.**	Aug 2	
8	**Aim:** **In this practical you will have to design four types of filter** **1. Butterworth,** **2. Chebyshev type 1** **3. Chebyshev type 2** **4. Elliptical** **They are designed for four type of frequency response High pass, Low pass, Band pass and Band stop. Write the program for High pass and Band stop. Give necessary observations** ● **Response in terms ripple** ● **Group delay effect** ● **Comparison with Cheby type1 and Cheby type 2 band pass filter response** ● **Comparison of all high pass response, Band stop response in terms of ripple existence** ● **Overall which type is preferable? Why?**	Sep 1	
9	**Aim:** **1.** **a) Read a '.wav' file in MATLAB. Plot it in time domain and frequency domain.** **b) Apply up sampling and down sampling and plot and plot that wave file.** **c) Apply the low pass filter and high pass filter plot the corresponding waveforms in each case and listen the result of the filter.**	Sep 1	

	2. Record your own speech and also perform same a) b) and c) file for your recorded version. In this file all the students are requested to record your own speech as follows "I am XYZ(your name) . I am doing waveread command for speech signal in DSP lab"		
10	**Aim:** **Design a 25-tap Bandpass Filter for the following specifications:** ● **Cut-off frequencies: 0.25pi and 0.75pi** ● **Windowing: Rectangular and Hamming** ● **Order: 25** **Compare different Windowing Techniques and comment on the same.**	Sep 8	
12	**AIM - Plot the Frequency response of LPF and HPF (with the following specifications) using different windowing technique especially Keiser Window for different values of β.** **Cut-off frequency: (Your Roll no/100) * pi - 0.6908** **Order of the filter N: 25** **Beta: 0.5, 3.5, 8.5**	Sep 15	

AIM : (a) Using the Matlab functions plot sawtooth, square, sine, cosine, rectangular pulse, triangular pulse, sinc function and signum function. The time axis should span t = -20 to 20 and amplitude axis between -1.5 to 1.5.

PROGRAM :

```
clc;
clear all;
close all;

%a)Using the Matlab functions plot sawtooth, square,
%sine, cosine, rectangular pulse, triangular pulse, sinc
%function and signum function. The time axis should
%span t = -20 to 20 and amplitude axis between -1.5 to 1.5.

f = input('Enter frequency: ')
A = input('input amplitude: ')
t = input('Enter time range for continuous: ');
%p = input('Enter time range for discrete: ');
x = -t:0.1:t;

y = A*sin(2*pi*f*x);
figure;
subplot(2,1,1);
plot(x,y);
title('SINE CONTINUOUS');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
%z = -p:0.1:p;
%q = A*sin(2*pi*f*z);
subplot(2,1,2);
stem(x,y);
title('SINE DISCRETE');
```

```
xlabel('Time');
ylabel(‘Amplitude');
axis ([-t t -A A]);
b = square(x);
figure;
subplot(2,1,1);
plot(x,b);
title('Square Continuous');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
subplot(2,1,2);
stem(x,b);
title('Square Discrete'); xlabel('Time');
ylabel('Amplitude'); axis ([-t t -A A]);
r = cos(x);
figure;
subplot(2,1,1);
plot(x,r);
title('Cosine Continuous'); xlabel('Time');
ylabel('Amplitude'); axis ([-t t -A A]);
subplot(2,1,2);
stem(x,r);
title('Cosine Discrete'); xlabel('Time');
ylabel('Amplitude'); axis ([-t t -A A]);

s = sawtooth(x);
figure;
subplot(2,1,1);
plot(x,s);
title('Sawtooth Continuous');
```

```
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
subplot(2,1,2);
stem(x,s);
title('Sawtooth Discrete');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);

d = rectpuls(x);
figure;
subplot(2,1,1);
plot(x,d);
title('Rectangular pulse Continuous');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
subplot(2,1,2);
stem(x,d);
title('Rectangular pulse Discrete'); xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);

e = tripuls(x);
figure;
subplot(2,1,1);
plot(x,e);
title('Triangular pulse Continuous');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
subplot(2,1,2);
```

```
stem(x,e);
title('Triangular pulse Discrete'); xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);

c = sinc(x);
figure;
subplot(2,1,1);
plot(x,c);
title('Sinc function Continuous'); xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
subplot(2,1,2);
stem(x,c);
title('Sinc function Discrete'); xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);

h = sign(x);
figure;
subplot(2,1,1);
plot(x,h);
title('Signum function Continuous');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
subplot(2,1,2);
stem(x,h);
title('Signum function Discrete');
xlabel('Time');
ylabel('Amplitude');
axis ([-t t -A A]);
```

OUTPUT:

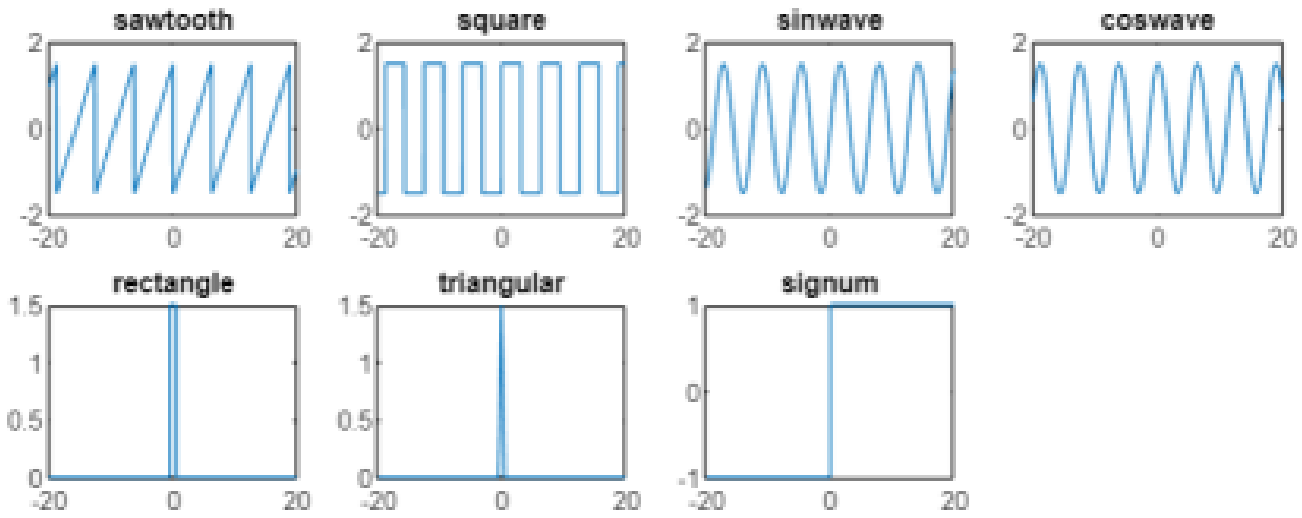
Enter frequency: 1

f = 1

input amplitude: 1.5

A = 1.5000

Enter time range for continuous: 20



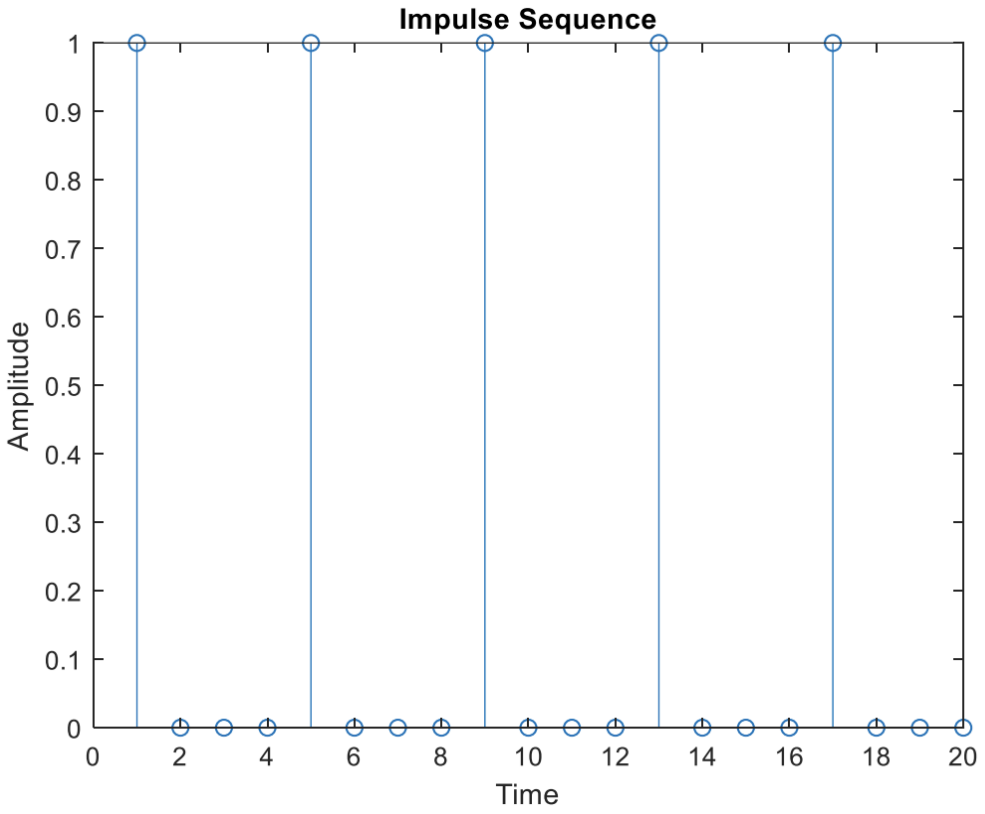
(b) Also generate Digital impulse, step and ramp sequences.

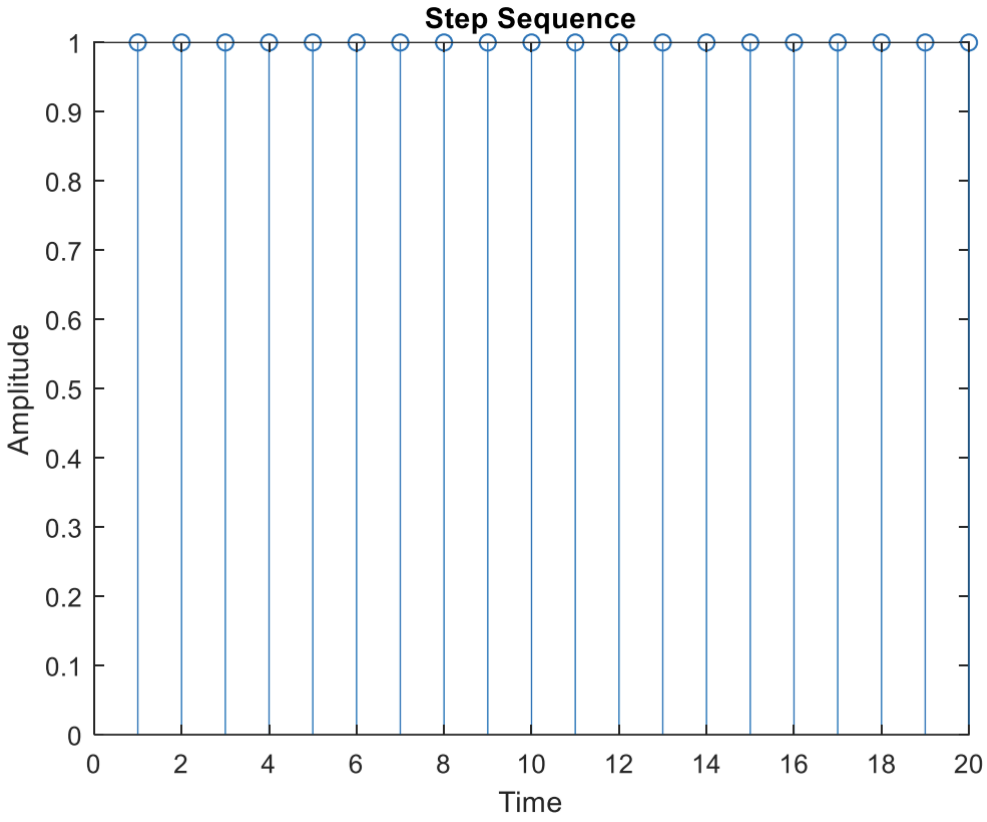
PROGRAM:

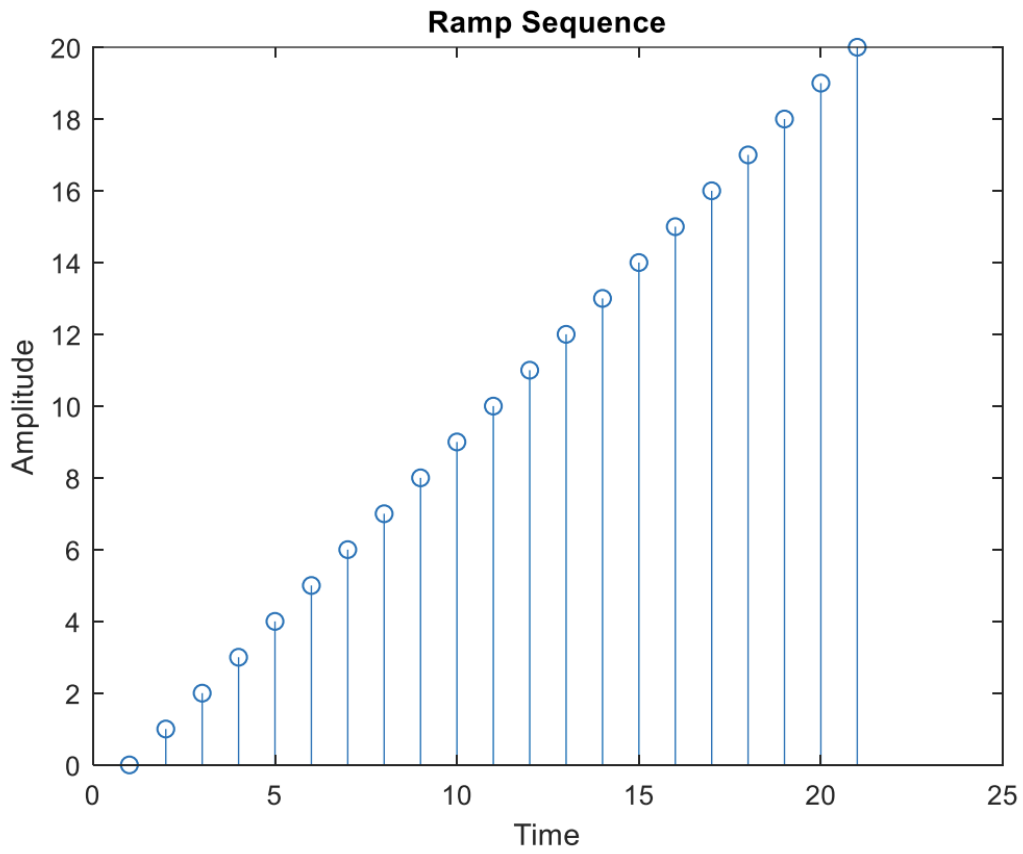
```
clc;
clear all;
close all;
a = input('Enter time');
t = -a:1:a;
b = 1;
imp = b*(t==0);
stem(t,imp);
```

```
subplot(3,1,1);
xlabel('Time');
ylabel('Amplitude');
title('Unit Impulse');

unit = 1*(t>=0);
stem(t,unit);
subplot(3,1,2);
xlabel('Time');
ylabel('Amplitude');
title('Unit Step');

ramp = t.*(t>=0);
stem(t,ramp);
subplot(3,1,3);
xlabel('Time');
ylabel('Amplitude');
title('Unit Ramp');
```

OUTPUT:


Impulse Sequence
Amplitude
Time
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14
16
18
20


Step Sequence
Amplitude
Time
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12
14
16
18
20



C) Implement a sinusoidal signal with 5 Hz frequency and 3 Vpp

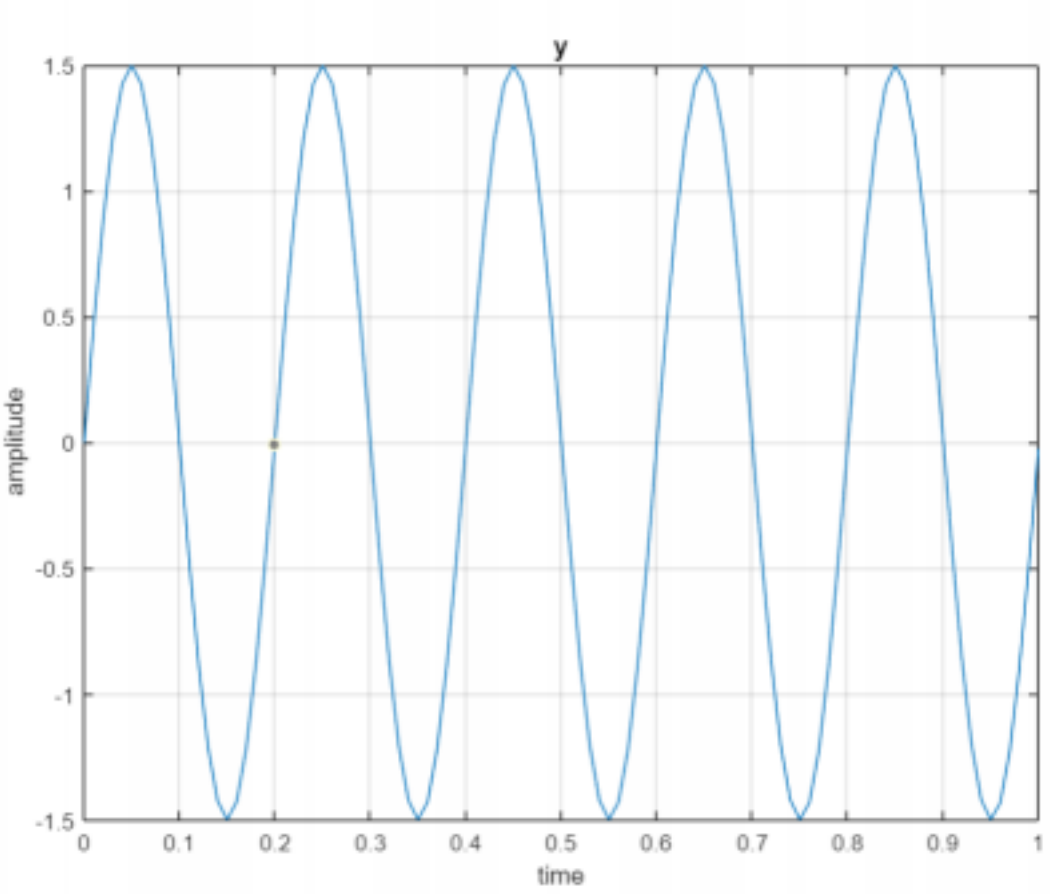
PROGRAM:

```
clc;
clear all;
close all;
a = input('value of amplitude : ');
f = input('value of frequency : ');
t = 0:0.01:2*pi*f;
y = a*sin(2*pi*f*t);
plot(t,y)
axis([0 1 -a a]);
xlabel('time');
ylabel('amplitude');
title('Sine Signal');
grid on;
```

OUTPUT :

value of amplitude : 1.5

value of frequency : 5



(d) Take one discrete signal x(n) with atleast5 samples and perform basic operations like advancement of the signal by 3 samples, delay of the signal by 2 samples, compression of the signal by 3 factor, expansion of the signal by 2 factor, folding of signal.

PROGRAM:

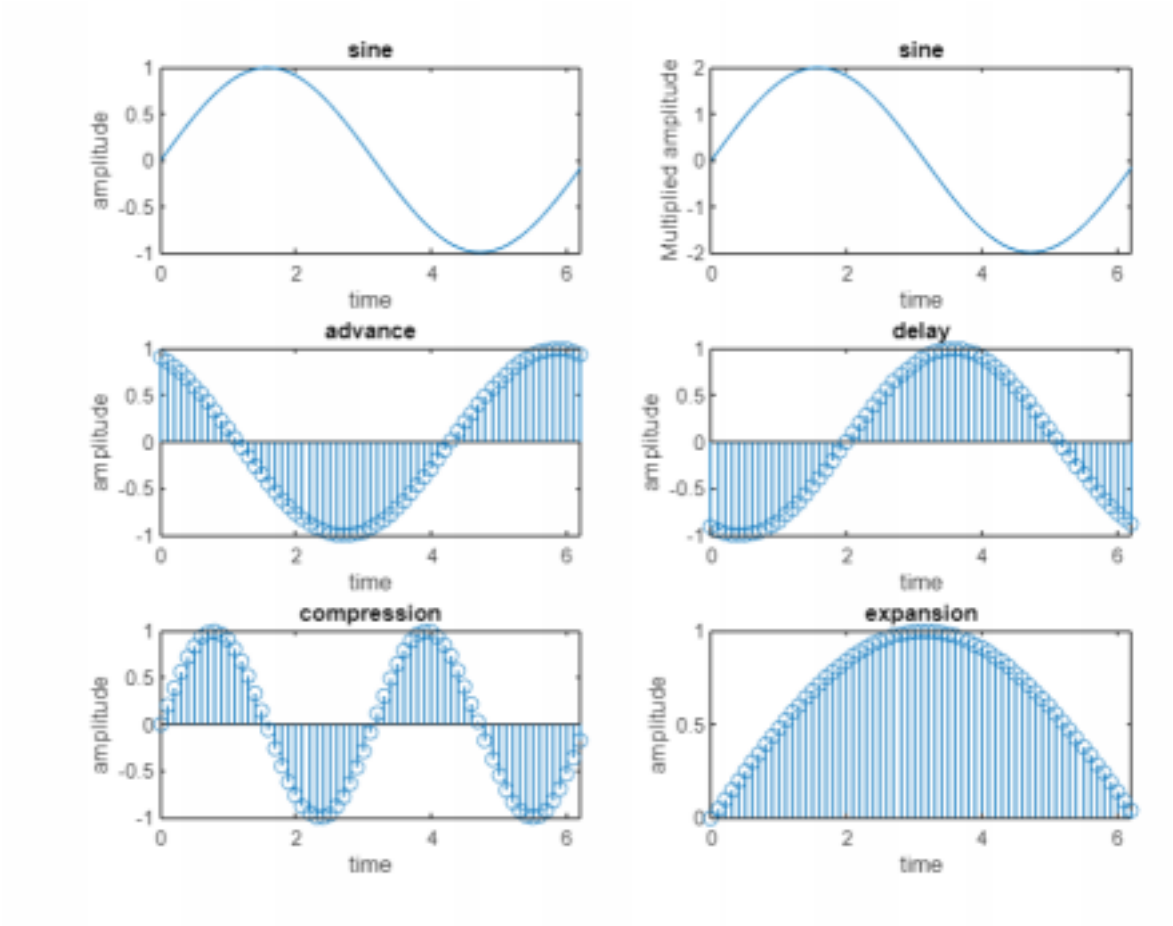
```
clc;
clear all;
close all;
```

```
a = input('Operate signal change by: ');
t = 0:0.1:(2*pi);
y = a*sin(t);
subplot(3,2,2)
plot(t,y)
xlabel('time')
ylabel('Multiplied amplitude')
title('sine')

subplot(3,2,1)
plot(t,sin(t))
xlabel('time')
ylabel('amplitude')
title('sine')

subplot(3,2,3)
stem(t,sin(t+a))
xlabel('time')
ylabel('amplitude')
title('advance')

subplot(3,2,4)
stem(t,sin(t-a))
xlabel('time')
ylabel('amplitude')
title('delay')
subplot(3,2,5)
stem(t,sin(t*a))
xlabel('time')
ylabel('amplitude')
title('compression')
```

```
subplot(3,2,6)
stem(t,sin(t/a))
xlabel('time')
ylabel('amplitude')
title('expansion')
```

OUTPUT :

Operate signal change by: 2

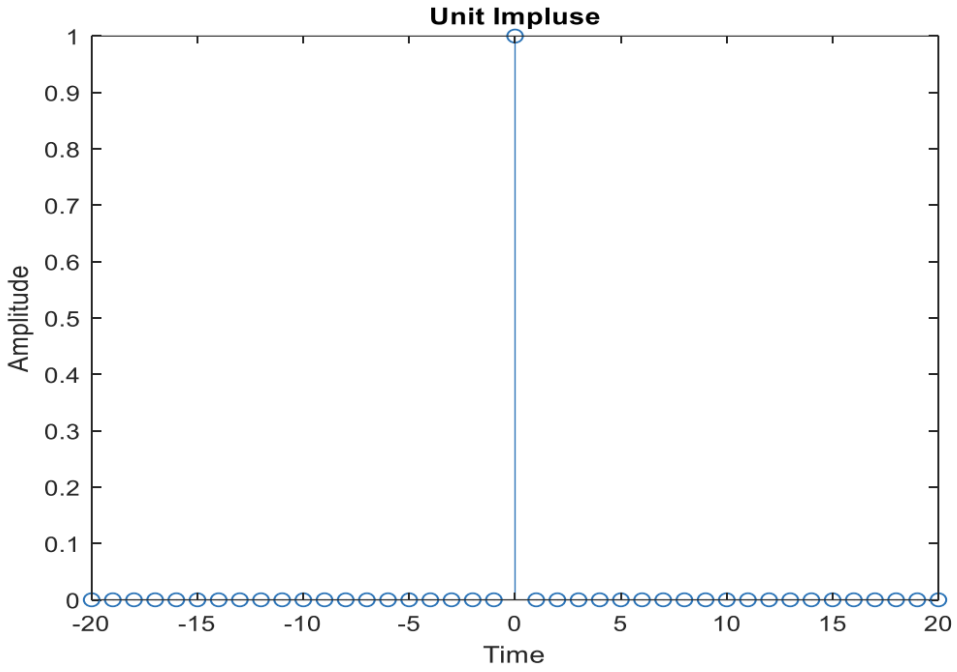


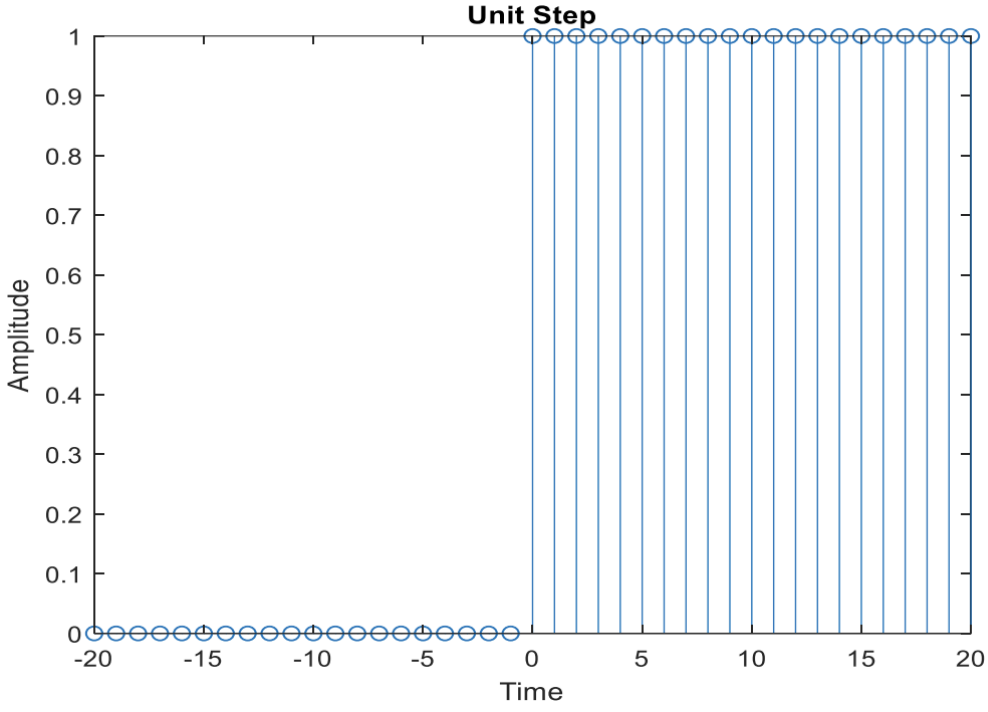
(E) Apply fft function on the signal and comment on observed waveform

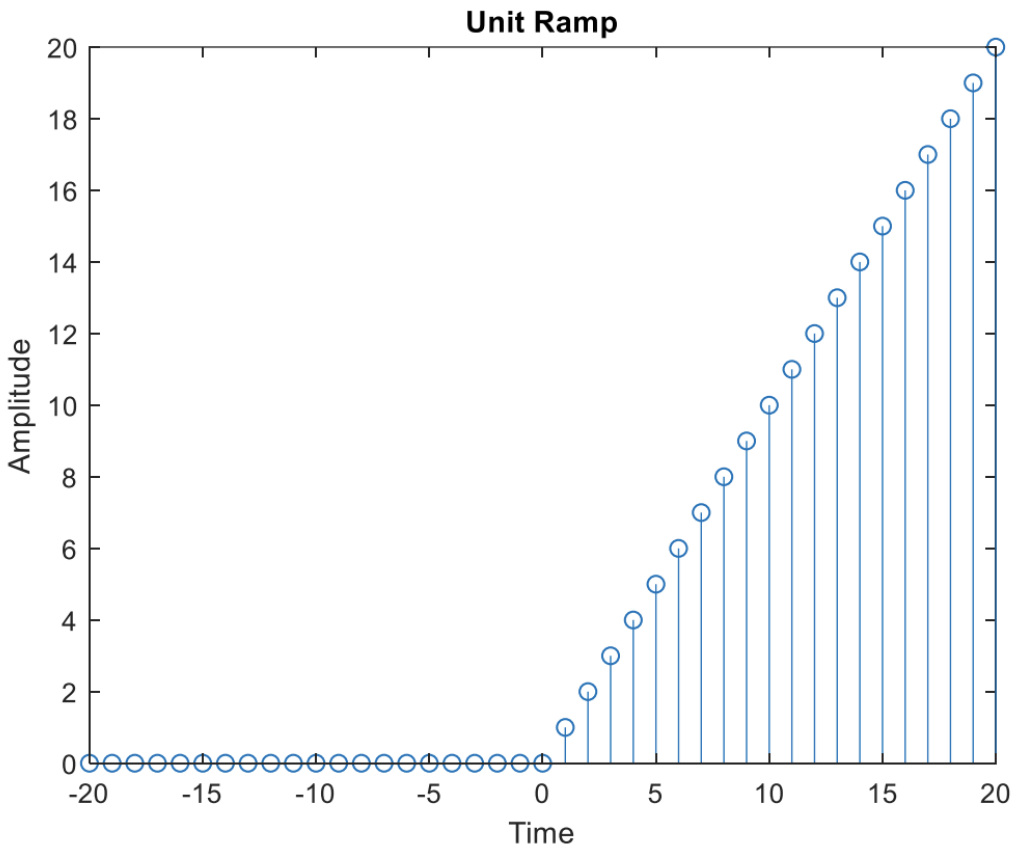
PROGRAM:

```
clc;
clear all;
close all;

a = input('Enter time');
t = -a:1:a;
b = 1;
imp = b*(t==0);
stem(t,imp);
subplot(3,1,1);
xlabel('Time');
ylabel('Amplitude');
title('Unit Impulse');

unit = 1*(t>=0); stem(t,unit);
subplot(3,1,2);
xlabel('Time');
ylabel('Amplitude');
title('Unit Step');

ramp = t.*(t>=0);
stem(t,ramp);
subplot(3,1,3);
xlabel('Time');
ylabel('Amplitude');
title('Unit Ramp');
```

OUTPUT:


Unit Impluse
Amplitude
Time
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-20
-15
-10
-5
0
5
10
15
20


Unit Step
Amplitude
Time
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-20
-15
-10
-5
0
5
10
15
20



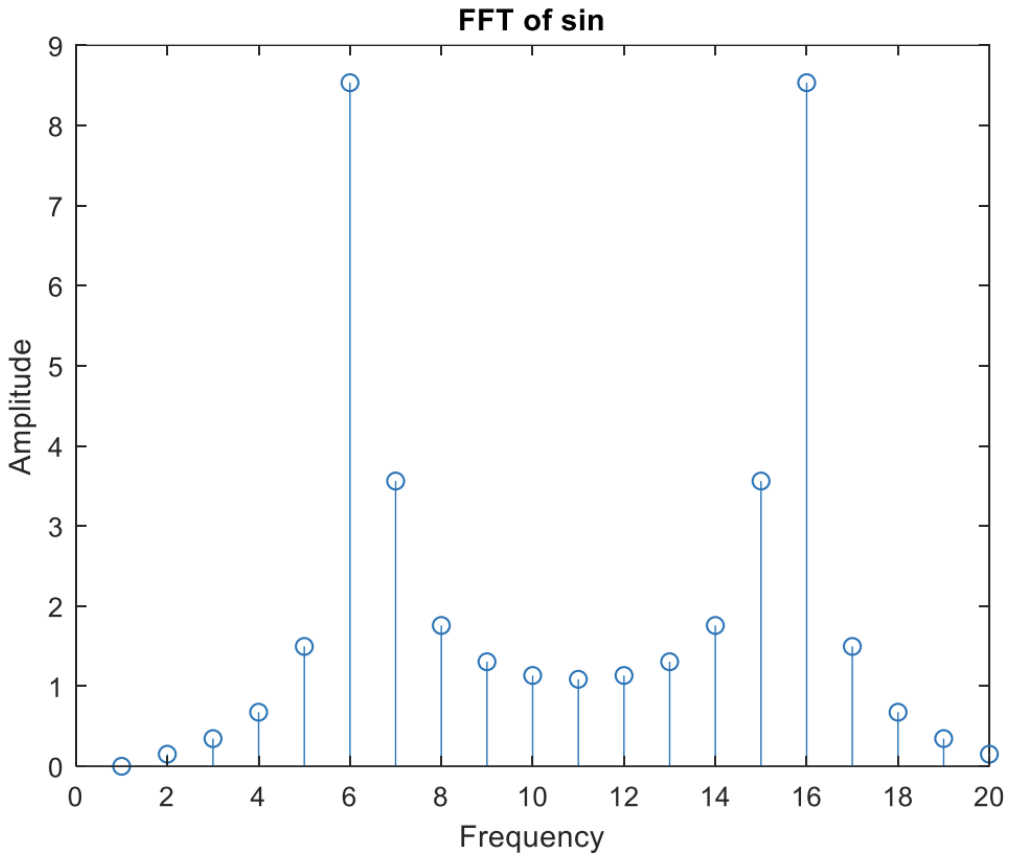
(F) Apply fft function on the signal and comment on observed waveform

PROGRAM:

```
clc;
clear all;
close all;


f = input('Enter frequency');
a = input('Enter Amplitude');
s = 1/f;
t = linspace(0,1,20);
y = sin(2*pi*f*t);
yfdt = fft(y);
stem(abs(yfdt));
```

```
xlabel('Frequency');
ylabel('Amplitude');
title('FFT of sin');
```

OUTPUT:



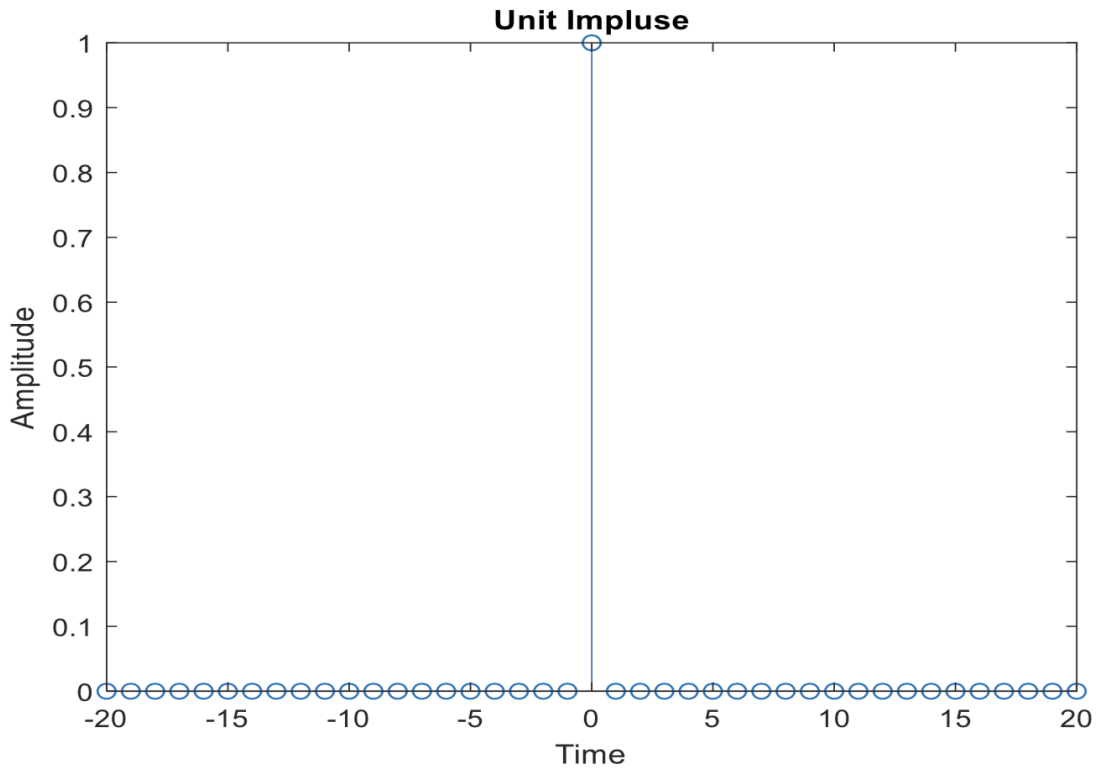
(remaining parts of practical : 1)

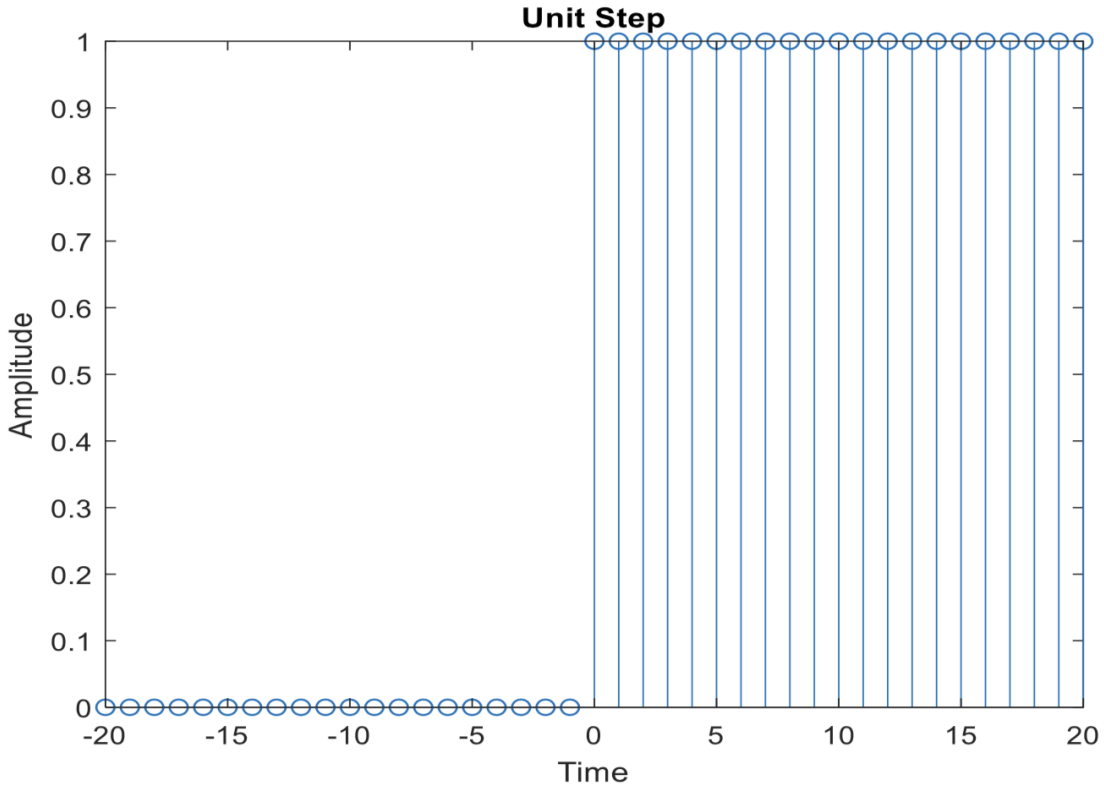
(E) Implement basic unit signals like unit impulse, unit step unit ramp

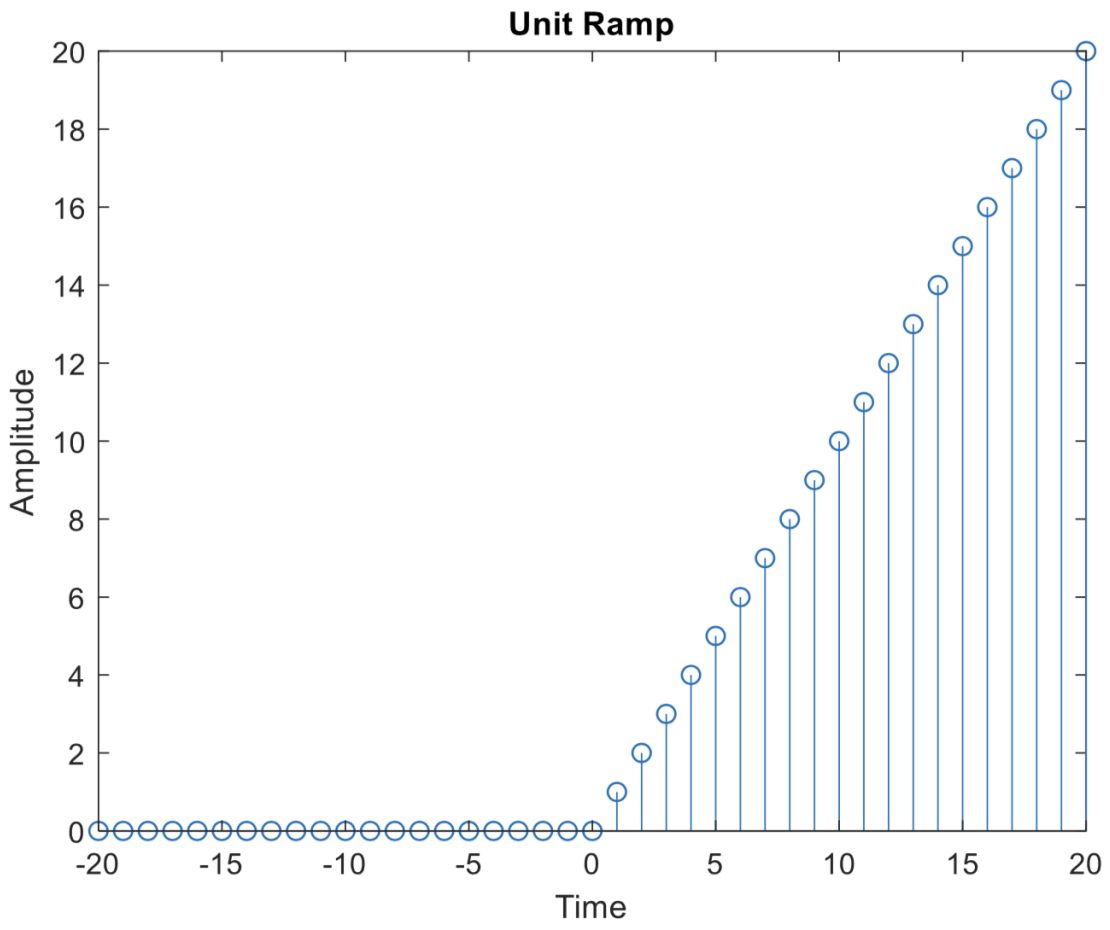
PROGRAM:

```
clc;
clear all;
close all;

a = input('Enter time');
t = -a:1:a;
b = 1;
imp = b*(t==0);
stem(t,imp);
subplot(3,1,1);
xlabel('Time');
ylabel('Amplitude');
title('Unit Impulse');

unit = 1*(t>=0);
stem(t,unit);
subplot(3,1,2);
xlabel('Time');
ylabel('Amplitude');
title('Unit Step');

ramp = t.*(t>=0);
stem(t,ramp);
subplot(3,1,3);
xlabel('Time');
ylabel('Amplitude');
title('Unit Ramp');
```

OUTPUT:


Unit Impluse
Amplitude
Time
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-20
-15
-10
-5
0
5
10
15
20


Unit Step
Amplitude
Time
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-20
-15
-10
-5
0
5
10
15
20



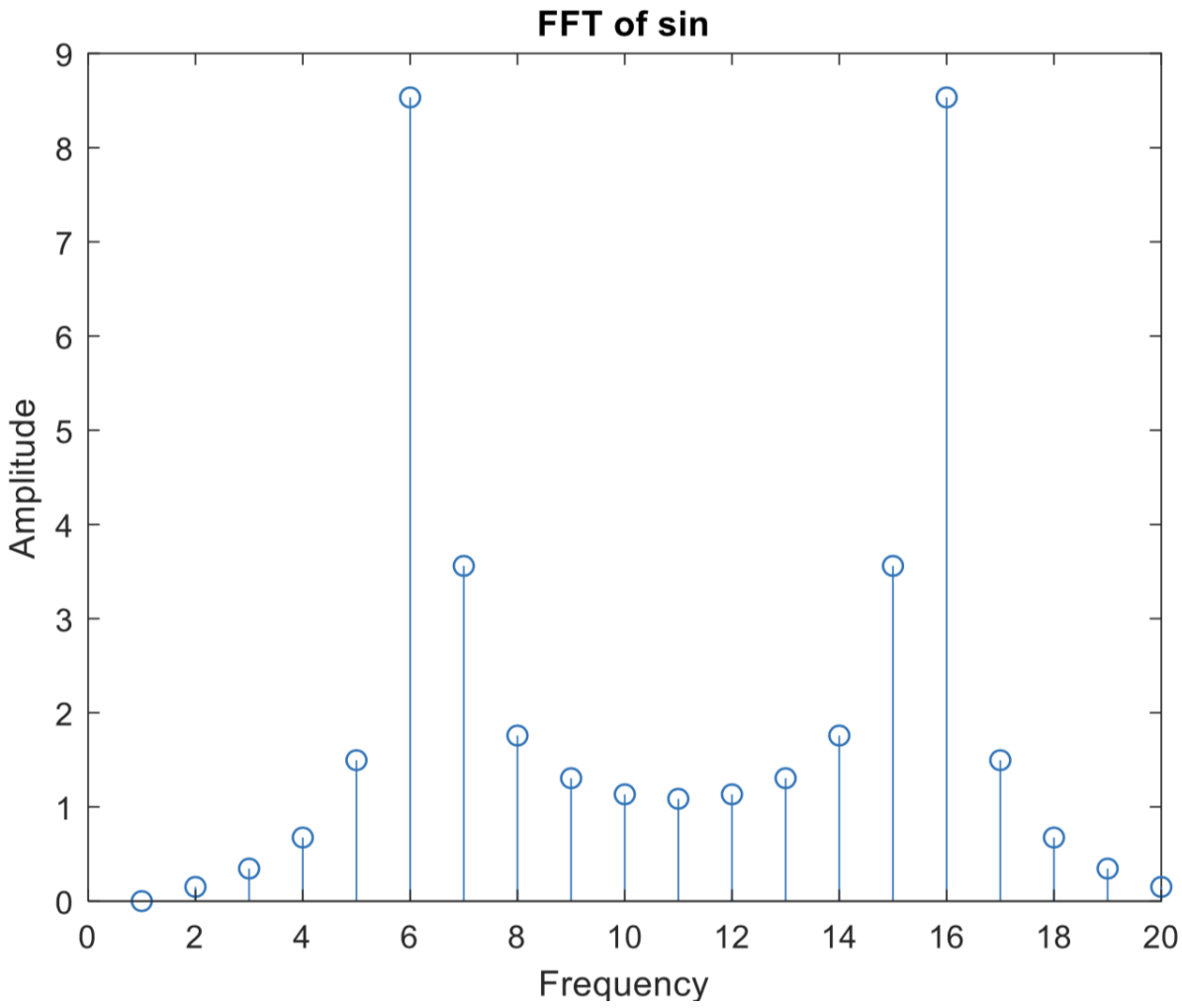
(F) Apply fft function on the signal and comment on observed waveform

PROGRAM:

```
clc;
clear all;
close all;

f = input('Enter frequency');
a = input('Enter Amplitude');
s = 1/f;
t = linspace(0,1,20);

y = sin(2*pi*f*t);
yfdt = fft(y);
stem(abs(yfdt));
```

```
xlabel('Frequency');
ylabel('Amplitude');
title('FFT of sin');
```

OUTPUT:



Practical : 2

(a) Implement convolution sum for two finite sequence using inbuilt MATLAB function conv (equal length)

(b) Implement convolution sum for two finite sequence using inbuilt MATLAB function conv (unequal length)

PROGRAM :

(a)

```
clc;

clear all;

close all;

%convolution sum for two finite sequences using inbuilt MATLAB
function conv.

display('Length of both signal should be same');

x = input('Enter a Finite sequence x[n] ');

h = input('Enter a finite sequence h[n] ');

m = length(x);

n = length(h);

subplot(3,1,1);


stem(x);

title('x[n]');

axis([0 10 0 8]);

grid on;
```

```
subplot(3,1,2);
stem(h);
title('h[n]');
axis([0 10 0 8]);
grid on
if (m==n)
    y = conv(x,h);
    subplot(3,1,3);
    stem(y);
    title('Convolution Sum y[n]');
    grid on;
else
   display('Length is not same');
end
```

OUTPUT:

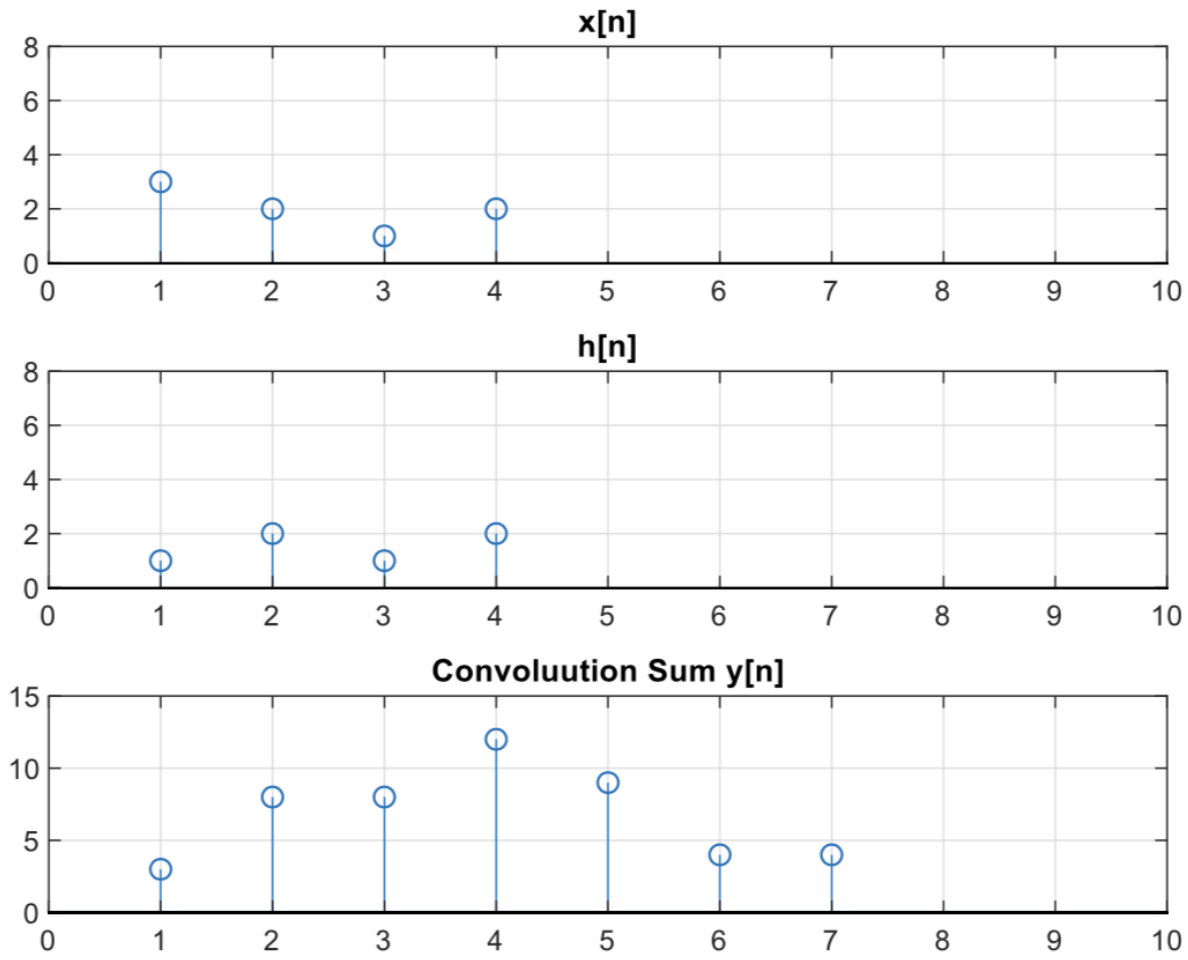
Length of both signal should be same

Enter a Finite sequence x[n] [3 2 1 2]

Enter a finite sequence h[n] [1 2 1 2]

y=

3 8 8 12 9 4 4



PROGRAM :

(b)

```
clc;

clear all;

close all;

%b. convolution sum for two finite sequence using inbuilt MATLAB
function conv ( unequal length )

m = input('Enter length of x[n] ');

x = input('Enter signal x[n] ');
```

```
if     (length(x) == m)
       subplot(3,1,1);
       stem(x);
       title('x[n]');
       grid on;
else
   display('Length should be same as specify for x[n]');
end
n = input('Enter length of h[n] ');
h = input('Enter signal h[n] ');
if (length(h) == n)
    subplot(3,1,2);
    stem(h);
    axis([0 n 0 10]);
    title('h[n]');
    grid on;
else
   display('Length should be same as specify for h[n]');
end
if (m == n)
    display('Length should not be same');
else
   y = conv(x,h);
```

```
    subplot(3,1,3);
    display(y);
    stem(y);
    title('Convolution Sum y[n]');
    grid on;
end
```

OUTPUT:

Enter length of x[n] 4

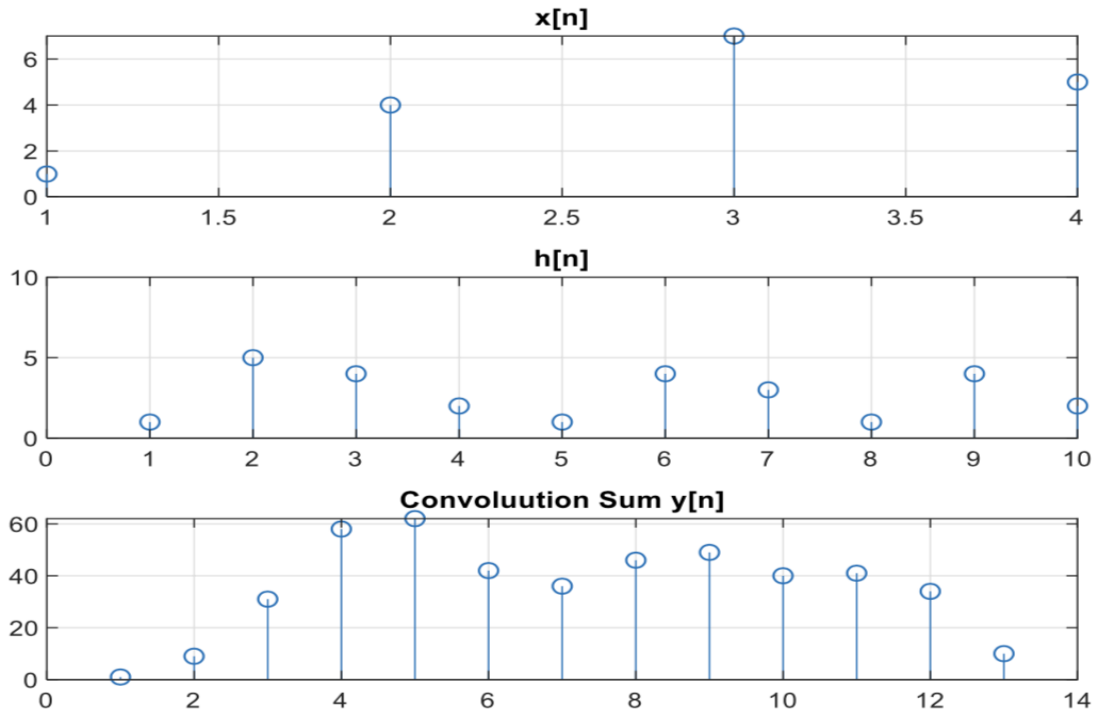
Enter signal x[n] [1 4 7 5]

Enter length of h[n] 10

Enter signal h[n] [1 5 4 2 1 4 3 1 4 2]

y=

1 9 31 58 62 42 36 46 49 40 41 34 10



LAB 3

AIM : Write MATLAB program to check for linearity and Time Variance and stability of various system given below:

a) Linearity: y(n) = en

b) Time Variance: y(n) = n2*cos(ώn) c)Stability:System1 Z / (Z2 + 0.5 Z+1)

PROGRAM:

```
clc;
clear all;
close all;

% Linearity y(n) = e^(x(n))
n = 0:1:50;
x1n = input('enter x1(n) ');
x2n = input('enter x2(n) ');
a = input('Enter weight of x1 ');
b = input('Enter weight of x2 ');
y1 = exp(x1n);
y2 = exp(x2n)

%Weighted sum of output is yw1
yw1 = a*y1+b*y2;

% Output due to weighted sum of input is yw2

yw2 = exp(a*x1n+b*x2n);
if (yw1 == yw2)
   disp('System is linear ')
else
   disp('System is Nonlinear')
end

%Time Variance: y(n) = n^2*cos(omega*n)

n = 0:1:50;
xn = cos(n);
yn = (n.^2).*xn;
```

```
k = input('Enter delay units k ')

% output due to input delayed by k units

x1 = cos(n-k);
y1 = (n.^2).*x1;

%output delayed by k units

y2 = ((n-k).^2).*x1;
if (y1==y2)
   disp('System is time invariance')
else
   disp('System is Time variance')
end

%%Stability: Z / (Z^2 + 0.5*Z+1)

d = [1 0.5 1]
r = roots(d)
for i = 1:2
    if (r(i)<1)
        disp('System is stable')
    else
       disp('System is Unstable')
    end
end
```

OUTPUT:

enter x1(n) 5*n

enter x2(n) n+1

Enter weight of x1 3

Enter weight of x2 2

System is Nonlinear

Enter delay units k 6

K = 6

System is Time variance

d = 1.0000 0.5000 1.0000

r = -0.2500 + 0.9682i , -0.2500 - 0.9682i

System is stable

THEORETICAL CALCULATION:

Dhwani Shah.
180420111054

DSP ~~Lab 3~~

Lab 3

a) Linearity : $y(n) = e^{x(n)}$

let $y_1(n) = T[x_1(n)] = e^{x_1(n)}$
$y_2(n) = T[x_2(n)] = e^{x_2(n)}$

Weighted sums of o/p is (y')
$= ay_1(n) + by_2(n) = a\,e^{x_1(n)} + be^{x_2(n)}$
$= y'$

o/p due to weighted sum of inputs

$y''(n) = T[ax_1(n) + bx_2(n)]$
$= e^{ax_1(n) + bx_2(n)}$

$\therefore y'(n) \neq y''(n)$

$\therefore$ Non linear system

b) Time variance : $y(n) = n^2 \cos(\omega n)$

$y(n) = n^2 \cos(\omega n) = T[x(n)]$
let $x(n) = \cos(\omega n) = T[x(n)]$
let $x(n) = \cos \omega n$

$\therefore$ o/p due to input delayed by k units

$y(n,k) = y(n)\big|_{x(n) = x(n-k)}$
$= n^2 \cos(\omega(n-k))$

o/p delayed by k units

$y(n-k) = y(n)\Big|_{n=n-k}$

$= (n-k)^2 \cos(w(n-k))$

$\therefore y(n,k) \neq y(n-k)$

$\therefore$ System is time variance

c) Stability

$$y(z) = \frac{z}{z^2 + 0.5z + 1}$$

roots : $z^2 + 0.5z + 1 = 0$

$$r_1, r_2 = \frac{b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$= \frac{0.5 \pm \sqrt{0.25 - 4}}{2}$$

$r_1 = -0.25 + 0.9682\,j$

$r_2 = -0.25 - 0.9682\,j$

$\therefore r_1$ and $r_2 < 1$

$\therefore$ The system is stable.

Conclusion:

Thus, in this lab we checked linearity, stability and time variance of the given signals and verified it theoretically.

PRACTICAL – 4

AIM : (a) Determine the impulse response for the system described by second order difference equation - y(n)=x(n)+0.6y(n-1)-0.08y(n-2)

(b) Also obtain the pole zero plot for the same and hence comment on the ROC and stability of the system.

(c) Check the stability of the system analytically (hint: use equation of h(n) and comment on stability)

THEORETICAL CALCULATION:

LAB-4

$\Rightarrow$ $y(n) = x(n) + 0.6y(n-1) - 0.08y(n-2)$

$\therefore y(n) - 0.6y(n-1) + 0.08y(n-2) = x(n)$

→ Convert z-transform.

$Y(z) - 0.6z^{-1}Y(z) + 0.08z^{(-2)}Y(z) = X(z)$

$\therefore Y(z)[1 - 0.6z^{-1} + 0.08z^{-2}] = X(z)$

$\rightarrow \frac{X(z)}{Y(z)} = (1 - 0.6z^{-1} + 0.08z^{-2})$

$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 - 0.6z^{-1} + 0.08z^{-2}}$

$H(z) = \frac{z^2}{z^2 - 0.6z + 0.08}$ (Multiply & divide by z^2)

$\frac{H(z)}{z} = \frac{z}{z^2 - 0.6z + 0.08}$

$= \frac{z}{(z-0.4)(z-0.2)}$

$\therefore \frac{H(z)}{z} = \frac{A}{z-0.4} + \frac{B}{z-0.2}$

$A = \left.\frac{z(z-0.4)}{(z-0.4)(z-0.2)}\right|_{z=0.4}$

$= 2$

$$B = \left.\frac{z(z-0.2)}{(z-0.4)(z-0.2)}\right|_{z=0.2}$$

$$= (-1)$$

$$\therefore \frac{H(z)}{z} = \frac{2}{z-0.4} + \frac{(-1)}{z-0.2}$$

$$H(z) = \frac{2z}{z-0.4} + \frac{(-z)}{z-0.2}$$

Taking inverse Z transform,

$$h(n) = 2(0.4)^n u(n) - (0.2)^n u(n)$$

⇒ ROC:-

$|z| > 0.4$

So output is Causal Sequence

$x(n) = 2(0.4)^n u(n) - (0.2)^n u(n)$

- for ROC:- $|z| < 0.2$

So Sequence is anti-causal

$x(n) = 2\left[-(0.4)^n u(-n-1)\right] + (0.2)^n u(-n-1)$

- for ROC:- $0.4 > |z| > 0.2$

· for this first term is anticausal and second term will be causal

$$x(n) = -2(0.4)^n u(-n-1) - (0.2)^n u(n)$$

⇒ Poles are 0.4 and 0.2

So system is Stable.

PROGRAM:

```
clc;
clear all;
close all;
%y(n)=x(n)+0.6y(n-1)-0.08y(n-2)
%taking z transform of above equation
zN = [0 0 1];
zD = [1 -0.6 0.08];

%% (a) Determine the impulse response for the system
%described by second order difference equation -
%y(n)=x(n)+0.6y(n-1)-0.08y(n-2)
h = impz(zN,zD);
disp(h)
figure;
impz(zN,zD);
grid on

%% (b)Also obtain the pole zero plot for the same and hence
%comment on the ROC and stability of the system.
[z,p,k] = tf2zp(z,zD)
disp(z)
disp(p)
disp(k)
figure;
zplane(z,p)
grid on

%% (c)Check the stability of the system analytically (hint:
%use the equation of h(n) and comment on stability.
if(p<1)
   disp('system is stable')
elseif(z==1
   if(p==1)
        disp('system is marginally stable')
   else
       disp('system is unstable')
   end
end
```

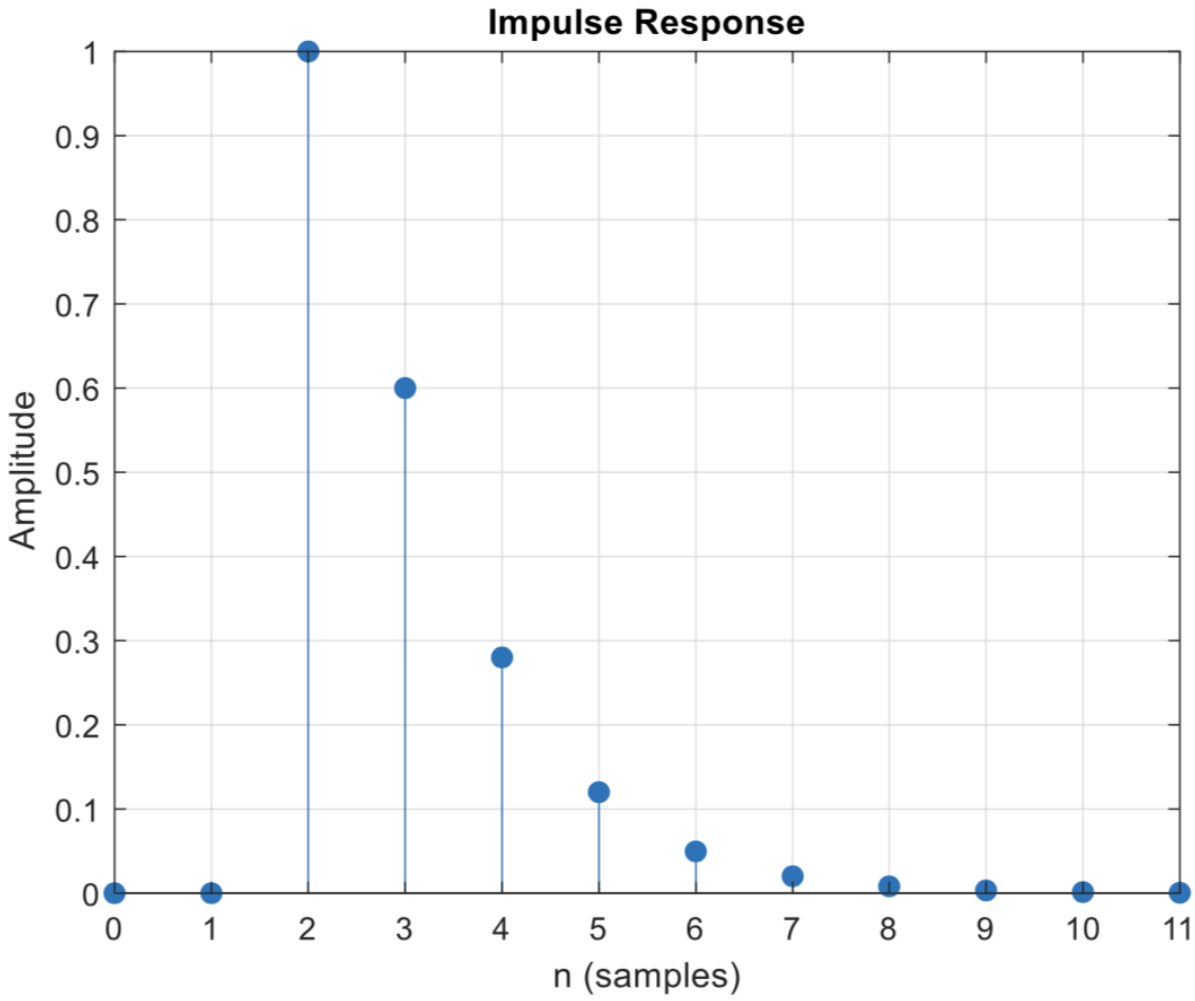
OUTPUT:

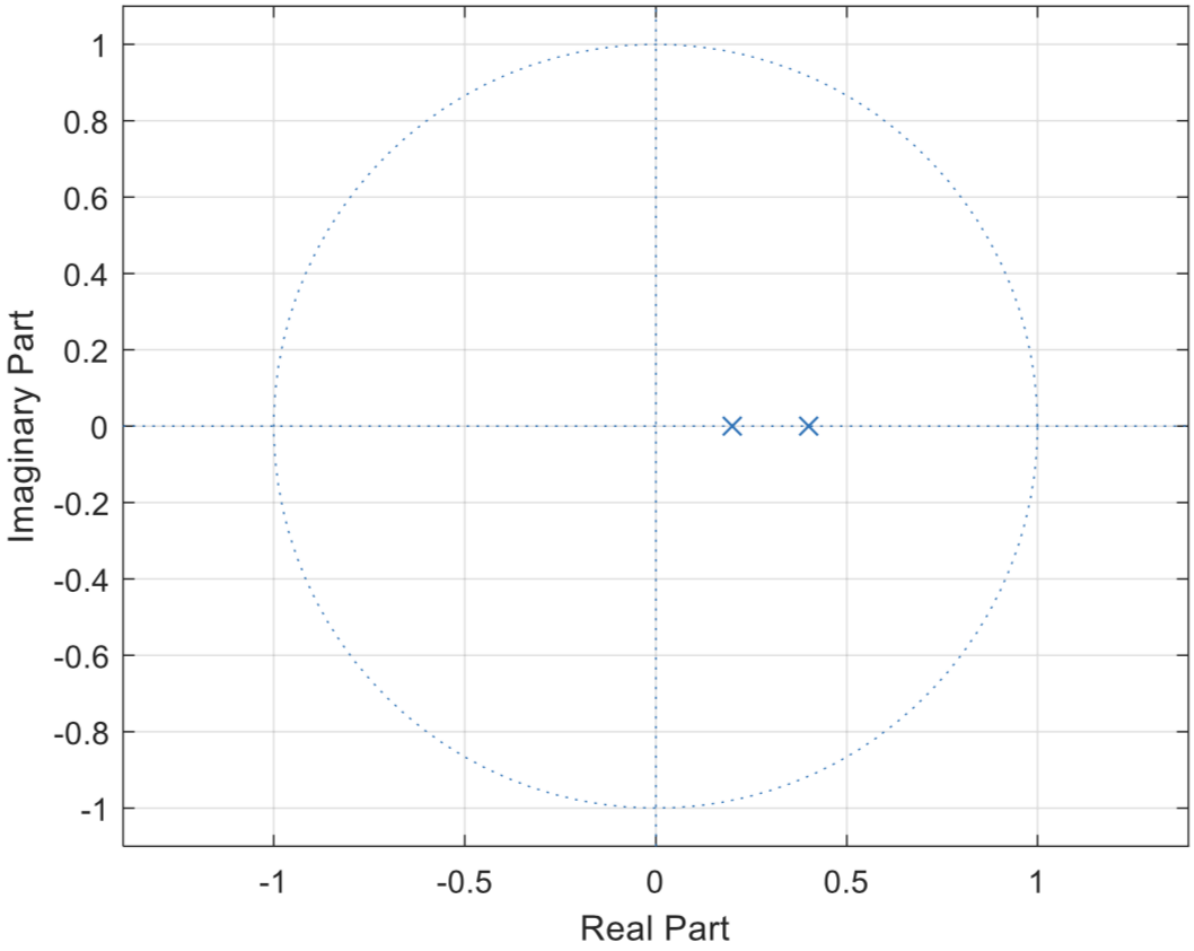
z=0

p = 0.4000
    0.2000

k= 1

system is stable




Imaginary Part
Real Part
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

Lab:05

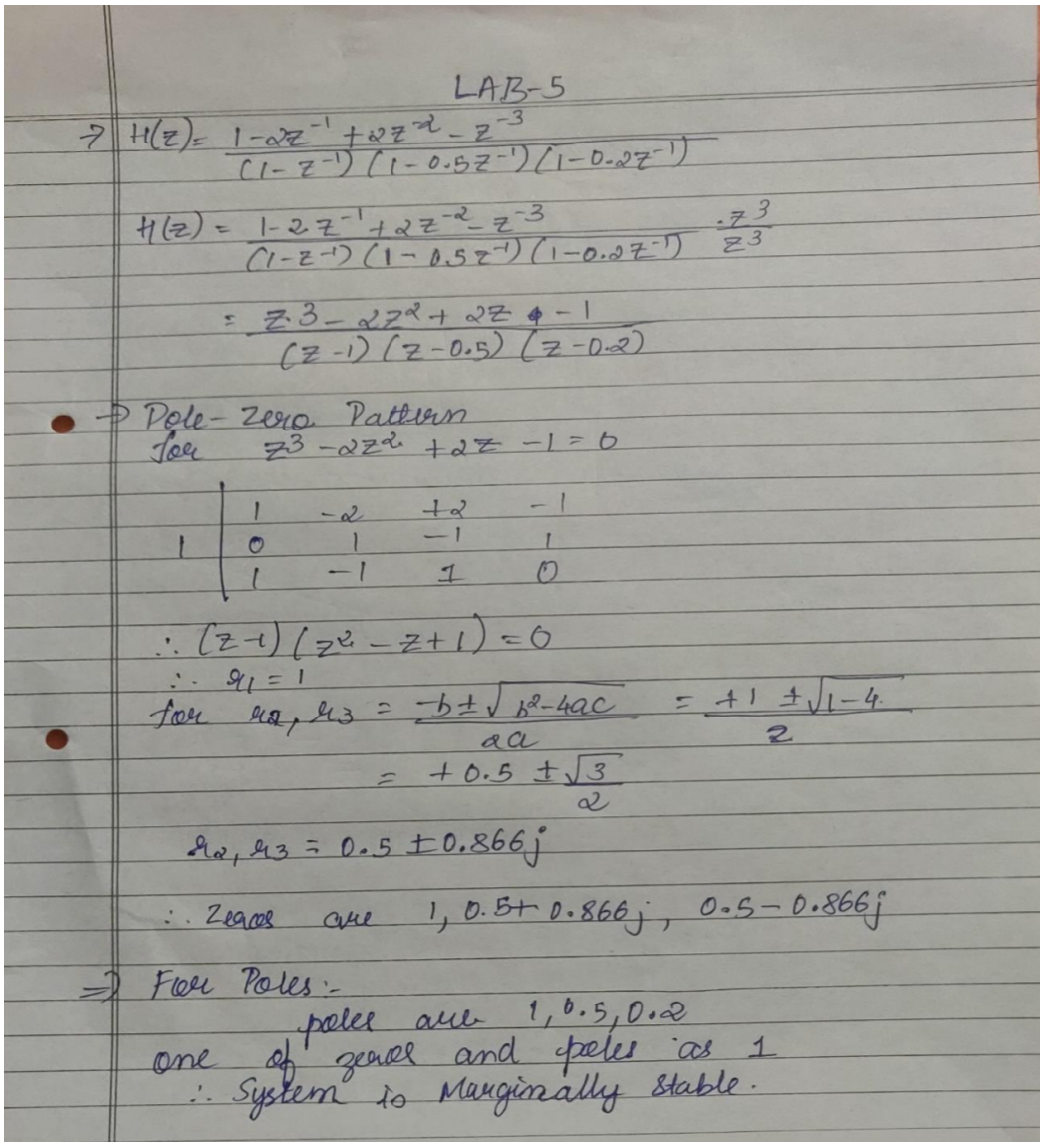
AIM: Consider the system H(z)= (1−2 z^−1+2z^−2−z^−3) /(1−z^−1)(1−0.5^ z−1)(1−0.2^ z−1): ROC : 0.5 < |z| < 1

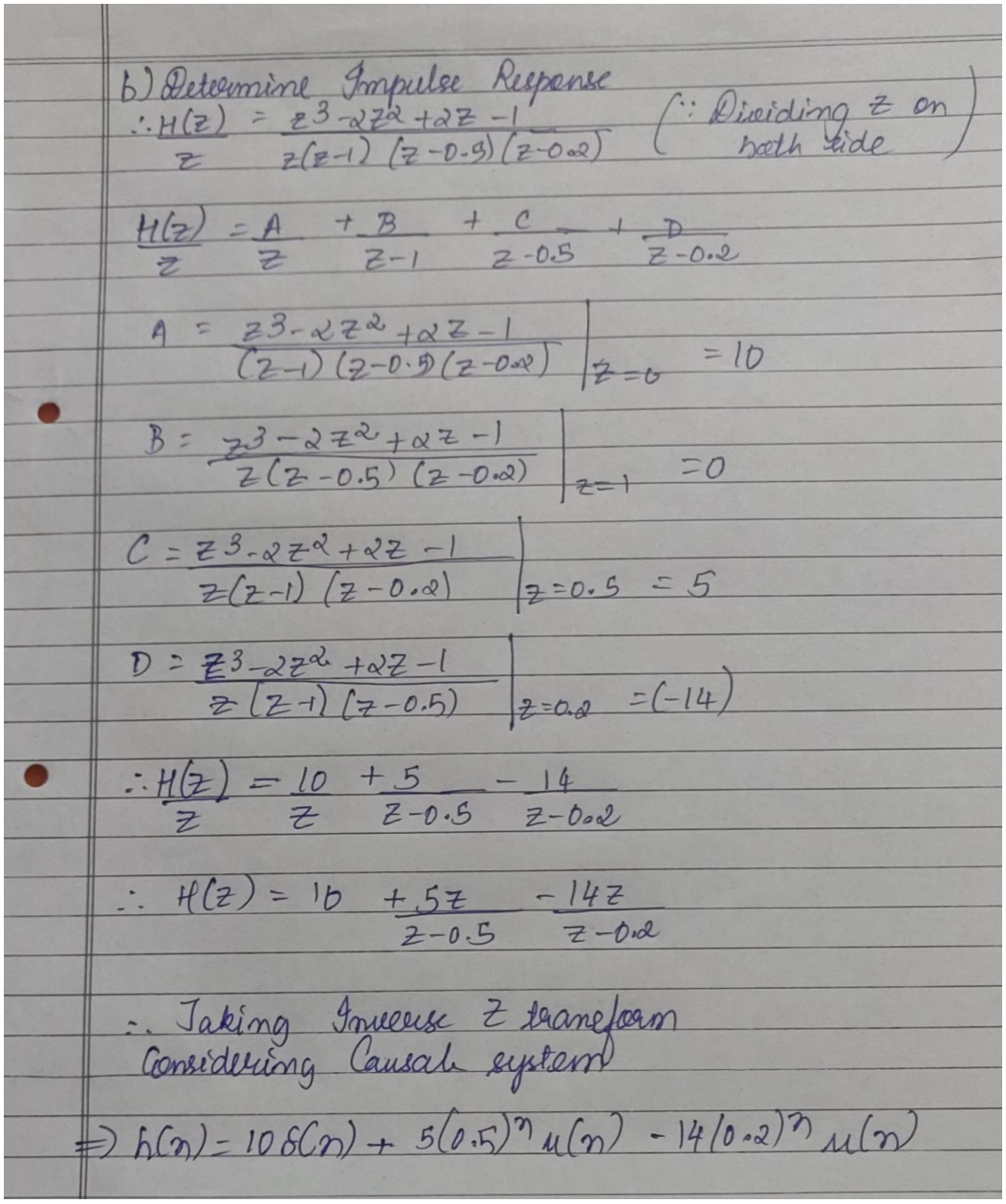
(a) Sketch Pole-Zero patterns. Find if the system is stable?

(b) Determine the impulse response of the system.

(c) Consider generalization of the system described H(z)= (1−a1* z^−1+ a2*z^−2− a3* z^−3) /(1−b1*z^−1)(1−b2* ^ z−1)(1−b3^ z−1):and think that it is all a distinct pole causal system. You should write the code that reads coefficients b1,b2,b3 of the system. and then find the stability of the system..

CALCULATION :





CODE :

```
clc;
clear all;
close all;

n = [1 -2 2 -1];
d = [1 -1.7 0.8 -0.1];
[z,p,a] = tf2zp(n,d)
figure;
```

```
zplane(z,p);
grid on;
ze = length(z);
poles = length(p);
h = impz(n,d);
figure;
stem(h);
grid on;
for k = 1:1:ze
   for i =1:1:poles
      if(z(k)<=1)
         if(p(i)<=1)
            disp('system is marginally stable')
         else
            disp('system is stable')
         end
      else
         disp('system is unstable')
      end
   end
End
```

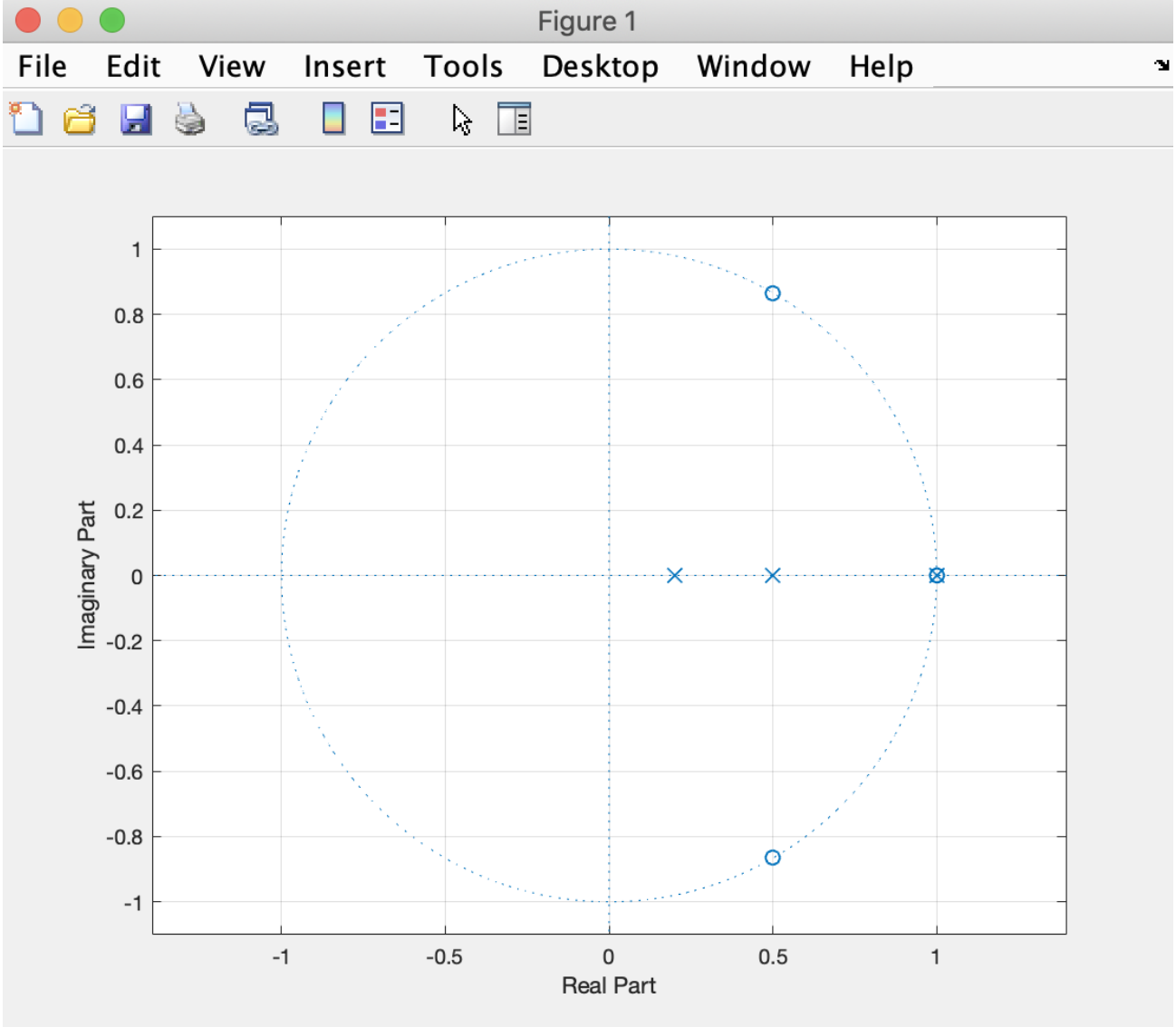
OUTPUT:

z = 1.0000 + 0.0000i 0.5000 + 0.8660i 0.5000 - 0.8660i

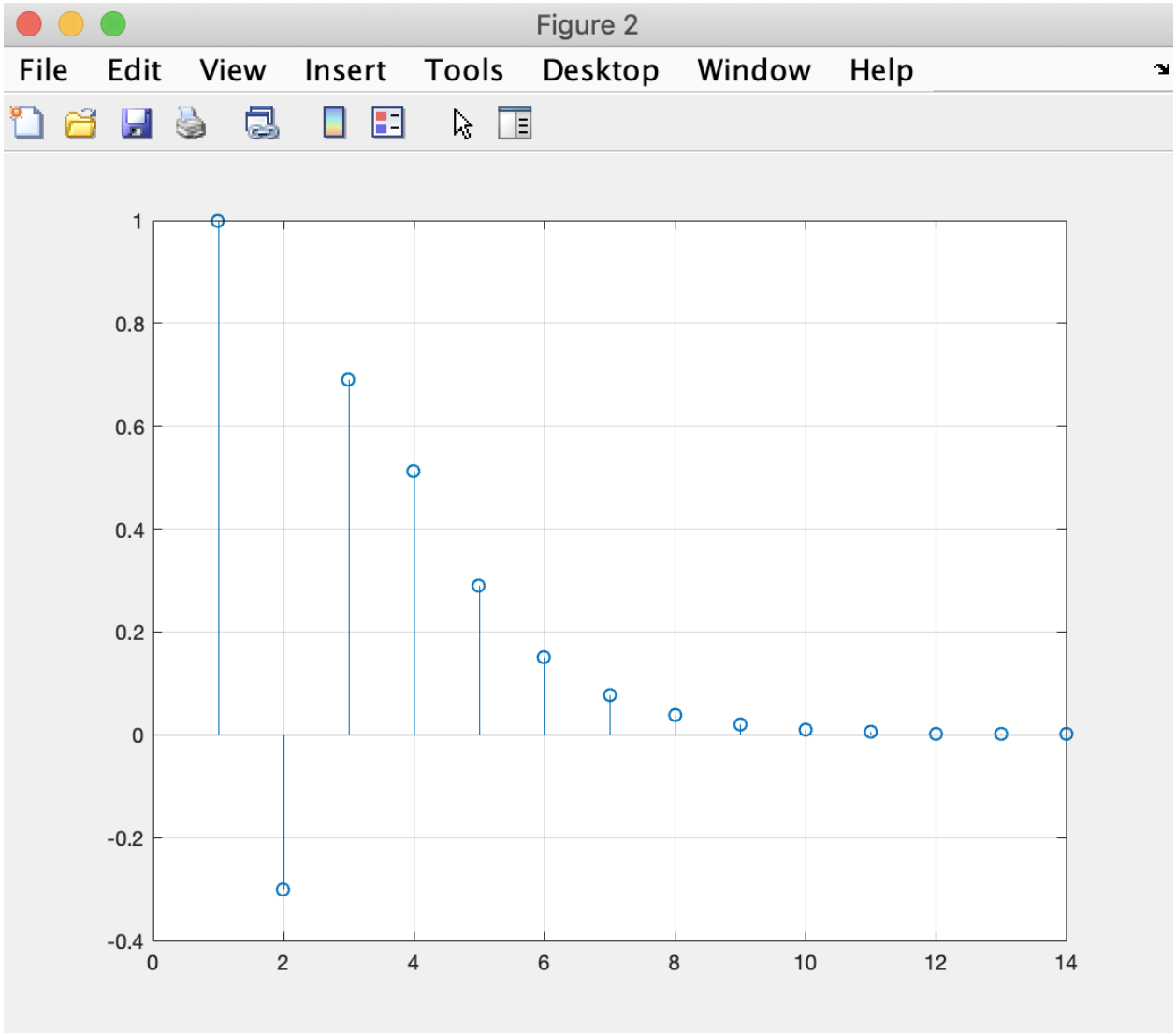
p = 1.0000 0.5000 0.2000

system is marginally stable

(A) Sketch Pole-Zero patterns



(B) Impulse Response:



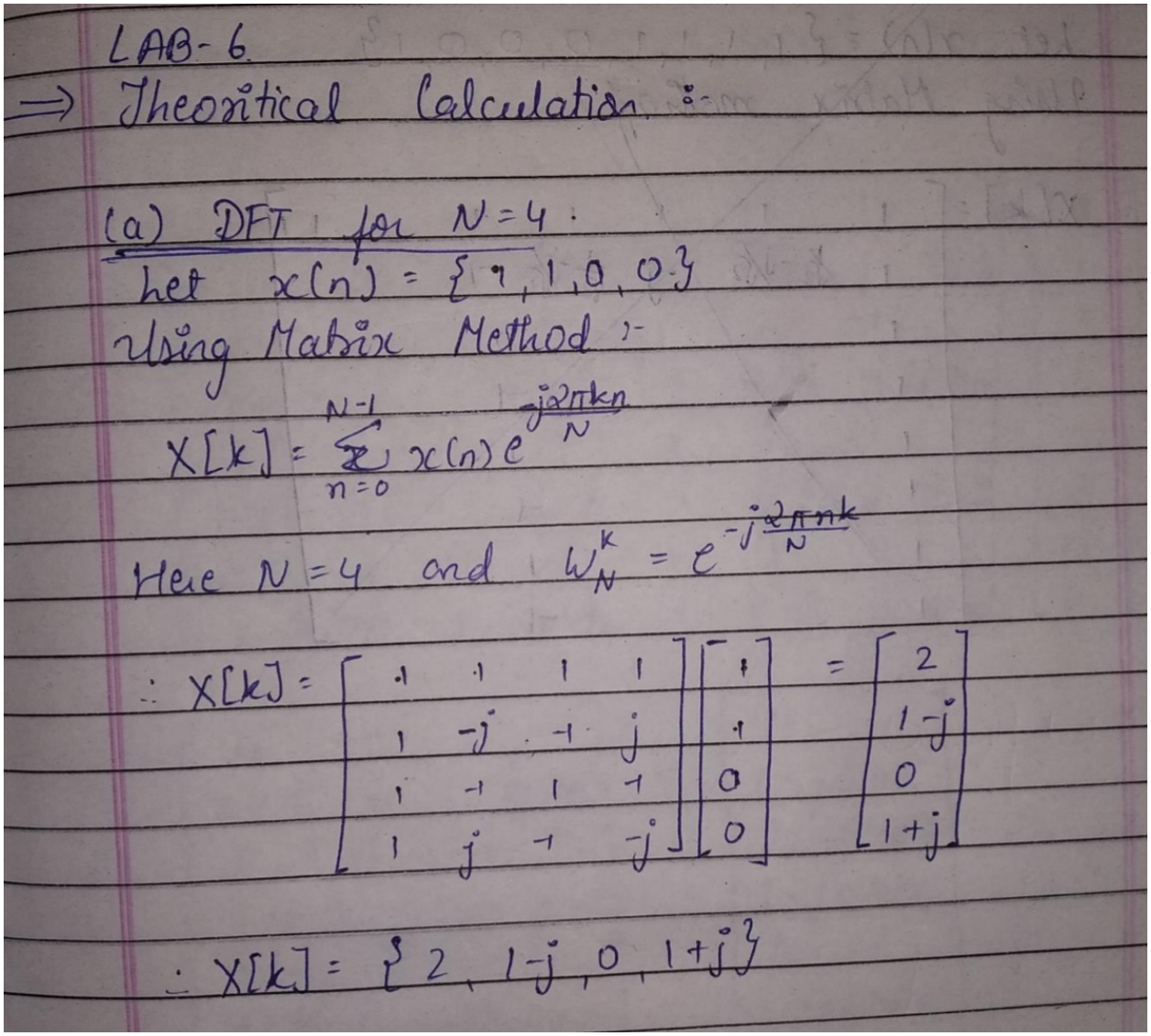
CONCLUSION:

Thus, in this experiment we have plotted the poles and zeros of the given Z transform function and also its impulse response. We verified our output theoretically.

PRACTICAL – 6

AIM:

(a)Perform DFT for N= 4 using fft function and compare the result



LAB-6

⇒ Theoritical Calculation :-

(a) DFT for N=4 :

let $x(n) = \{1, 1, 0, 0\}$

Using Matrix Method :-

$$X[k] = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}$$

Here $N = 4$ and $W_N^k = e^{-j\frac{2\pi nk}{N}}$

$$\therefore X[k] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1-j \\ 0 \\ 1+j \end{bmatrix}$$

$$\therefore X[k] = \{2, 1-j, 0, 1+j\}$$

PROGRAM:

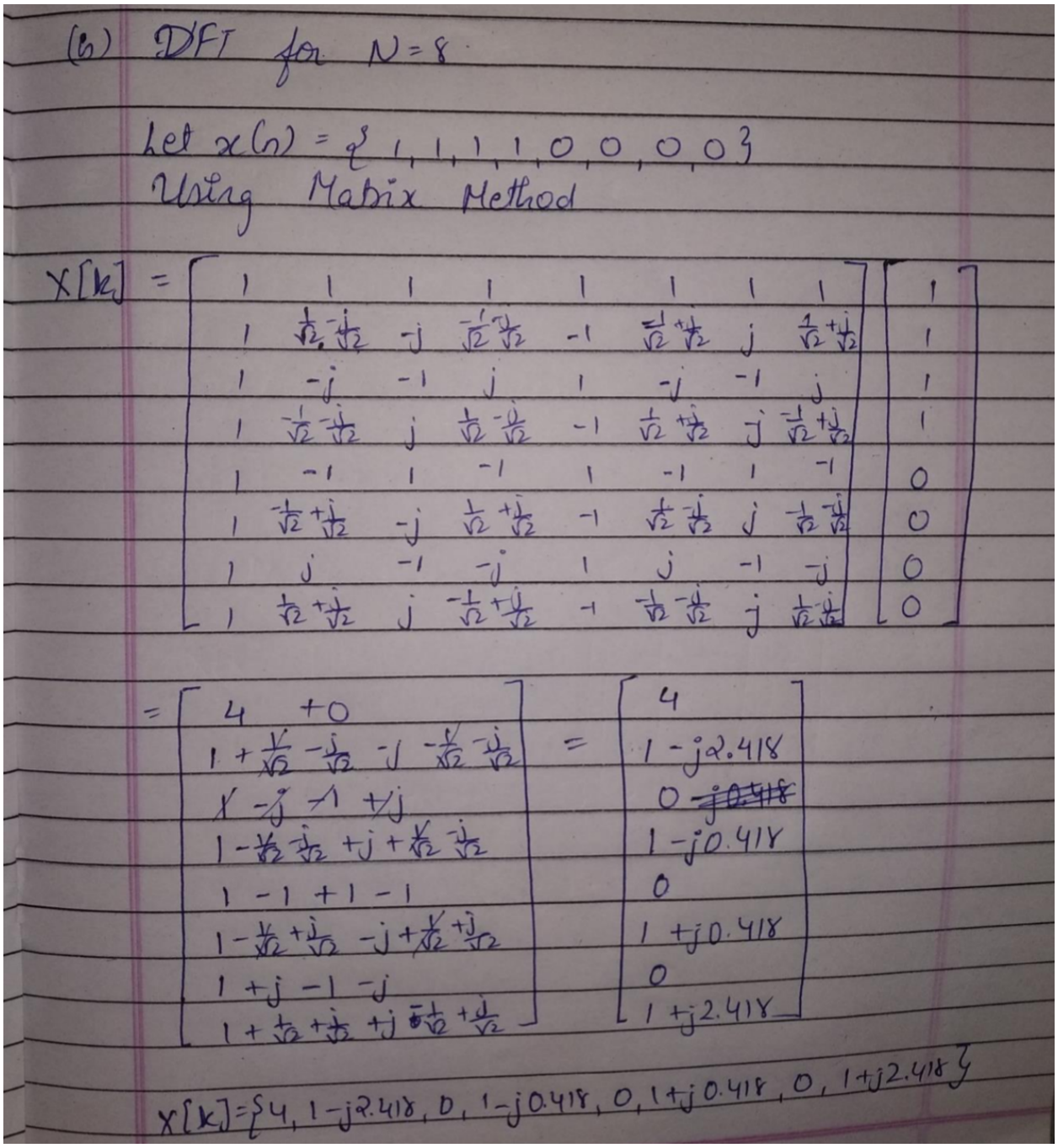
```
clc;
clear all;
close all;
%%
%a)Perform DFT for N= 4 using fft function and without function and
compare the result
xn = input('Enter x(n): ');
N1 = length(xn);
if (N1<4)
   a = zeros(1,4-N1);
   xn1 = cat(2,xn,a);
   disp(xn1)
else
        xn1 = xn;
end

% using function fft
Xk_fft = fft(xn1,4)

% without using fft function
N = length(xn1);
for n = 0:1:N-1
        for k =0:1:N-1
        p = exp((-i*2*pi*k*n)/N);
        xn2(k+1,n+1) = p;
        end
end
Xk = xn1*xn2
```

OUTPUT:

```
Enter x(n): [1 1 0 0]

Xk_fft = 2.0000 + 0.0000i 1.0000 - 1.0000i 0.0000 + 0.0000i

Xk = 2.0000 + 0.0000i 1.0000 - 1.0000i 0.0000 - 0.0000i 1.0000 + 1.0000i
```

b) Perform DFT for N=8 using fft function and without function and compare the result

THEORETICAL CALCULATION:



PROGRAM:

```
clc;
clear all;
close all;
%%
%b)Perform DFT for N=8 using fft function and without function and
compare the result
xn8 = input('Enter x(n): ');
N18 = length(xn8);
if (N18<4)
   a = zeros(1,8-N18);
   xn18 = cat(2,xn,a);
   disp(xn18)
else
    xn18 = xn8;
end

% using function fft
Xk_fft_8 = fft(xn18,8)

% without using fft function
N8 = length(xn18);
for n = 0:1:N8-1
  for k =0:1:N8-1
      p = exp((-j*2*pi*k*n)/N8);
      xn28(k+1,n+1) = p;
  end
end
Xk_8 = xn18*xn28
```

OUTPUT:

Enter x(n): [1 1 1 1 0 0 0 0]

Xk_fft_8 = 4.0000 + 0.0000i , 1.0000 - 2.4142i , 0.0000 + 0.0000i , 1.0000 - 0.4142i, 0.0000 + 0.0000i , 1.0000 + 0.4142i , 0.0000 + 0.0000i , 1.0000 + 2.4142i

Xk_8 = 4.0000 + 0.0000i , 1.0000 - 2.4142i , -0.0000 - 0.0000i , 1.0000 - 0.4142i , 0.0000 - 0.0000i , 1.0000 + 0.4142i , 0.0000 - 0.0000i , 1.0000 + 2.4142i

c) Consider Frequency analysis of the Amplitude Modulated Discrete-time Signal x(n) = Cos 2πf1n + Cos 2πf2n Where f1 = and f2 = modulates the Amplitude modulated signal: f1=1/128 .f2=5/128 Xc(n)= Cos 2πfcn where fc = The resulting AM signal is Fc=50/128 xam(n) = x(n)Cos 2πfcn. Using MATLABProgram: Sketch the signals x(n), xc(n) and xam(n); 0 < n < 255 Compute and sketch the 128-point DFT of the signal xam(n); 0 < n < 127 Compute and sketch the 128-point DFT of the signal xam(n):

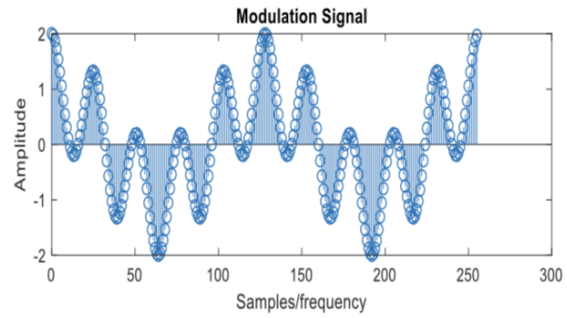
PROGRAM:

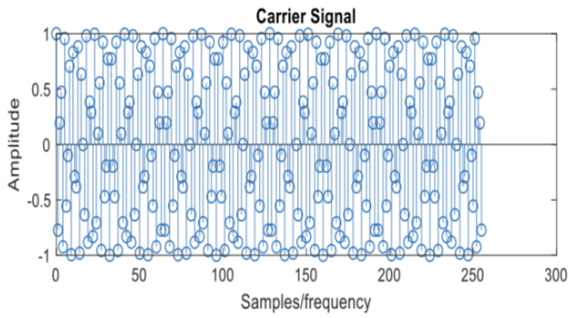
```
clc;
clear ll;
close all;

%c)Consider Frequency analysis of the Amplitude Modulated Discrete-time
Signal (DSB - SC)
%x(n) = Cos 2?f1n + Cos 2?f2n Where f1 = and f2 = modulates the Amplitude
modulated signal:
%f1=1/128 .f2=5/128 Xc(n) = Cos 2?fcn where fc = The resulting AM signal
is
%Fc=50/128 xam(n) = x(n)Cos 2?fcn. Using MATLAB Program:
%(A)Sketch the signals x(n),xc(n) and xam(n); 0 < n < 255
%(B)Compute and sketch the 128-point DFT of the signal xam(n); 0 < n <
127
%(C)Compute and sketch the 100-point DFT of the signal xam(n): 0 < n < 99
%(D)Compute and sketch the 128-point DFT of the signal xam(n): 0 < n < 99

%% PART A

f1=1/128;
f2=5/128;
fc=50/128;

n=0:1:255;
xn = cos(2*pi*f1*n) + cos(2*pi*f2*n);
subplot(321)
stem(n,xn)
```

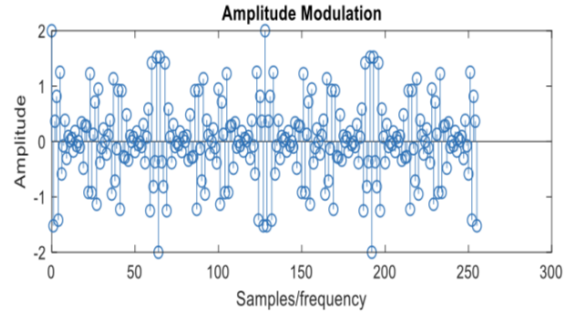
```
title('Modulation Signal')
xlabel('Samples/frequency')
ylabel('Amplitude')

xc = cos(2*pi*fc*n);
subplot(322)
stem(n,xc)
title('Carrier Signal')
xlabel('Samples/frequency')
ylabel('Amplitude')

x_dsbsc = xn.*xc;
subplot(323)
stem(n,x_dsbsc)
title('Amplitude Modulation')
xlabel('Samples/frequency')
ylabel('Amplitude')

%%
%PART B
n1 = 0:1:127;
xn1 = cos(2*pi*f1*n1)+cos(2*pi*f2*n1);
xc1 = cos(2*pi*fc*n1);
z_dsbsc1 = xn1.*xc1;
subplot(324)
y1=fft(z_dsbsc1,128);
stem(y1)
title('Amplitude Modulation for Xam=0<n<127')
xlabel('Samples/frequency')
ylabel('Amplitude')

%%
% PART C
n2 = 0:1:99;
xn2 = cos(2*pi*f1*n2)+cos(2*pi*f2*n2);
xc2 = cos(2*pi*fc*n2);
x_dsbsc2 = xn2.*xc2;
```

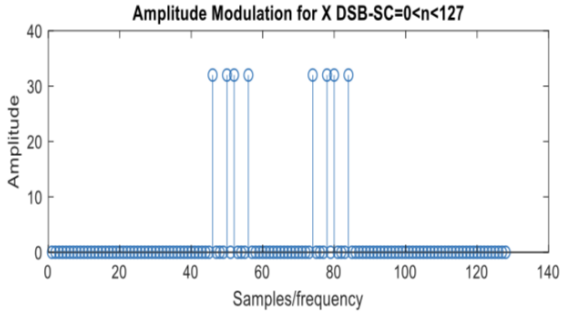
```
subplot(325)
y2=fft(x_dsbsc2,100);
stem(y2)
title('Amplitude Modulation for Xam=0<n<99')
xlabel('frequency')
ylabel('Amplitude')

%%
% PART D
n3 = 0:1:99;
xn3 = cos(2*pi*f1*n3)+cos(2*pi*f2*n3);
xc3 = cos(2*pi*fc*n3);
x_dsbsc3 = xn3.*xc3;
subplot(326)
y3=fft(x_dsbsc3,127);
stem(y3)
title('Amplitude Modulation for Xam=0<n<127')
xlabel('frequency')
ylabel('Amplitude')
```

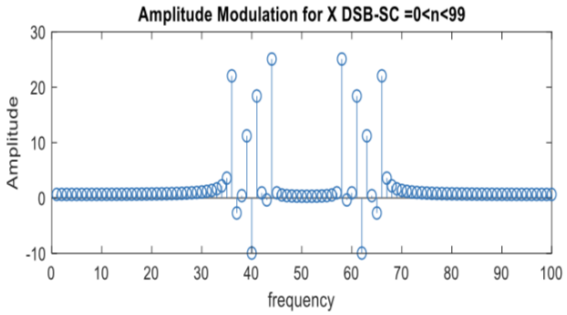
OUTPUT:

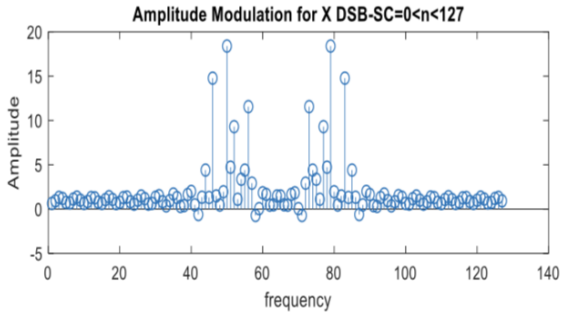












CONCLUSION:

When 0<n<127 and length of DFT = 128 i.e. length of DFT and x(n) is same then there are no negative amplitudes and all maximum amplitudes are equal and all minimum amplitudes are 0.

When length of DFT is less than length of x(n) then DSB signal is expanded When length of DFT is more than length of x(n) then DSB signal is compressed

Lab7 : Butterworth low pass and high pass filter design

AIM : (a) Design Butterworth LPF for following specifications:
% Pass band Attenuation: 0.4dB
% Stop band Attenuation: 35 dB
% Pass band frequency: your Roll no*5 + 400Hz
% Stop band frequency: your Roll no*5 + 800 Hz
% • Sampling frequency: 3000 Hz
% • Write the Matlab program to accomplish the same.

(b) What changes are required to design a HPF for the same specifications? Hence design and program the same

(c) Compute the values of N and Wc theoretically and compare your results with MATLAB program output.

CALCULATION:

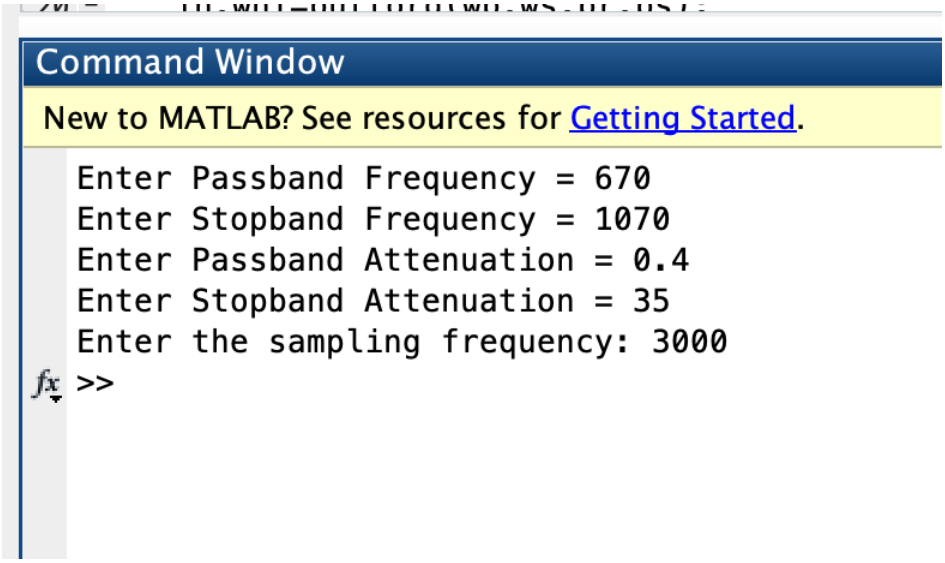
Pass band frequency: your Roll no*5 + 400Hz = 54*5 + 400 = 270 + 400 = 670
Stop band frequency: your Roll no*5 + 800 Hz = 54*5 + 800 = 270 + 800 = 1070

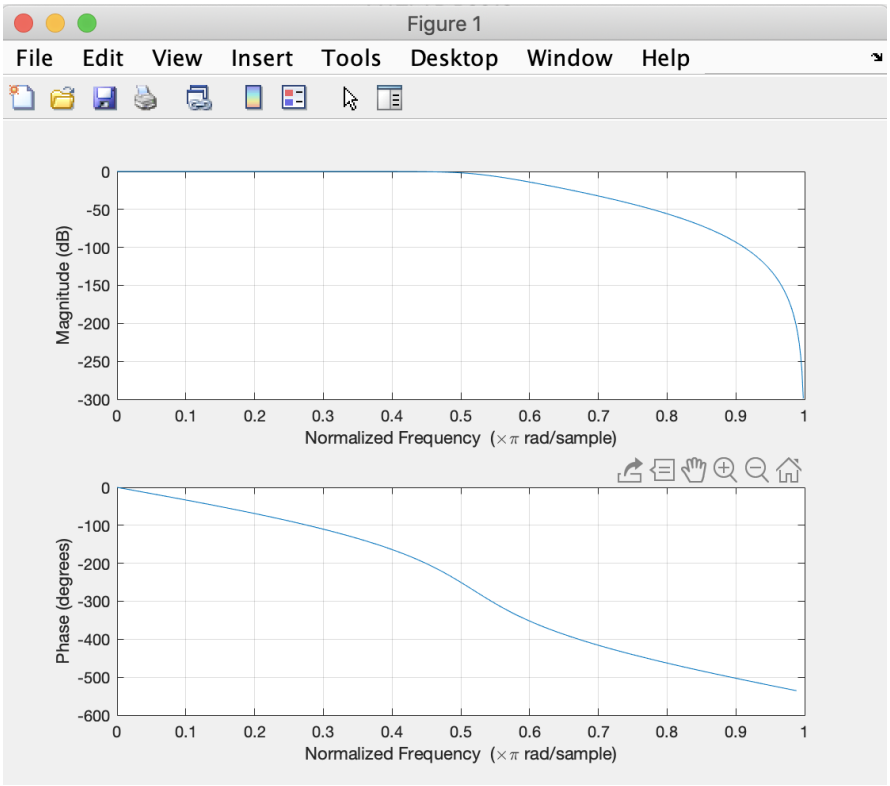
CODE :

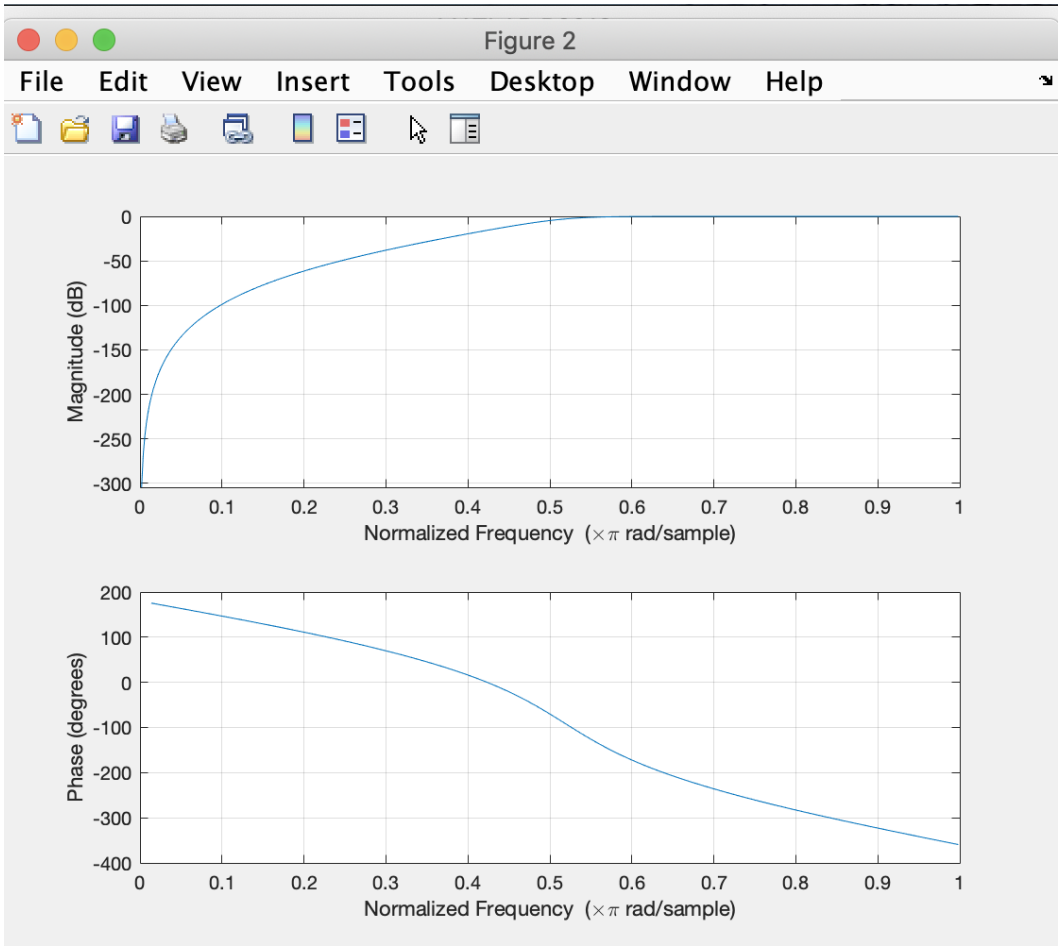
```
clc
close all
pf = input('Enter Passband Frequency = ');
sf = input('Enter Stopband Frequency = ');
pr = input('Enter Passband Attenuation = ');
ps = input('Enter Stopband Attenuation = ');
f_s=input('Enter the sampling frequency: ');

%% LPF
div = f_s/2;
wp = pf/div;
ws = sf/div;
[n,wn] = buttord(wp,ws,pr,ps); [b,a] = butter(n,wn); figure(1)
freqz(b,a)

%% hpf %%
div = f_s/2;
wp=pf/div;
```

```
ws=sf/div;
[n,wn]=buttord(wp,ws,pr,ps);
[b,a]=butter(n,wn,'high');
figure(2)
freqz(b,a)
```

OUTPUT :







CONCLUSION :

Thus, from this experiment we observed that we can use fdatool or fvtool inbuilt in MATLAB to design the filter of our specification without any code and observe the graphs, group delay and phase delay.

(c) Compute the values of N and Wc theoretically and compare your results with MATLAB program output.

(c) Theoretical Analysis.

Passband Attenuation : $A_p = 0.4$ dB
Stopband Attenuation : $A_s = 35$ dB
Passband freq : $\Omega_p = 670$ Hz
Stopband freq : $\Omega_s = 1070$ Hz

$$\therefore N \geq \frac{\log \sqrt{\frac{10^{0.1A_s} - 1}{10^{0.1A_p} - 1}}}{\log\left(\frac{\Omega_s}{\Omega_p}\right)}$$

$$N \geq \frac{\log \sqrt{\frac{10^{(0.1)(35)} - 1}{10^{(0.1)(0.4)} - 1}}}{\log\left(\frac{1070}{670}\right)}$$

$$N \geq \frac{\log \left(\frac{56.22}{0.3106}\right)}{\log (1.597)}$$

$$N \geq \frac{2.25}{0.201}$$

$$N \geq 11.19$$

$$N \approx 11$$

Now,

$$\Omega_c = \frac{\Omega_p}{(10^{0.1A_f} - 1)^{1/2N}}$$

$$= \frac{670}{((10)^{0.1(0.4)} - 1)^{1/2(11)}}$$

$$\Omega_c = \frac{670}{(0.096)^{1/22}} = \frac{670}{(0.096)^{0.045}}$$

$$= \frac{670}{0.899} = \frac{670}{0.9}$$

$$= 744.4 \text{ Hz}$$

PRACTICAL – 8

IIR FILTER PRACTICE SESSION

AIM :

Design four types of filter

1. Butterworth,

2. Chebyshev type 1

3. Chebyshev type 2

4. Elliptical

They are designed for four type of frequency response

High pass , Low pass, Band pass and Band stop

Write the program for High pass , and Band stop

Give necessary observations

1. Response in terms ripple

2. Group delay effect

3. Comparison with cheb type1 and Chenby type 2 band pass filter response

4. Comparison of all high pass response , Bandstop response in terms of ripple existance

5. Overall, which type is preferable ?Why?

LOW PASS FILTER

CODE :

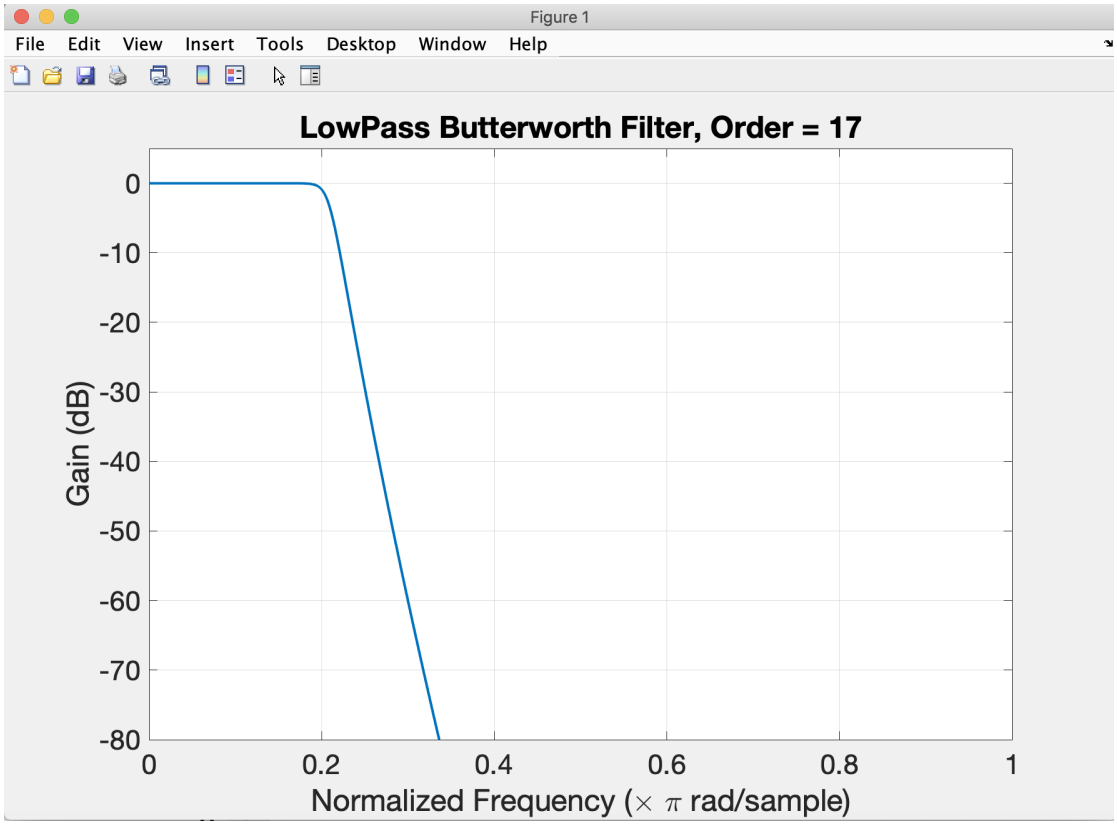
```
clc;
clear all ;
close all;

set(0,'defaultaxesfontsize',22);
set(0,'defaulttextfontsize',22);
linwidth = 2;

% Case 1: Lowpass filter
% Specifications:
% Passband 0 to 0.2pi rads, gain between 1 and 0.9
% Stopband 0.3pi to pi rads, atten 0.001 = -60 dB
% Compare Butterworth, Chebyshev I & II, and elliptic designs.

passband = 0.2*pi/pi; % convert to normalized frequency
stopband = 0.3*pi/pi;
passrip = -20*log10(0.9) % ripple in positive dB
stopatten = -20*log10(0.001) %stopband attenuation in positive dB

% Find order and natural frequency to meet these specs

[Nb, Wnb] = buttord(passband, stopband, passrip, stopatten); % Butterworth filter

[Nc1, Wnc1] = cheb1ord(passband, stopband, passrip, stopatten); % Cheby 1 filter

[Nc2, Wnc2] = cheb2ord(passband, stopband, passrip, stopatten);% Cheby 2 filter

[Ne, Wne] = ellipord(passband, stopband, passrip, stopatten); % Elliptic filter

% Now use the order and frequencies just identified to design these filters
% by finding their difference equation coefficients

[Bb,Ab] = butter(Nb, Wnb);
[Bc1,Ac1] = cheby1(Nc1,passrip,Wnc1);
[Bc2,Ac2] = cheby2(Nc2,stopatten,Wnc2);
[Be,Ae] = ellip(Ne,passrip,stopatten,Wne);
```

```
% Display frequency responses figure(1)

[Hb,W]= freqz(Bb,Ab,2048);
h = plot(W/pi,20*log10(abs(Hb)));
set(h,'LineWidth',linwidth);
title(['LowPass Butterworth Filter, Order = ',num2str(Nb)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(2)
[Hc1,W]= freqz(Bc1,Ac1,2048);

h = plot(W/pi,20*log10(abs(Hc1)));
set(h,'LineWidth',linwidth);
title(['LowPass Chebyshev Type 1 Filter, Order = ',num2str(Nc1)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(3)
[Hc2,W]= freqz(Bc2,Ac2,2048);
h = plot(W/pi,20*log10(abs(Hc2)));
set(h,'LineWidth',linwidth);
title(['LowPass Chebyshev Type 2 Filter, Order = ',num2str(Nc2)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(4)
[He,W]= freqz(Be,Ae,2048);
```

```
h = plot(W/pi,20*log10(abs(He)));
set(h,'LineWidth',linwidth);
title(['LowPass Elliptic Filter, Order = ',num2str(Ne)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

% Compute group delay for these filters
[Gdb,W] = grpdelay(Bb,Ab,1024);

% Gdb is group delay, W is freqs for Gdb

[Gdc1,W] = grpdelay(Bc1,Ac1,1024);
[Gdc2,W] = grpdelay(Bc2,Ac2,1024);
[Gde,W] = grpdelay(Be,Ae,1024);

% Display limited range (passband vicinity) of the group delay in normalized freq

Wnm = W/pi; % divide by pi to normalize W to 0 to 1,

% select first 150 frequencies to capture passband of filter (0 to 0.1pi)
figure(5)
h = plot(Wnm,Gdb,'k',Wnm,Gdc1,'g',Wnm,Gdc2,'r',Wnm,Gde,'b');
xlim([0 .3])
set(h,'LineWidth',linwidth);
legend('Butterworth','Chebyshev 1', 'Chebyshev 2', 'Elliptic')
title('Group Delay of LowPass')
xlabel('Normalized Frequency (\times \pi rad/sample)')
ylabel('Samples of delay')
grid on
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])
```

OUTPUT :

```
Command Window
New to MATLAB? See resources fo

passrip =

    0.9151


stopatten =

    60

fx >>
```



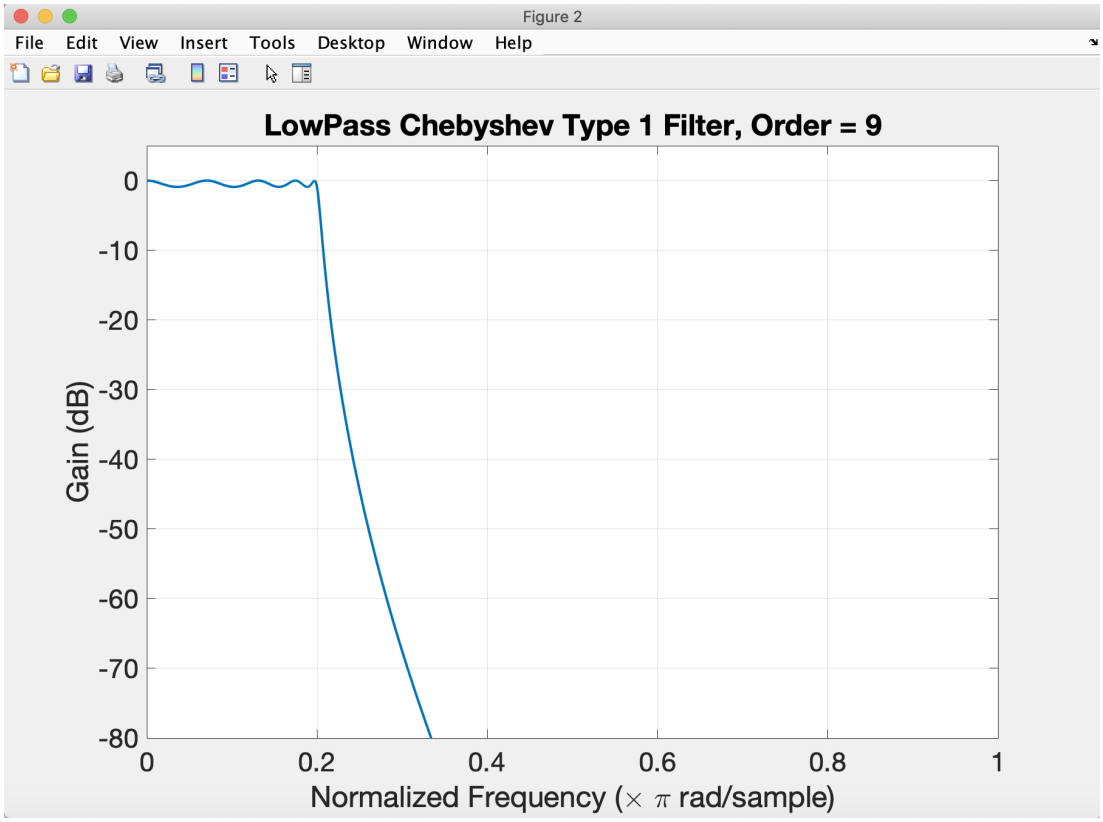


Figure 2
File Edit View Insert Tools Desktop Window Help
LowPass Chebyshev Type 1 Filter, Order = 9
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

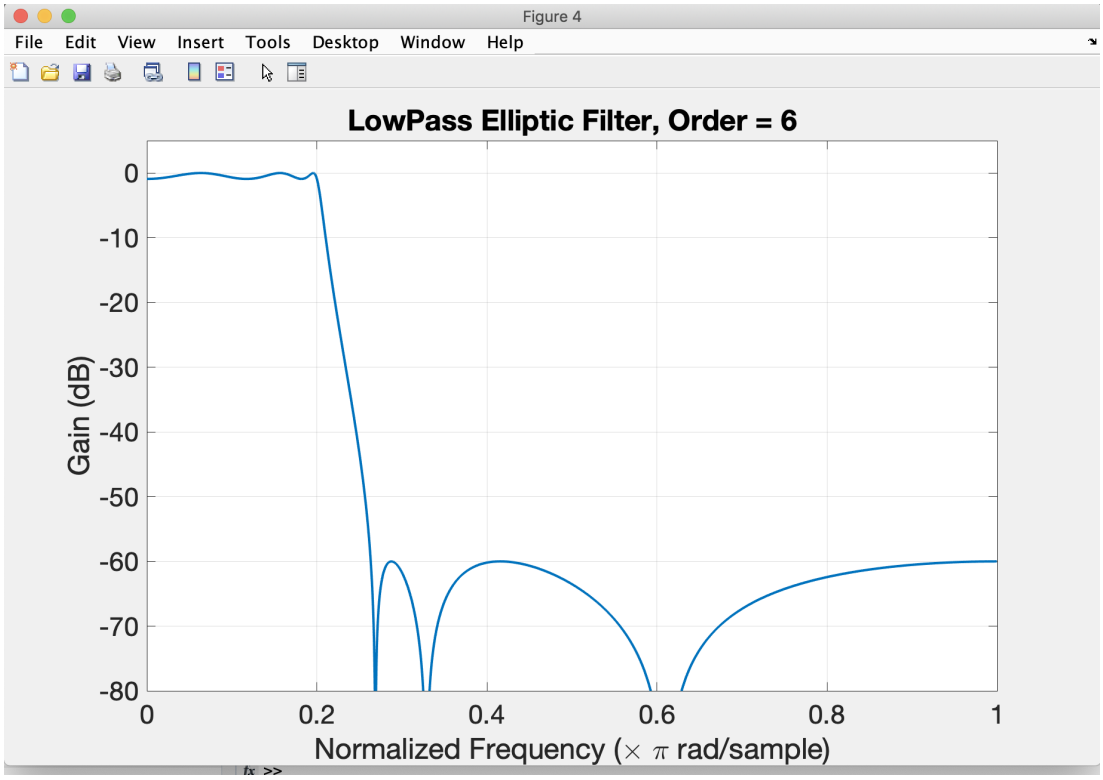
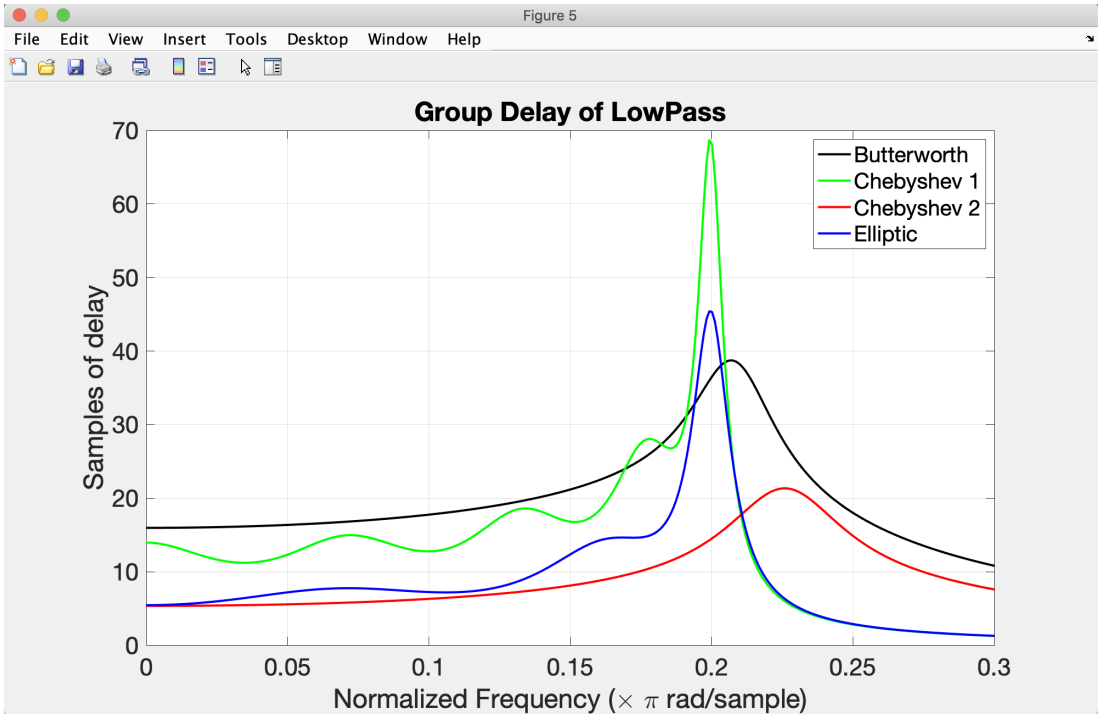


Figure 4
File Edit View Insert Tools Desktop Window Help
LowPass Elliptic Filter, Order = 6
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)



HIGH PASS FILTER

CODE :

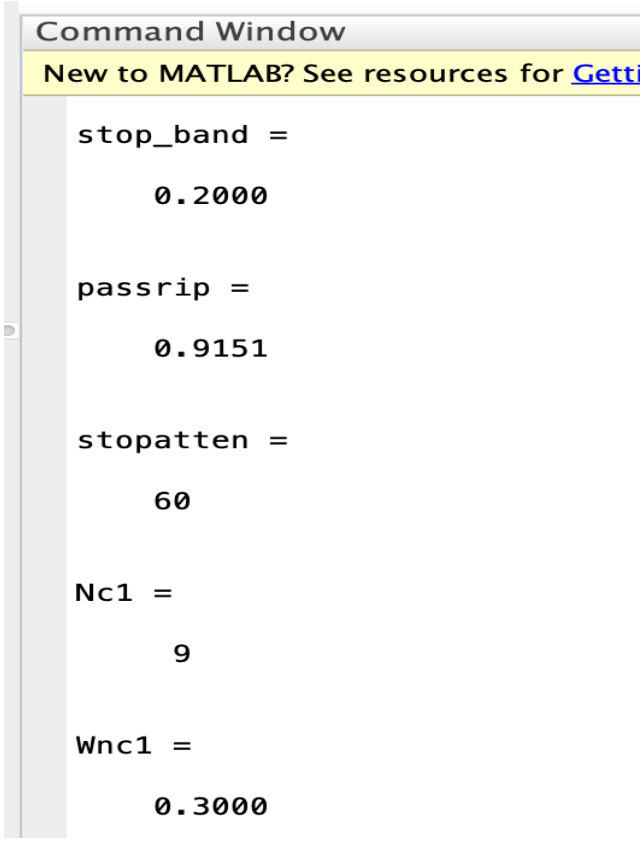
```
clc;
clear all;
close all;

set(0,'defaultaxesfontsize',22);
set(0,'defaulttextfontsize',22);
linwidth = 2;

% Case 3 High Pass Filter
%Butterworth
pass_band = 0.3*pi/pi
stop_band = 0.2*pi/pi

passrip = -20*log10(0.9) % ripple in positive dB
stopatten = -20*log10(0.001) %stopband attenuation in positive dB

% Find order and natural frequency to meet these specs
```

```
[Nb, Wnb] = buttord(pass_band, stop_band, passrip, stopatten);
% Butterworth filter

[Nc1, Wnc1] = cheb1ord(pass_band, stop_band, passrip, stopatten) % Cheby 1 filter
[Nc2, Wnc2] = cheb2ord(pass_band, stop_band, passrip, stopatten);% Cheby 2 filter
[Ne, Wne] = ellipord(pass_band, stop_band, passrip, stopatten); % Elliptic filter

% Now use the order and frequencies just identified to design these filters
% by finding their difference equation coefficients

[Bb,Ab] = butter(Nb, Wnb,'high');
[Bc1,Ac1] = cheby1(Nc1,passrip,Wnc1,'high');
[Bc2,Ac2] = cheby2(Nc2,stopatten,Wnc2,'high');
[Be,Ae] = ellip(Ne,passrip,stopatten,Wne,'high');

% Display frequency responses
figure(1)
[Hb,W]= freqz(Bb,Ab,2048);
h = plot(W/pi,20*log10(abs(Hb)));

set(h,'LineWidth',linwidth);
title(['Butterworth Filter, Order = ',num2str(Nb)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(2)
[Hc1,W]= freqz(Bc1,Ac1,2048);
h = plot(W/pi,20*log10(abs(Hc1)));
set(h,'LineWidth',linwidth);
title(['Chebyshev Type 1 Filter, Order = ',num2str(Nc1)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])
```

```
figure(3)
[Hc2,W]= freqz(Bc2,Ac2,2048);
h = plot(W/pi,20*log10(abs(Hc2)));
set(h,'LineWidth',linwidth);
title(['Chebyshev Type 2 Filter, Order = ',num2str(Nc2)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(4)
[He,W]= freqz(Be,Ae,2048);
h = plot(W/pi,20*log10(abs(He)));
set(h,'LineWidth',linwidth);
title(['Elliptic Filter, Order = ',num2str(Ne)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

% Compute group delay for these filters
[Gdb,W] = grpdelay(Bb,Ab,1024);
% Gdb is group delay, W is freqs for Gdb
[Gdc1,W] = grpdelay(Bc1,Ac1,1024);
[Gdc2,W] = grpdelay(Bc2,Ac2,1024);
[Gde,W] = grpdelay(Be,Ae,1024);

% Display limited range (passband vicinity) of the group delay in normalized freq

Wnm = W/pi; % divide by pi to normalize W to 0 to 1,
% select first 150 frequencies to capture passband of filter (0 to 0.1pi)

figure(5)
h = plot(Wnm,Gdb,'k',Wnm,Gdc1,'g',Wnm,Gdc2,'r',Wnm,Gde,'b');
xlim([0 .3])
```

```
set(h,'LineWidth',linwidth);
legend('Butterworth','Chebyshev 1', 'Chebyshev 2', 'Elliptic')
title('Group Delay')
xlabel('Normalized Frequency (\times \pi rad/sample)')
ylabel('Samples of delay')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])
```

OUTPUT :



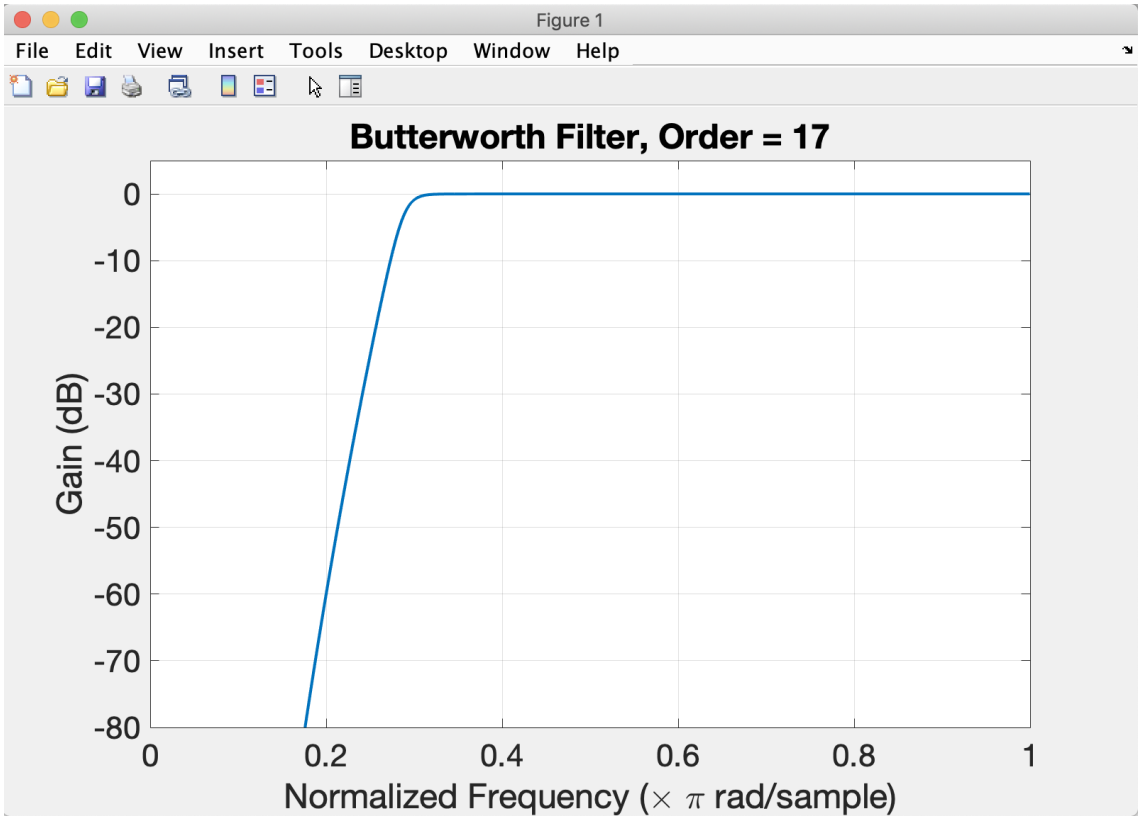


Figure 1
File Edit View Insert Tools Desktop Window Help
Butterworth Filter, Order = 17
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

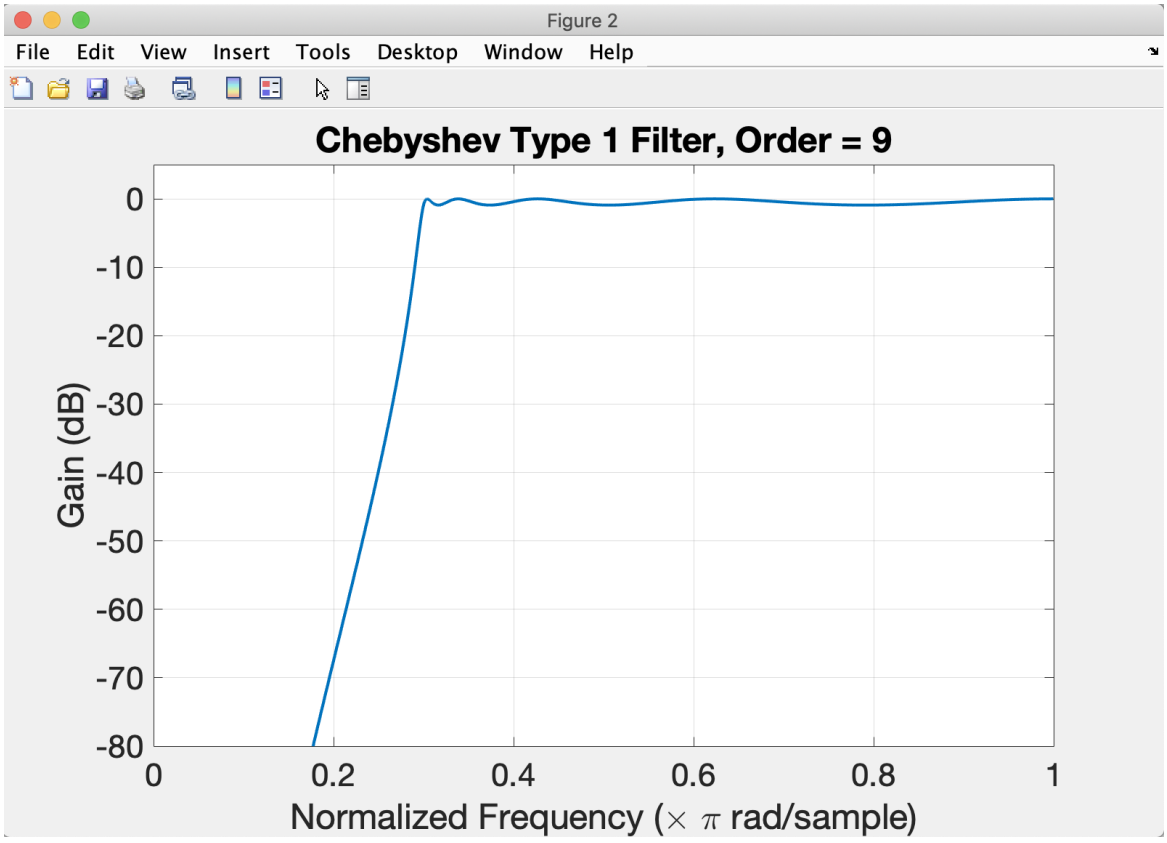


Figure 2
File Edit View Insert Tools Desktop Window Help
Chebyshev Type 1 Filter, Order = 9
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

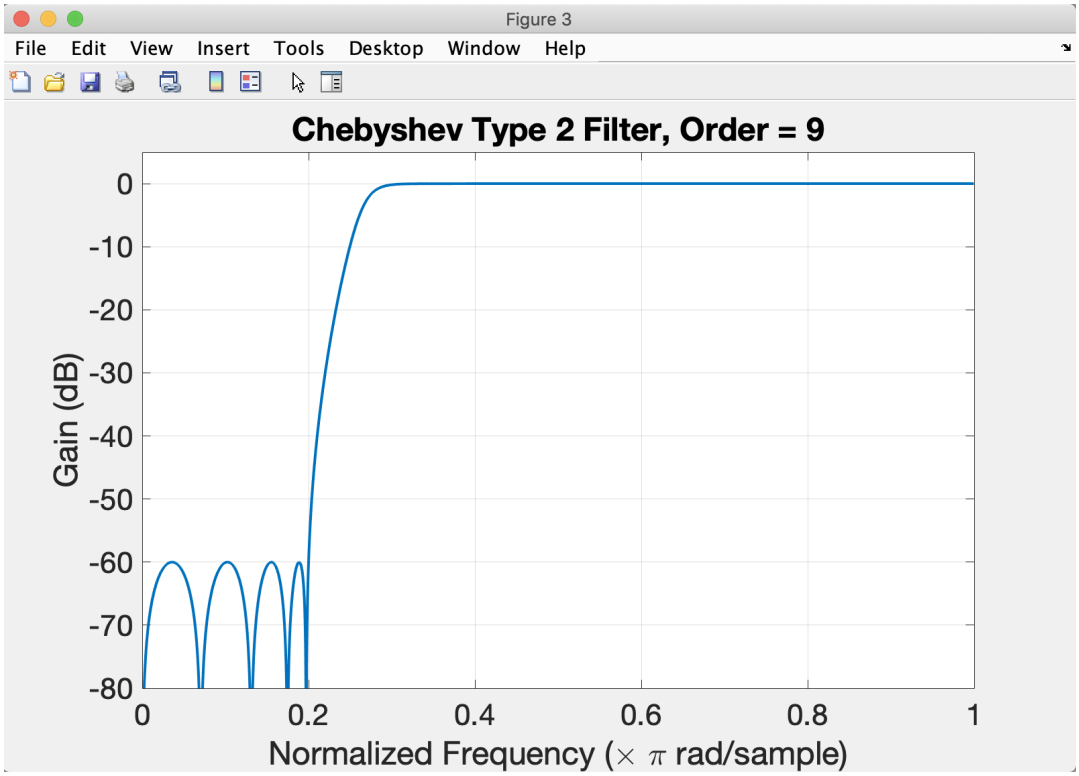


Figure 3
File Edit View Insert Tools Desktop Window Help
Chebyshev Type 2 Filter, Order = 9
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

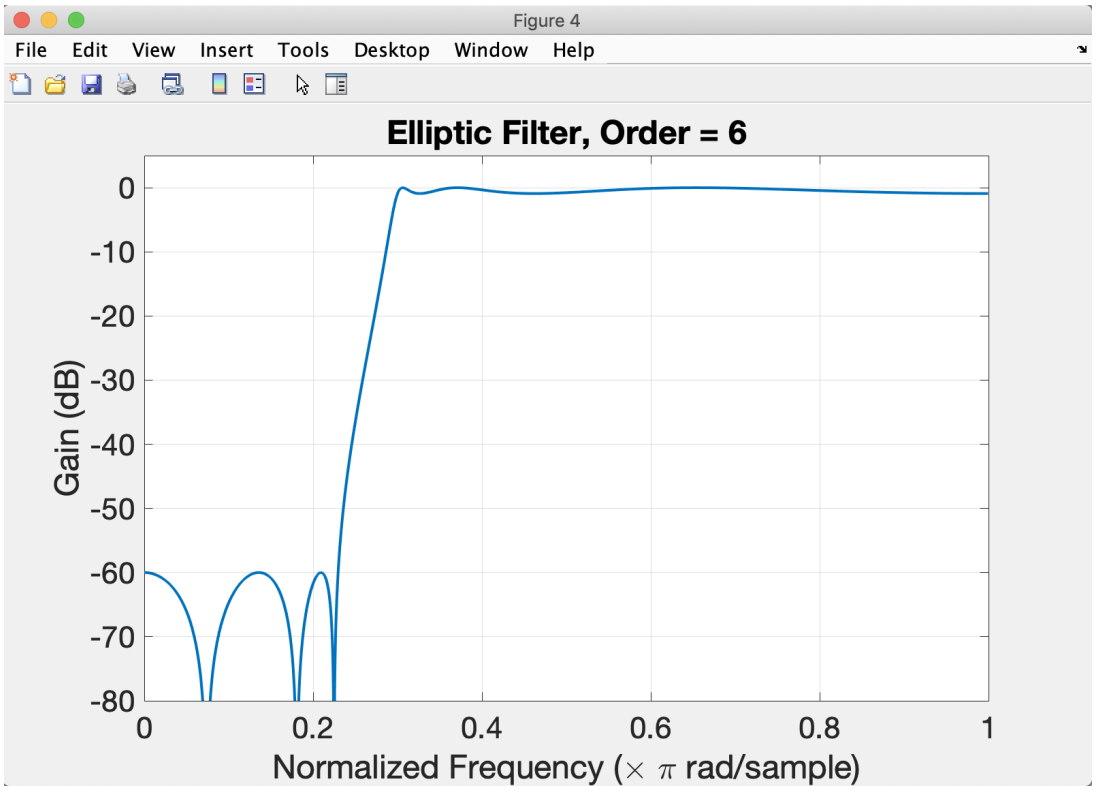
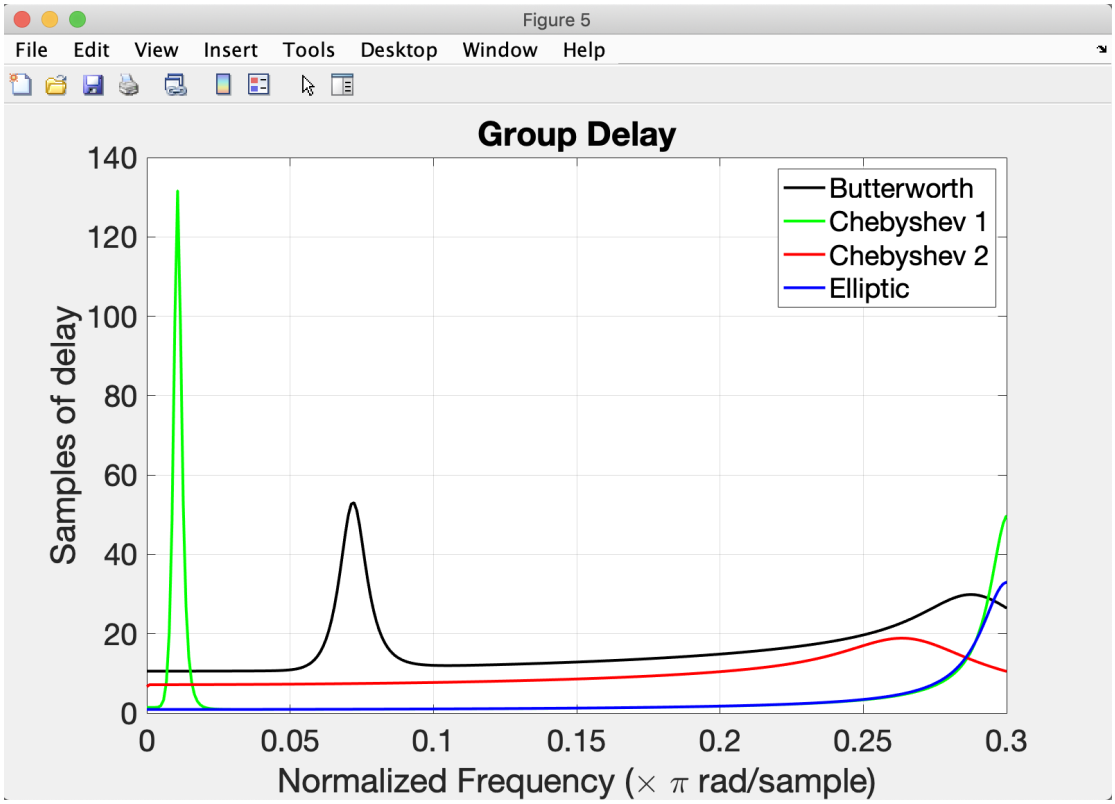


Figure 4
File Edit View Insert Tools Desktop Window Help
Elliptic Filter, Order = 6
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)



BAND PASS FILTER

CODE:

```
% Case 2: Bandpass filter

% Specifications:
% Passband 0.4pi to 0.6pi rads, gain 0.95 to 1
% Stopband 0 to 0.2pi and 0.8pi to pi rads, atten 0.01 = 40 dB %
% Compare Butterworth and elliptic filters

passband = [ 0.4*pi 0.6*pi]/pi; % convert to normalized frequency
stopband = [ 0.2*pi 0.8*pi]/pi;
passrip = -20*log10(0.95); % ripple in dB
stopatten = -20*log10(0.01); %stopband attenuation in dB

[Ne,Wne] = ellipord(passband,stopband,passrip,stopatten);
[Nc1, Wnc1] = cheb1ord(passband, stopband, passrip, stopatten); % Cheby 1 filter
[Nc2, Wnc2] = cheb2ord(passband, stopband, passrip, stopatten);% Cheby 2 filter
```

```
[Nb,Wnb] = buttord(passband,stopband,passrip,stopatten);

[Bc1,Ac1] = cheby1(Nc1,passrip,Wnc1);
[Bc2,Ac2] = cheby2(Nc2,stopatten,Wnc2);
[Be,Ae] = ellip(Ne,passrip,stopatten,Wne);
[Bb,Ab] = butter(Nb,Wnb);

figure(1)
[Hb,W]= freqz(Bb,Ab,2048);
h = plot(W/pi,20*log10(abs(Hb)));
set(h,'LineWidth',linwidth);
title(['Bandpass Butterworth Filter, Order = ',num2str(2*Nb)])
axis([0 1 -60, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(2)
[Hc1,W]= freqz(Bc1,Ac1,2048);
h = plot(W/pi,20*log10(abs(Hc1)));
set(h,'LineWidth',linwidth);
title(['BandPass Chebyshev Type 1 Filter, Order = ',num2str(Nc1)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(3)
[Hc2,W]= freqz(Bc2,Ac2,2048);
h = plot(W/pi,20*log10(abs(Hc2)));
set(h,'LineWidth',linwidth);
title(['BandPass Chebyshev Type 2 Filter, Order = ',num2str(Nc2)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
```

```
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(4)
[He,W]= freqz(Be,Ae,2048);
h = plot(W/pi,20*log10(abs(He)));
set(h,'LineWidth',linwidth);
title(['Bandpass Elliptic Filter, Order = ',num2str(2*Ne)])
axis([0 1 -60, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

% Compute group delay for these filters

[Gdb,W] = grpdelay(Bb,Ab,1024); % Gdb is group delay, W is freqs for Gdb
[Gdc1,W] = grpdelay(Bc1,Ac1,1024);
[Gdc2,W] = grpdelay(Bc2,Ac2,1024);
[Gde,W] = grpdelay(Be,Ae,1024);

% Display limited range (passband vicinity) of the group delay in normalized freq
Wnm = W/pi; % divide by pi to normalize W to 0 to 1,
% select first 150 frequencies to capture passband of filter (0 to 0.1pi)

figure(5)
h = plot(Wnm,Gdb,'k',Wnm,Gdc1,'g',Wnm,Gdc2,'r',Wnm,Gde,'b');
set(h,'LineWidth',linwidth);
legend('Butterworth','Chebyshev 1', 'Chebyshev 2', 'Elliptic')
title('Group Delay of BandPass')
xlabel('Normalized Frequency (\times \pi rad/sample)')
ylabel('Samples of delay')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])
```

OUTPUT :

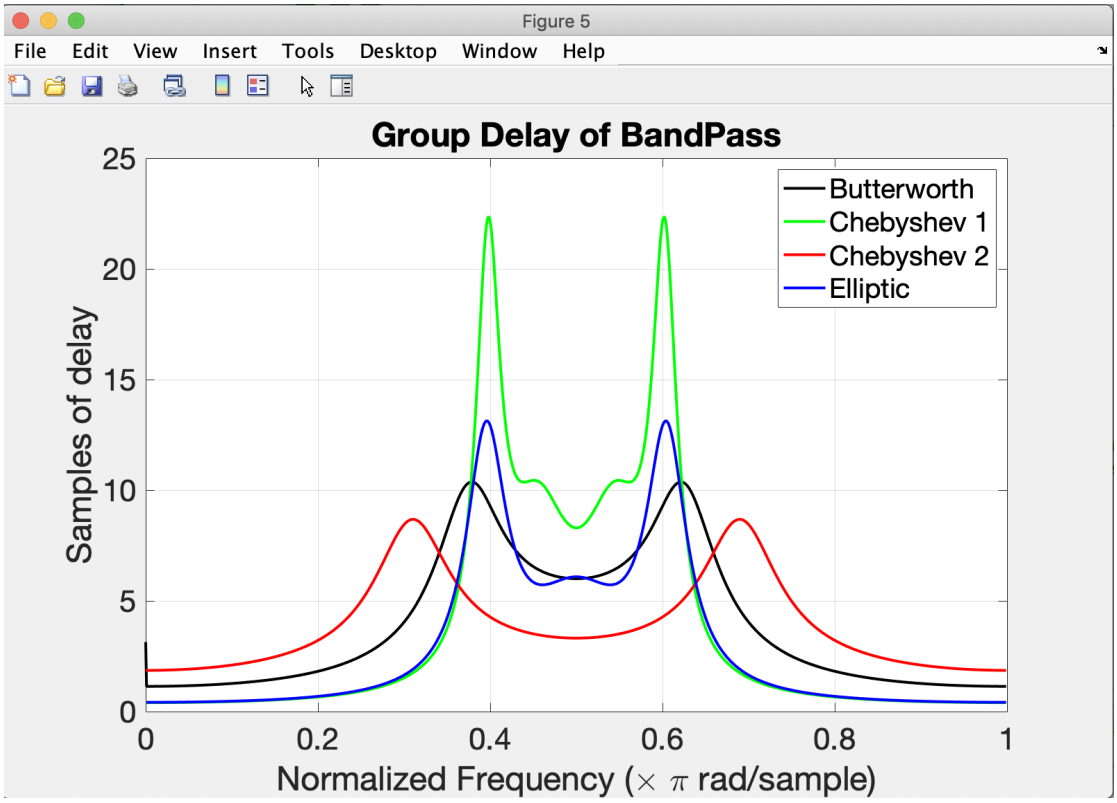


Figure 5
File Edit View Insert Tools Desktop Window Help
Group Delay of BandPass
Butterworth
Chebyshev 1
Chebyshev 2
Elliptic
Samples of delay
0
5
10
15
20
25
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

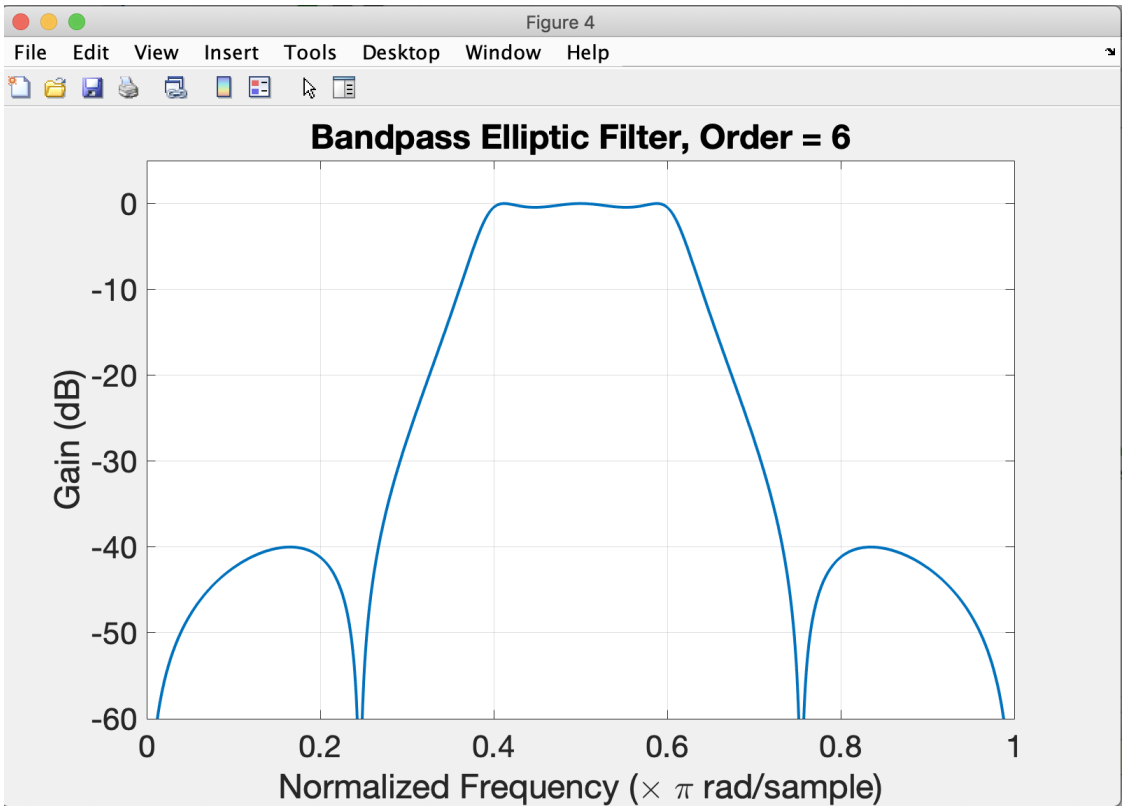


Figure 4
File Edit View Insert Tools Desktop Window Help
Bandpass Elliptic Filter, Order = 6
Gain (dB)
0
-10
-20
-30
-40
-50
-60
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

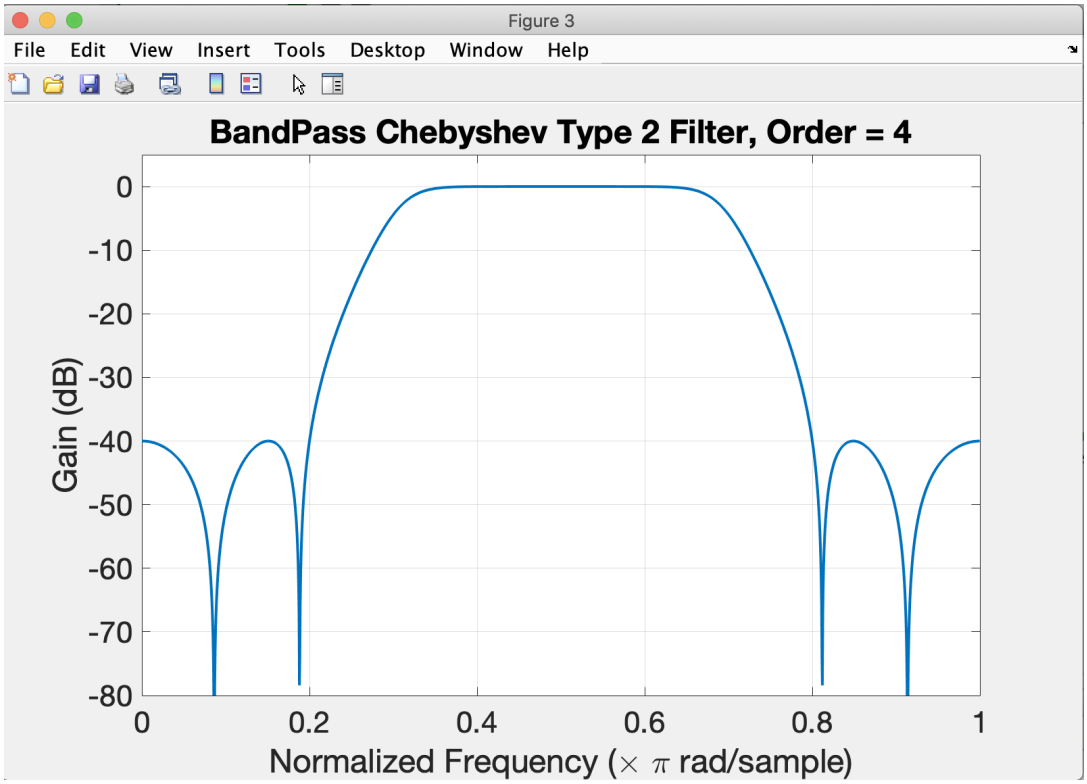


Figure 3
File Edit View Insert Tools Desktop Window Help
BandPass Chebyshev Type 2 Filter, Order = 4
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

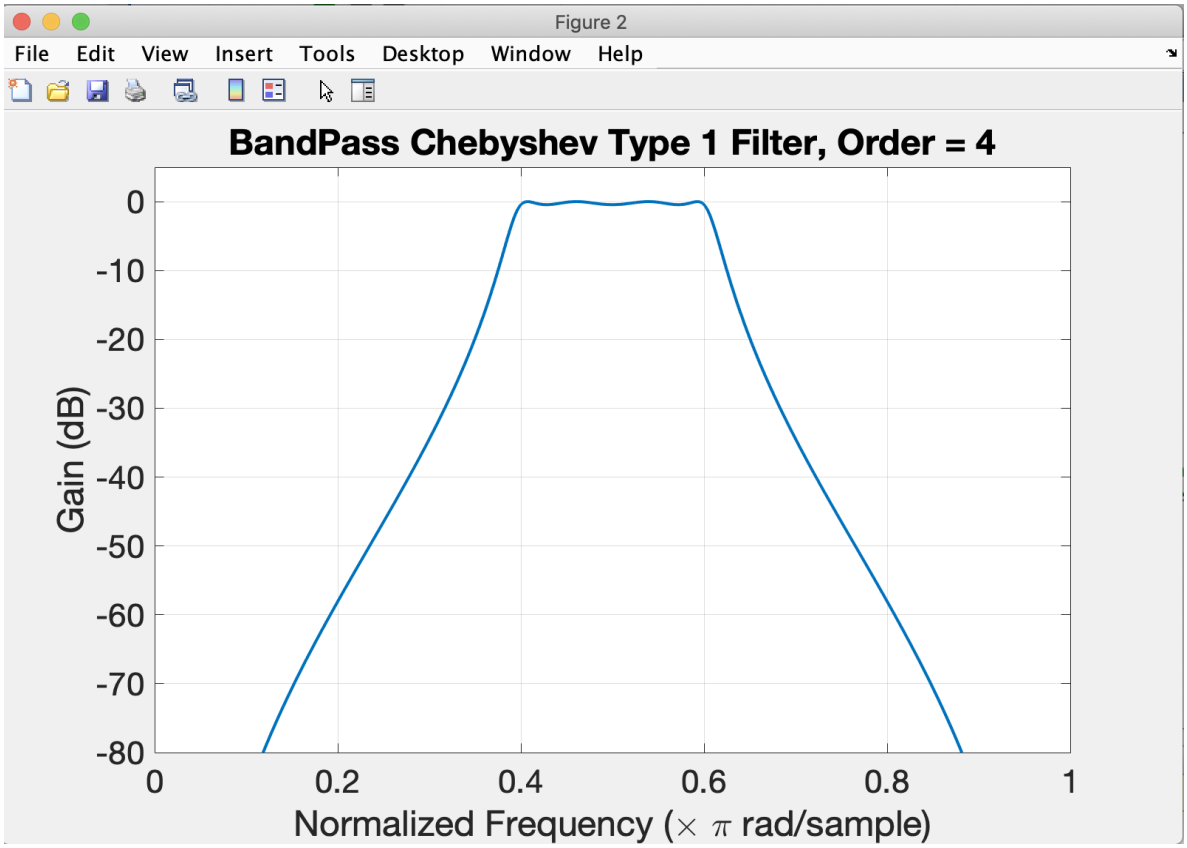


Figure 2
File Edit View Insert Tools Desktop Window Help
BandPass Chebyshev Type 1 Filter, Order = 4
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

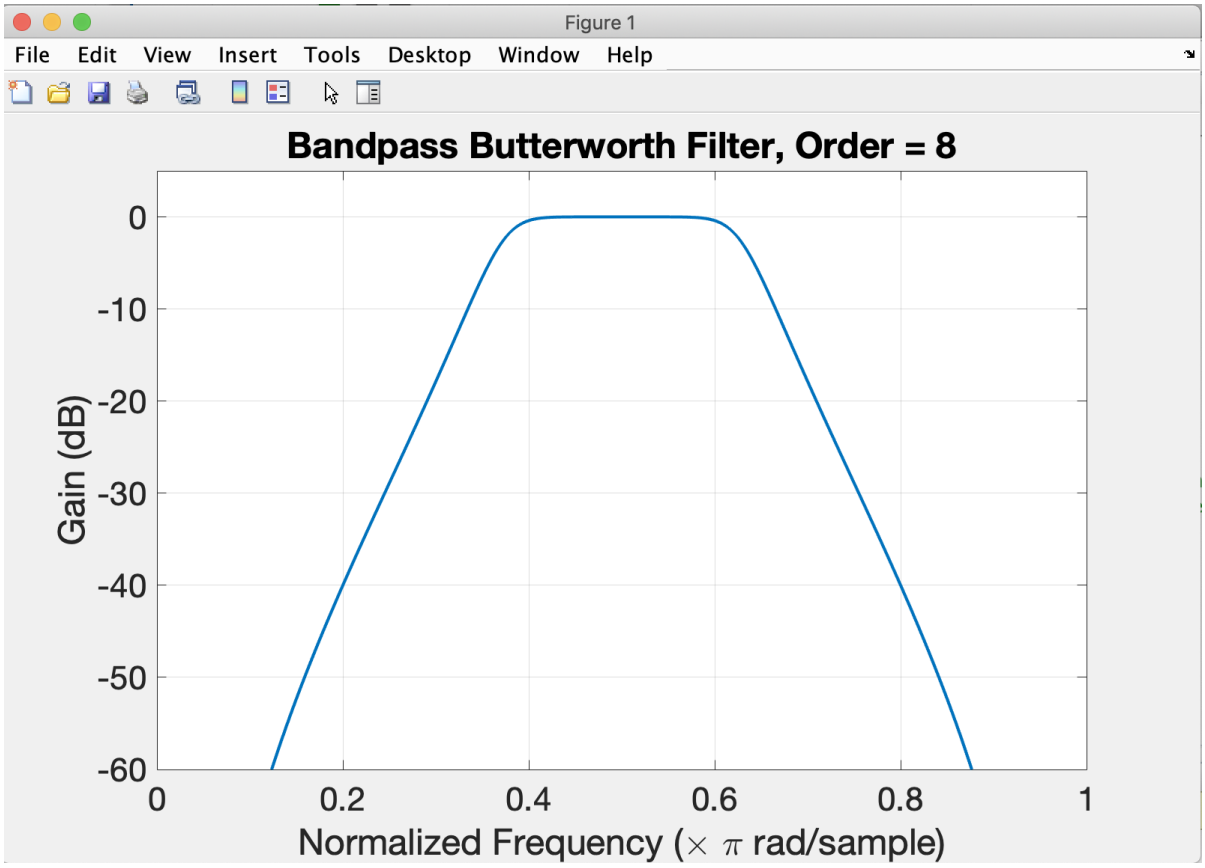


Figure 1
File Edit View Insert Tools Desktop Window Help
Bandpass Butterworth Filter, Order = 8
Gain (dB)
0
-10
-20
-30
-40
-50
-60
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

Comparision with Chebyshev type I and Chebyshev type II band pass filter response

In Chebyshev type I as we can see in all filters in pass band the gain in not constant is varies slightly and in Chebyshev type II there are many ripple between stop band

BAND STOP FILTER

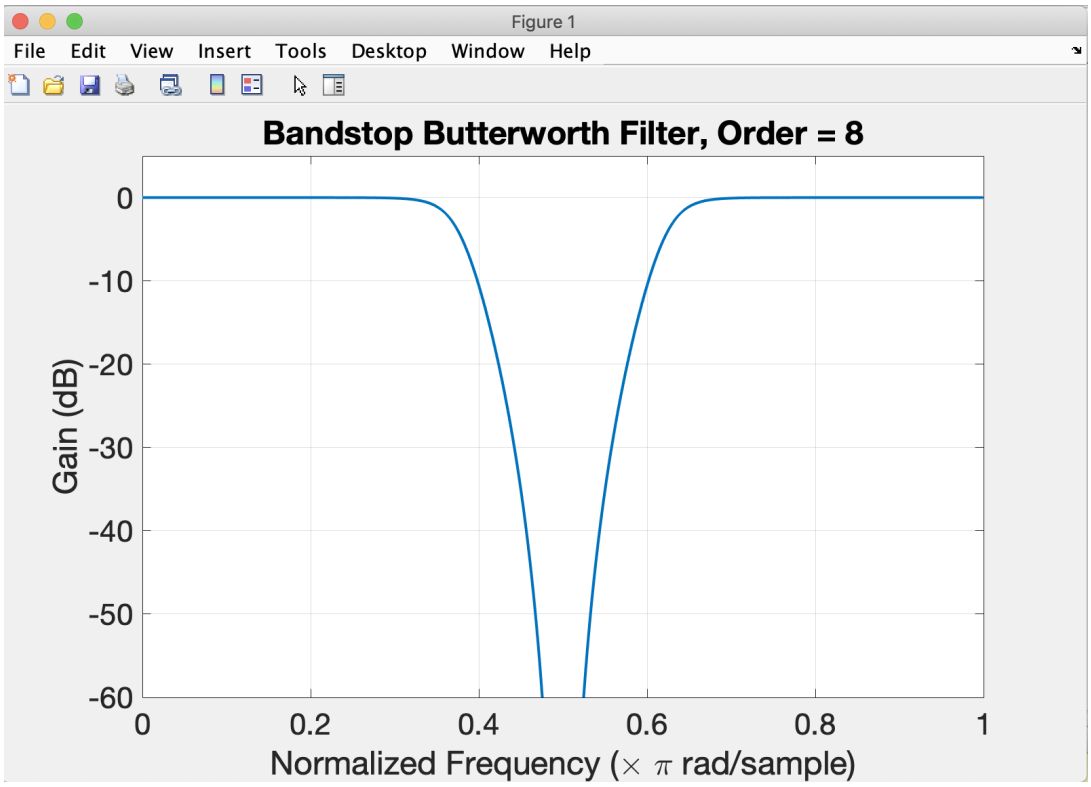
CODE:

```
% Case 2: Bandstop filter
% Specifications:
% Passband 0.4pi to 0.6pi rads, gain 0.95 to 1
% Stopband 0 to 0.2pi and 0.8pi to pi rads, atten 0.01 = 40 dB
% Compare Butterworth and elliptic filters

passband = [ 0.4*pi 0.6*pi]/pi; % convert to normalized frequency
stopband = [ 0.2*pi 0.8*pi]/pi;

passrip = -20*log10(0.95); % ripple in dB
stopatten = -20*log10(0.01); %stopband attenuation in dB

[Ne,Wne] = ellipord(passband,stopband,passrip,stopatten);
[Nc1, Wnc1] = cheb1ord(passband, stopband, passrip, stopatten); % Cheby 1 filter
[Nc2, Wnc2] = cheb2ord(passband, stopband, passrip, stopatten);% Cheby 2 filter
[Nb,Wnb] = buttord(passband,stopband,passrip,stopatten);

[Bc1,Ac1] = cheby1(Nc1,passrip,Wnc1,'stop');
[Bc2,Ac2] = cheby2(Nc2,stopatten,Wnc2,'stop');
[Be,Ae] = ellip(Ne,passrip,stopatten,Wne,'stop');
[Bb,Ab] = butter(Nb,Wnb,'stop');

figure(1)
[Hb,W]= freqz(Bb,Ab,2048);
h = plot(W/pi,20*log10(abs(Hb)));
set(h,'LineWidth',linwidth);
title(['Bandstop Butterworth Filter, Order = ',num2str(2*Nb)])
axis([0 1 -60, 5])
ylabel('Gain (dB)')
```

```
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(2)
[Hc1,W]= freqz(Bc1,Ac1,2048);
h = plot(W/pi,20*log10(abs(Hc1)));
set(h,'LineWidth',linwidth);
title(['Bandstop Chebyshev Type 1 Filter, Order = ',num2str(Nc1)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(3)
[Hc2,W]= freqz(Bc2,Ac2,2048);
h = plot(W/pi,20*log10(abs(Hc2)));
set(h,'LineWidth',linwidth);
title(['Bandstop Chebyshev Type 2 Filter, Order = ',num2str(Nc2)])
axis([0 1 -80, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])

figure(4)
[He,W]= freqz(Be,Ae,2048);
h = plot(W/pi,20*log10(abs(He)));
set(h,'LineWidth',linwidth);
title(['Bandstop Elliptic Filter, Order = ',num2str(2*Ne)])
axis([0 1 -60, 5])
ylabel('Gain (dB)')
xlabel('Normalized Frequency (\times \pi rad/sample)')
grid on;
h = get(0,'CurrentFigure'); set(h,'Position',[160,180,800,500])
```

```
% Compute group delay for these filters
[Gdb,W] = grpdelay(Bb,Ab,1024); % Gdb is group delay, W is freqs for Gdb
[Gdc1,W] = grpdelay(Bc1,Ac1,1024);
[Gdc2,W] = grpdelay(Bc2,Ac2,1024);
[Gde,W] = grpdelay(Be,Ae,1024);

% Display limited range (passband vicinity) of the group delay in normalized freq
Wnm = W/pi; % divide by pi to normalize W to 0 to 1,
% select first 150 frequencies to capture passband of filter (0 to 0.1pi)

figure(5)
h = plot(Wnm,Gdb,'k',Wnm,Gdc1,'g',Wnm,Gdc2,'r',Wnm,Gde,'b');
set(h,'LineWidth',linwidth);
legend('Butterworth','Chebyshev 1', 'Chebyshev 2', 'Elliptic')
title('Group Delay of Bandstop')
xlabel('Normalized Frequency (\times \pi rad/sample)')
ylabel('Samples of delay')
grid on;
h = get(0,'CurrentFigure');
set(h,'Position',[160,180,800,500])
```

OUTPUT :



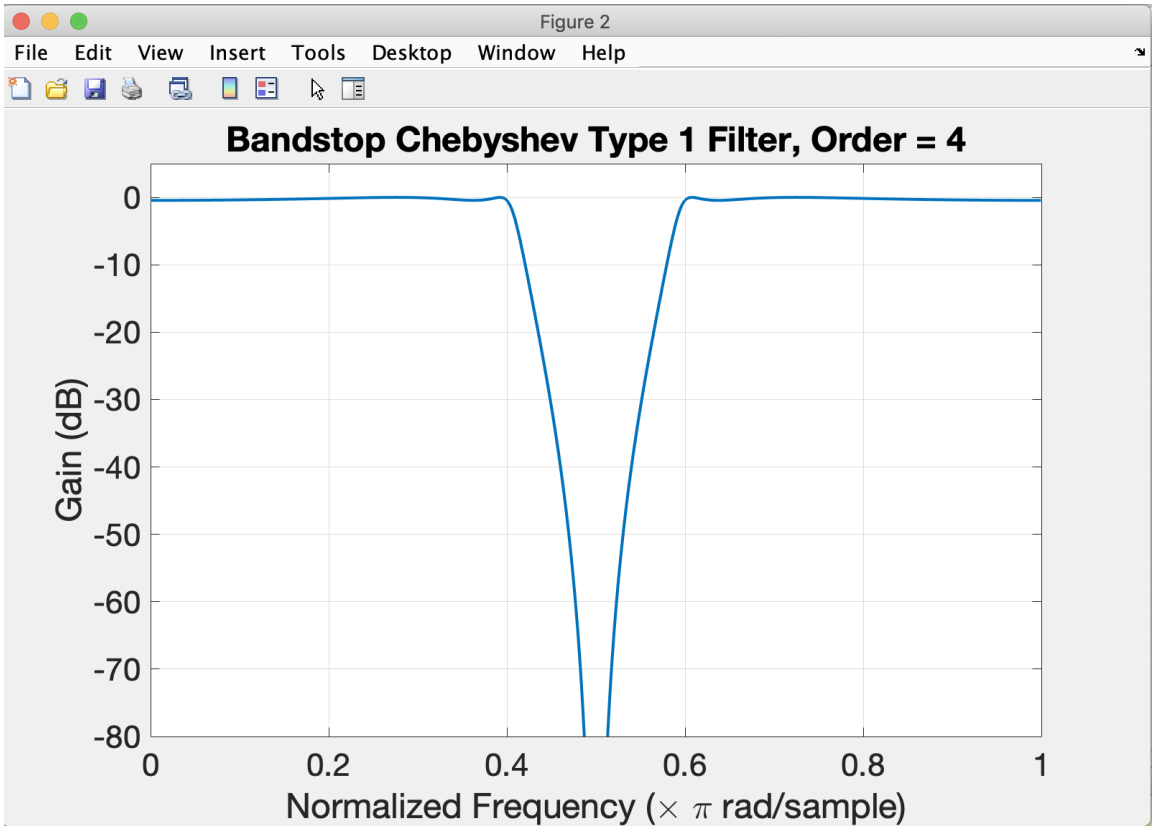


Figure 2
File Edit View Insert Tools Desktop Window Help
Bandstop Chebyshev Type 1 Filter, Order = 4
Gain (dB)
0
-10
-20
-30
-40
-50
-60
-70
-80
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)

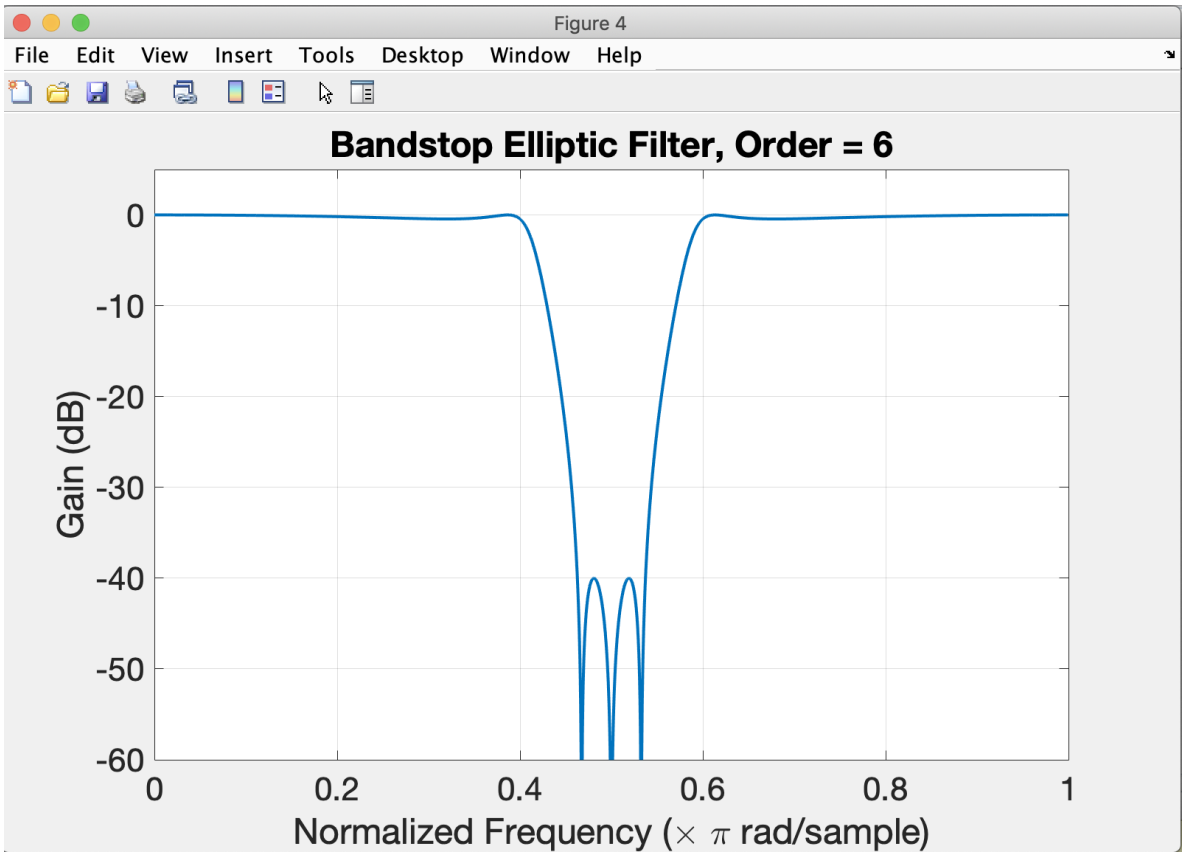
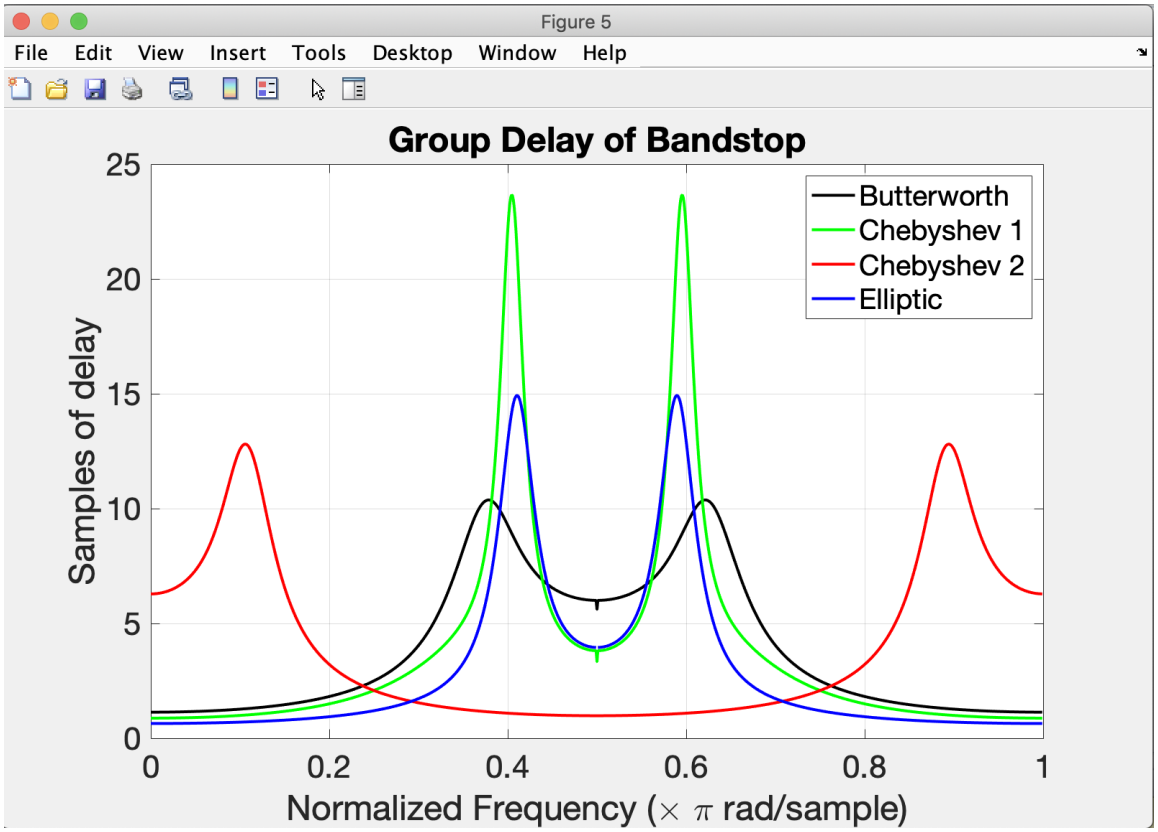


Figure 4
File Edit View Insert Tools Desktop Window Help
Bandstop Elliptic Filter, Order = 6
Gain (dB)
0
-10
-20
-30
-40
-50
-60
0
0.2
0.4
0.6
0.8
1
Normalized Frequency (× π rad/sample)



Comparison of all high pass response , Band stop response in terms of ripple existence

In highpass , the ripples are from –infinity to the start of a frequency of highpass filter so there may be so many ripple but in bandstop filter the ripples are between 2 frequencies so they are limited

CONCLUSION:

(5) Overall, which type is preferable ?Why?

I would prefer the butterworth because the pass band is having a constant gain and there are no ripples in this type whether it is low pass, high pass , band pass or band stop

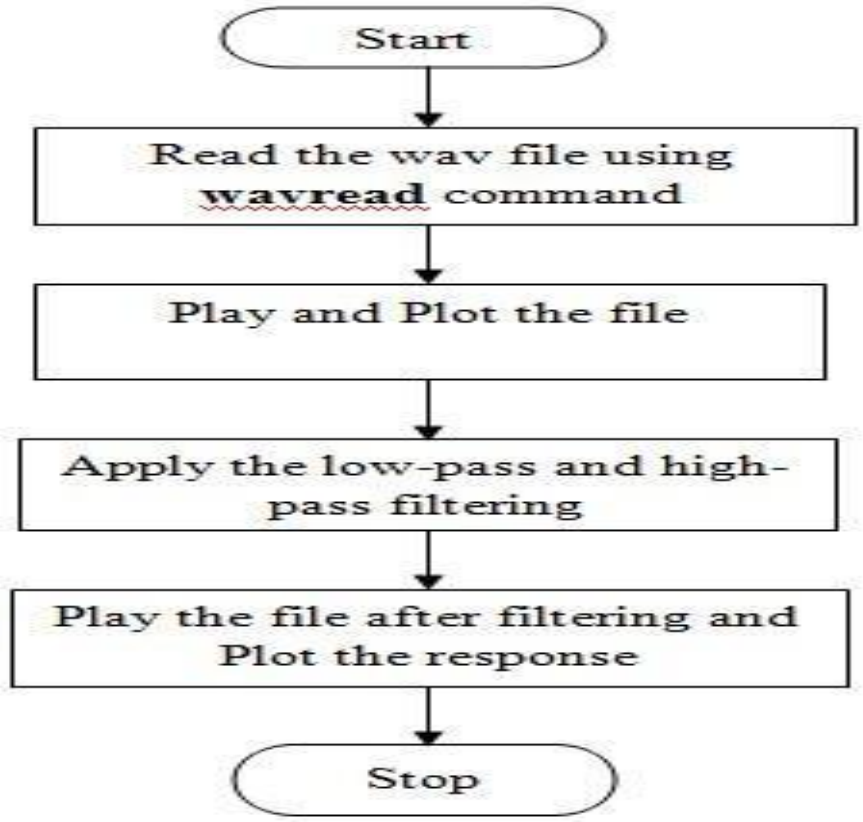
Aim:

a) Read a '.wav' file in MATLAB. Plot it in the time domain and frequency domain.

b) Apply upsampling and downsampling and plot and plat that wave file

c)Apply the low pass filter, high pass filter, plot the corresponding waveforms in each case and listen to the result of the filter.

d) Record your own speech and also perform the same a), b) and c) file for your recorded version.



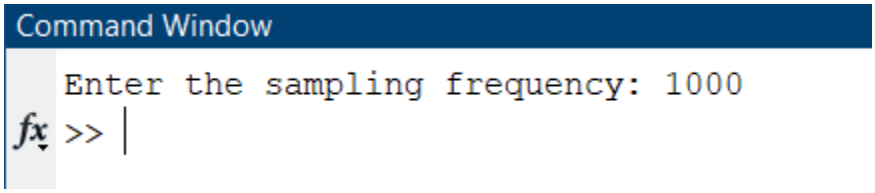
❖ Music Audio

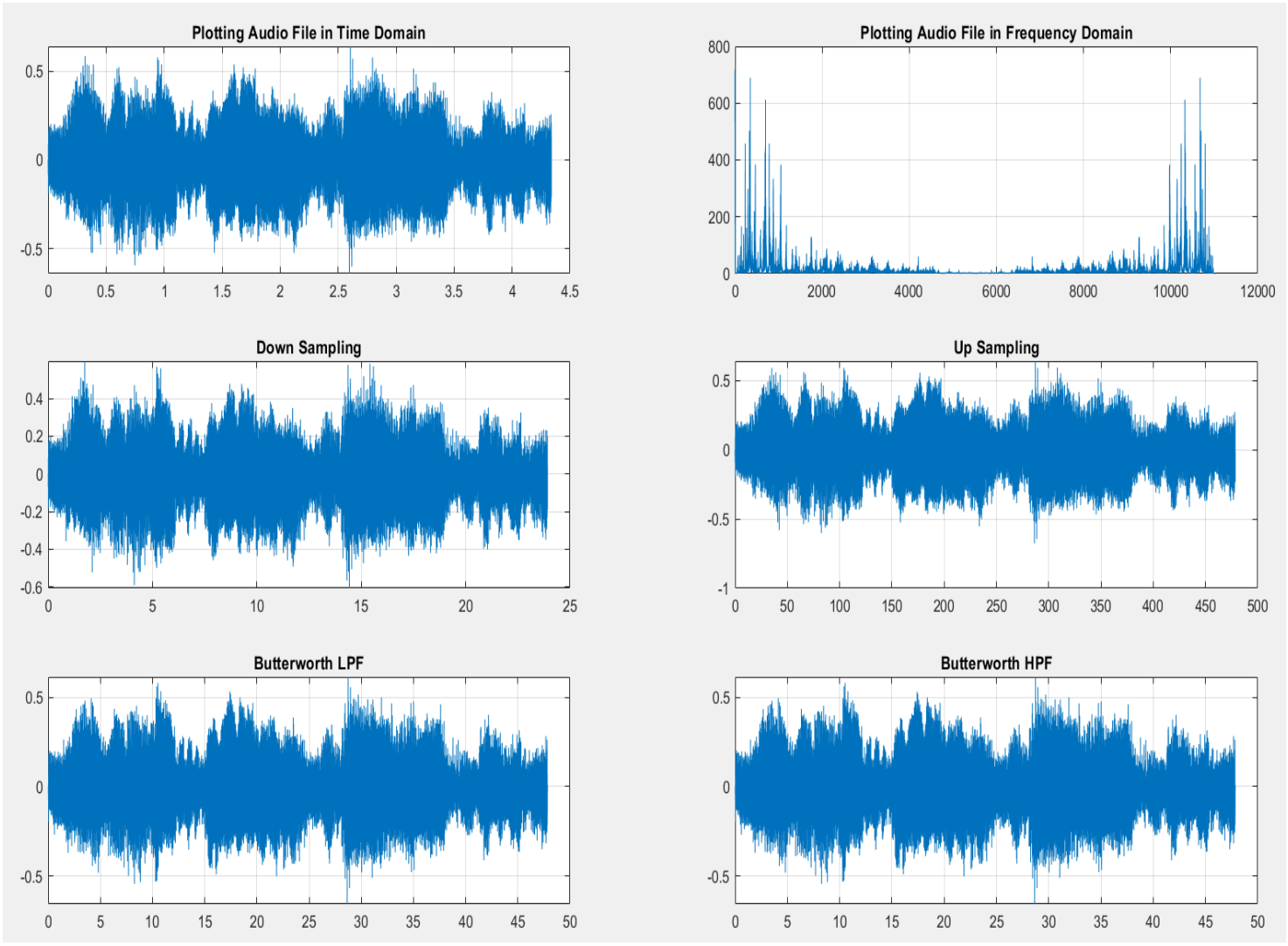
Code:

```
clc;
clear all; close all;
% Plotting Audio File in Time Domain
fs = input('Enter the sampling frequency: ');
```

```
%a=audiorecorder(fs,8,1);
%record(a,5);
%pause(5);
%b=getaudiodata(a);
%sound(b);
%audiowrite('C:\Users\LENOVO\Desktop\SEM-7\DSP\LAB9\s.wav',double(b),fs);
[wave,f] = audioread('fantastic.wav');
ts = 1/f;
t = 0:ts:(length(wave)-1)*ts; subplot(3,2,1);
plot(t,wave);
title('Plotting Audio File in Time Domain'); grid on;

% Plotting Audio File in Frequency Domain
n = length(wave)-1;
f = 0:f/n:f;
wavefft = abs(fft(wave)); subplot(3,2,2); plot(f,wavefft);
title('Plotting Audio File in Frequency Domain'); grid on;

% Down Sampling
y1 = resample(wave,1,2);
t1 = 0:1/fs:(length(y1)-1)/fs; subplot(3,2,3);
plot(t1,y1); title('Down Sampling'); grid on;
%sound(y1,fs)

% Up Sampling
y2 = resample(wave,10,1);
t2 = 0:1/fs:(length(y2)-1)/fs; subplot(3,2,4);
plot(t2,y2); title('Up Sampling'); grid on;
%sound(y2,fs)

% Butterworth LPF
[b,a] = butter(50,0.5,'low'); y3 = filter(b,a,wave);
t3 = 0:1/fs:(length(y3)-1)/fs; subplot(3,2,5);
plot(t3,y3); title('Butterworth LPF'); grid on;
%sound(y3,fs)

% Butterworth HPF
[b,a] = butter(50,0.5,'high'); y4 = filter(b,a,wave);
t4 = 0:1/fs:(length(y4)-1)/fs; subplot(3,2,6);
plot(t4,y3); title('Butterworth HPF'); grid on;
%sound(y4,fs)
```

Output:





e:

For Recording Audio

```
clc;
clear all; close all;
%  I am XYZ(your name) . I am doing  waveread command for speech signal in DSP lab
fs = input('Enter the sampling frequency: ');
```

```
a=audiorecorder(fs,8,1);
display('Recording...');
record(a,5);
pause(5);
display('Recorded..');
b=getaudiodata(a);
sound(b);
audiowrite('C:\Users\LENOVO\Desktop\SEM-7\DSP\LAB9\s.wav',double(b),fs);
```

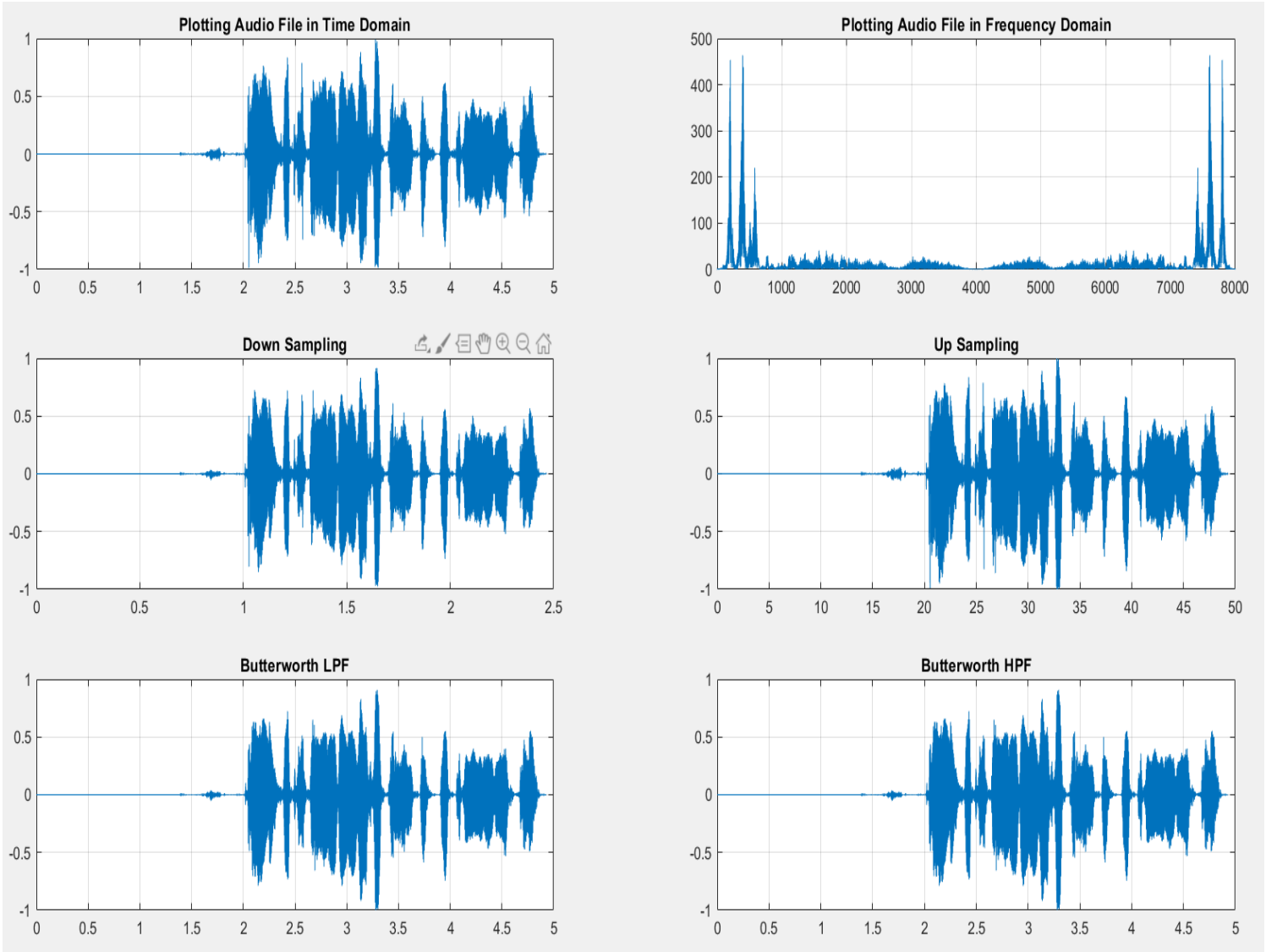
For Plotting

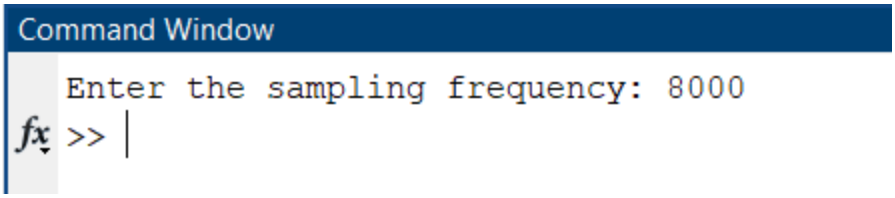
```
clc;
clear all; close all;
% Plotting Audio File in Time Domain
fs = input('Enter the sampling frequency: ');
[wave,f] = audioread('s.wav');
ts = 1/f;
t = 0:ts:(length(wave)-1)*ts; subplot(3,2,1);
plot(t,wave);
title('Plotting Audio File in Time Domain'); grid on;

% Plotting Audio File in Frequency Domain
n = length(wave)-1;
f = 0:f/n:f;
wavefft = abs(fft(wave)); subplot(3,2,2); plot(f,wavefft);
title('Plotting Audio File in Frequency Domain'); grid on;

% Down Sampling
y1 = resample(wave,1,2);
t1 = 0:1/fs:(length(y1)-1)/fs; subplot(3,2,3);
plot(t1,y1); title('Down Sampling'); grid on;
%sound(y1,fs)

% Up Sampling
y2 = resample(wave,10,1);
t2 = 0:1/fs:(length(y2)-1)/fs; subplot(3,2,4);
plot(t2,y2); title('Up Sampling'); grid on;
%sound(y2,fs)

% Butterworth LPF
[b,a] = butter(50,0.5,'low'); y3 = filter(b,a,wave);
t3 = 0:1/fs:(length(y3)-1)/fs; subplot(3,2,5);
plot(t3,y3); title('Butterworth LPF'); grid on;
```

```
%sound(y3,fs)

% Butterworth HPF
[b,a] = butter(50,0.5,'high'); y4 = filter(b,a,wave);
t4 = 0:1/fs:(length(y4)-1)/fs; subplot(3,2,6);
plot(t4,y3); title('Butterworth HPF'); grid on;
%sound(y4,fs)
```

Output:

```
Command Window
   Enter the sampling frequency: 8000
   Recording...
   Recorded.
fx >>
```




Command Window
Enter the sampling frequency: 8000
>>

Lab10_FIR filter using windowing techniques

AIM : Design a 25-tap Bandpass Filter for the following specifications:
- Cut-off frequencies: 0.25pi and 0.75pi
- Windowing: Rectangular and Hamming
- Order: 25
- Compare different Windowing Techniques and comment on the same.

CODE :

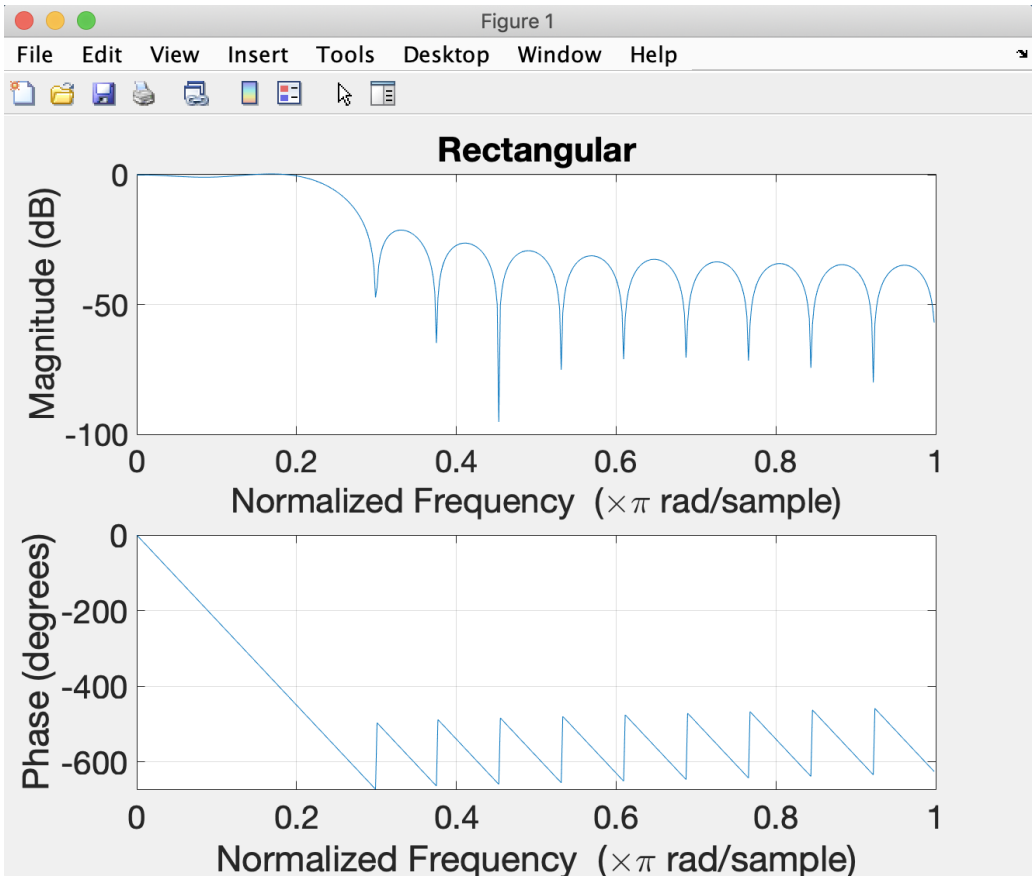
```
clc
clear all
close all

N=input('Enter order of filter:');
Wc=input('Ener cutoff frequency:');

y=rectwin(N+1); %%rectangular filter
r=fir1(N,Wc,y);
figure(1);
freqz(r);
title('Rectangular')

y1=chebwin(N+1); %%chebyshev filter
r=fir1(N,Wc,y1);
figure(2);
freqz(r);
title('Chebyshev')

y2=hanning(N+1); %%hanning filter
r=fir1(N,Wc,y2);
figure(3);
freqz(r);
title('Hanning')

y3=hamming(N+1); %%hamming filter
r=fir1(N,Wc,y3);
figure(4);
freqz(r);
```

```
title('Hamming')
```

OUTPUT :

Enter order of filter:25

Ener cutoff frequency:0.25



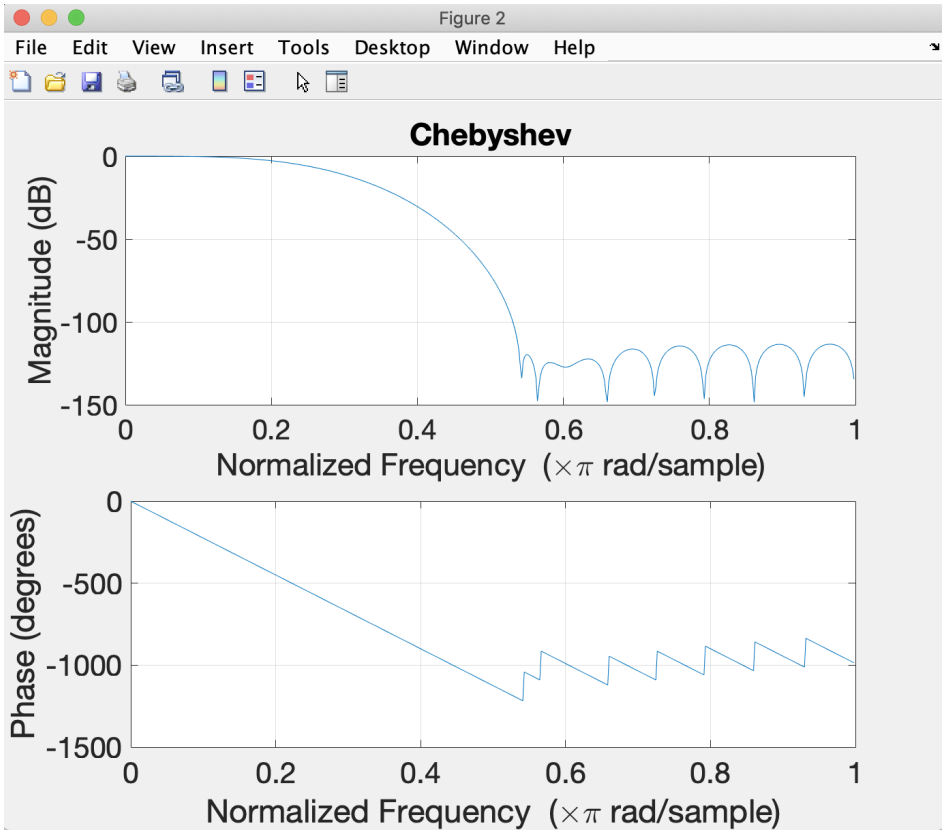


Figure 2
File Edit View Insert Tools Desktop Window Help
Chebyshev
Magnitude (dB)
0
-50
-100
-150
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-500
-1000
-1500
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

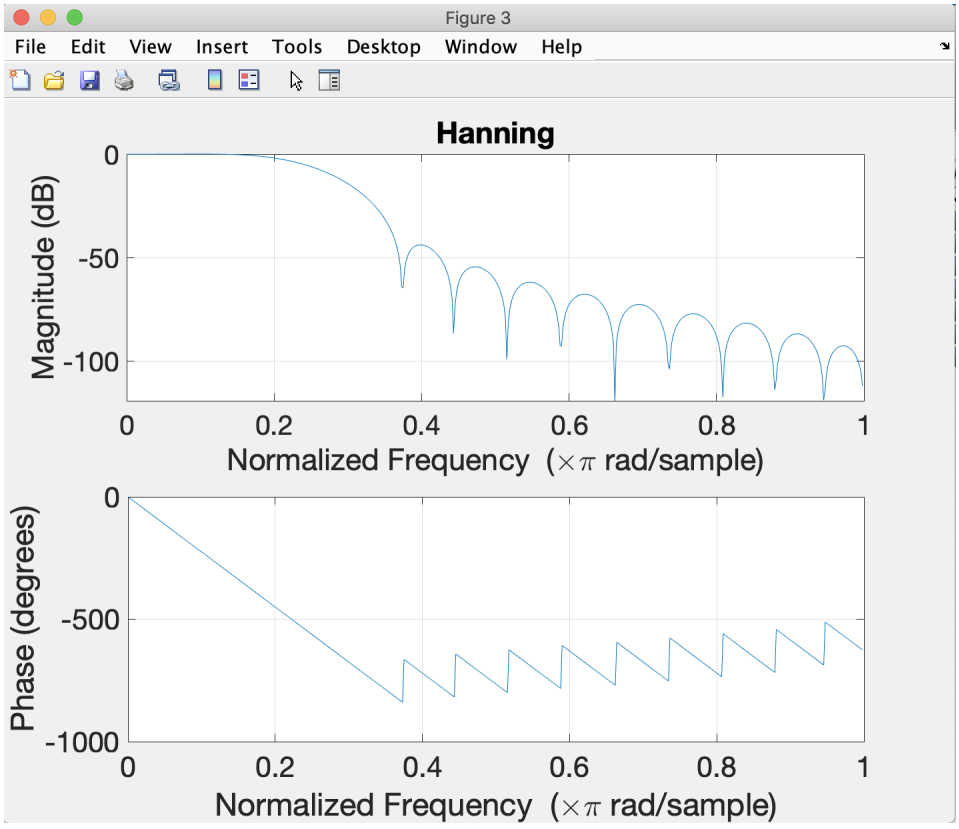
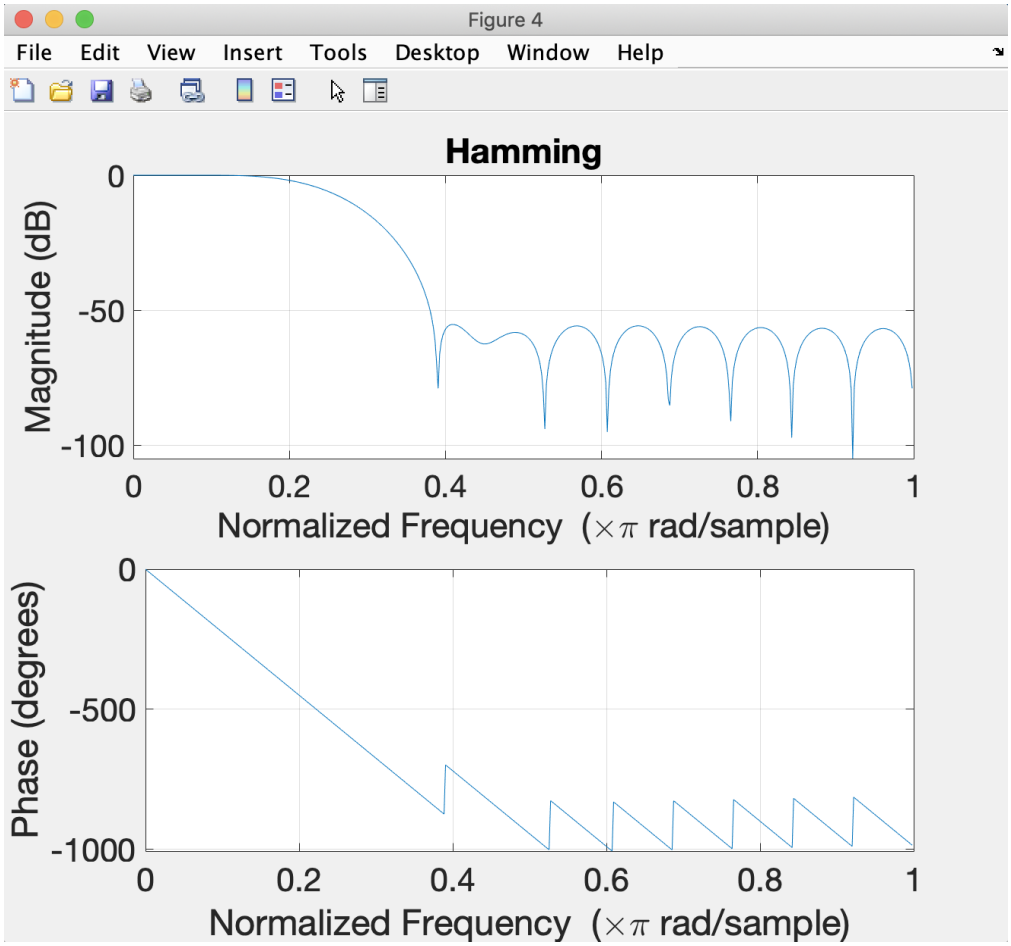


Figure 3
File Edit View Insert Tools Desktop Window Help
Hanning
Magnitude (dB)
0
-50
-100
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-500
-1000
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)



Enter order of filter:25
Ener cutoff frequency:0.75

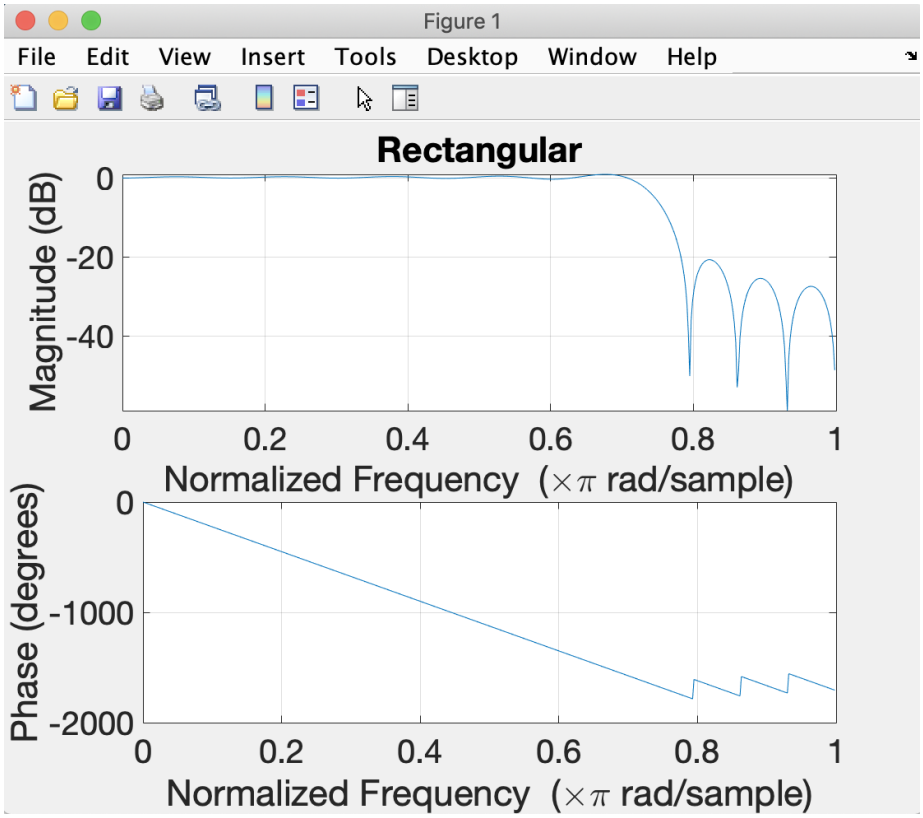


Figure 1
File Edit View Insert Tools Desktop Window Help
Rectangular
Magnitude (dB)
0
-20
-40
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-1000
-2000
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

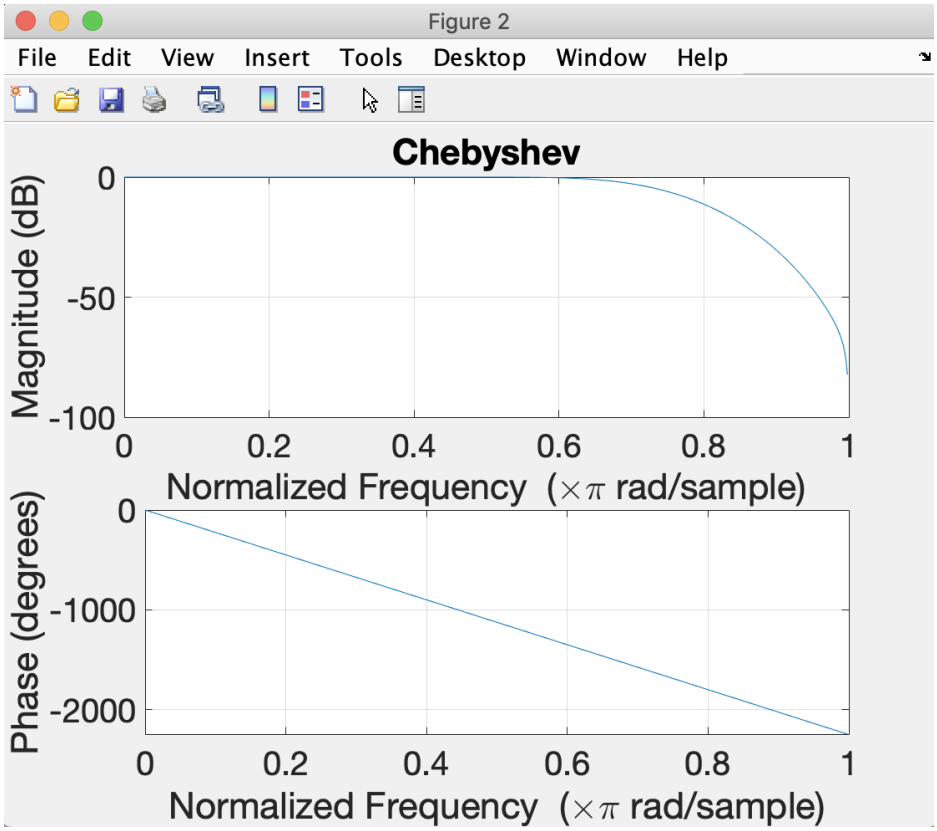


Figure 2
File Edit View Insert Tools Desktop Window Help
Chebyshev
Magnitude (dB)
0
-50
-100
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-1000
-2000
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

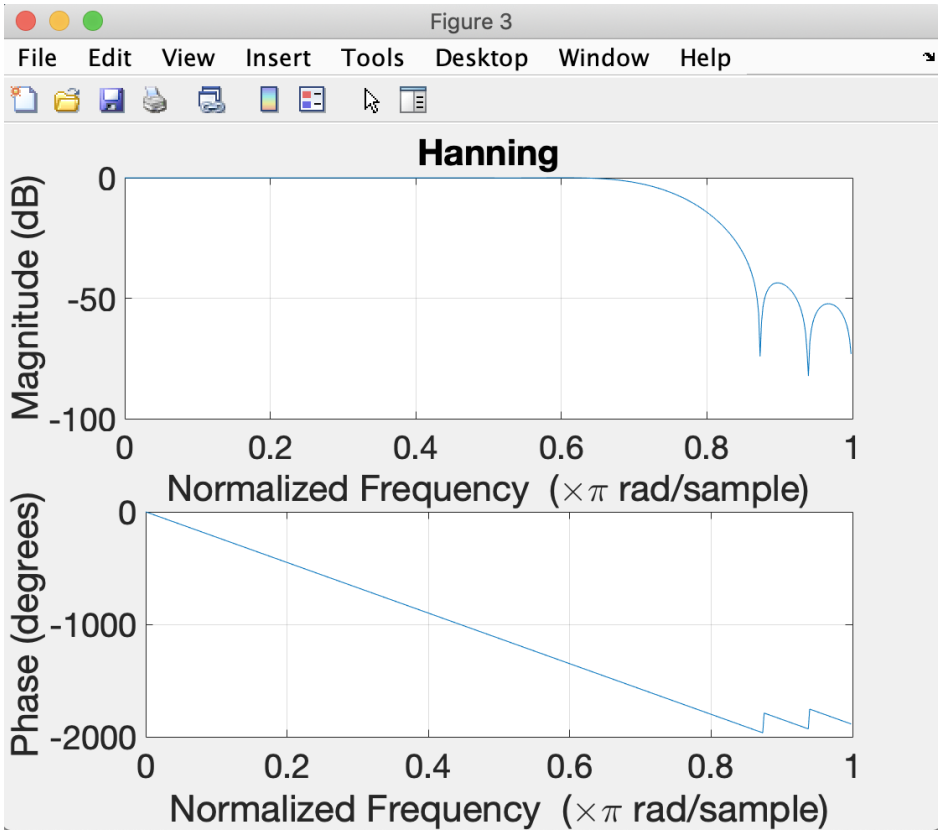


Figure 3
File Edit View Insert Tools Desktop Window Help
Hanning
Magnitude (dB)
0
-50
-100
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-1000
-2000
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

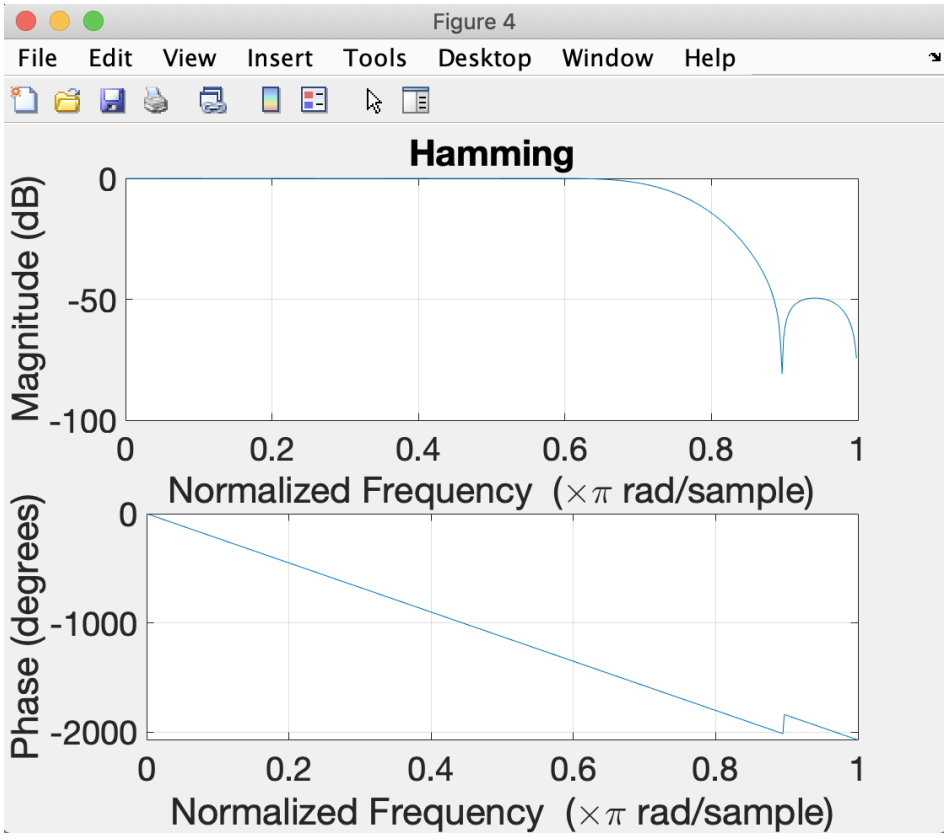


Figure 4
File Edit View Insert Tools Desktop Window Help
Hamming
Magnitude (dB)
0
-50
-100
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)
Phase (degrees)
0
-1000
-2000
0 0.2 0.4 0.6 0.8 1
Normalized Frequency (×π rad/sample)

CONCLUSION : In this experiment we observed and learned about windowing technique and from the graphs we can say that cheyshev filter should be used as its ripple has less magnitude and at high cut off frequency

Lab12_Keiser window

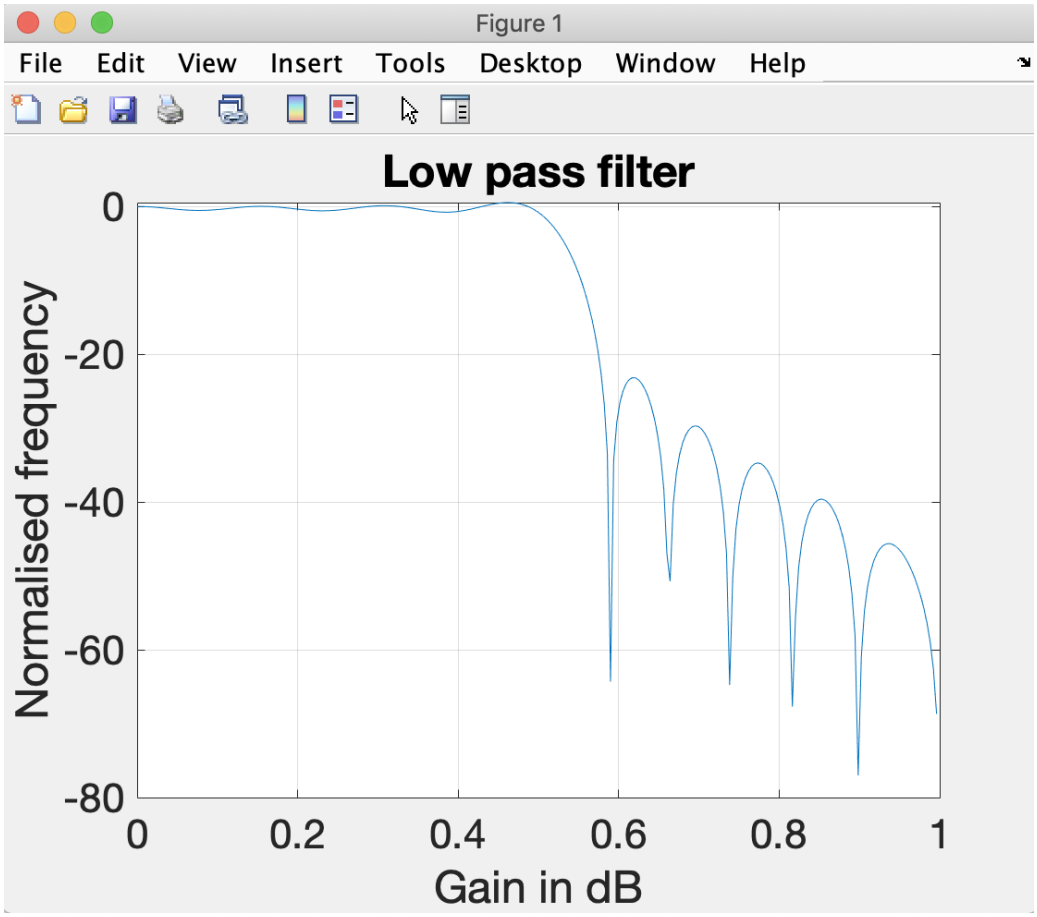
AIM : Plot the Frequency response of LPF and HPF (with the following specifications) using different windowing technique especially Keiser Window for different values of β.
Cut-off frequency: (Your Roll no/100) * pi
Order of the filter N: 25
Beta: 0.5, 3.5, 8.5

CODE :

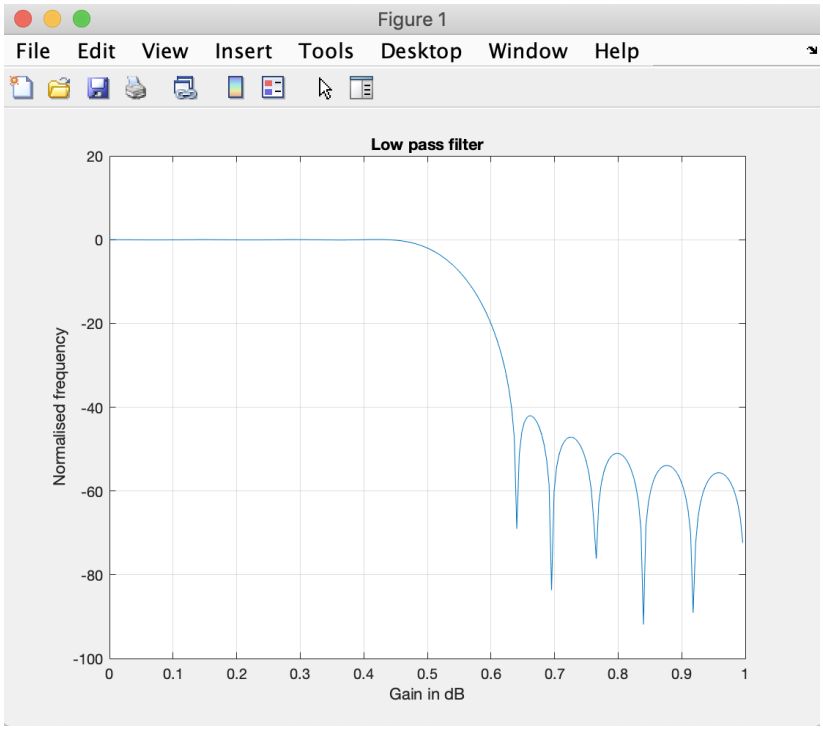
```
clc;
clear all;
close all;

n=input('Enter order of filter:'); %filter order
wc=input('Enter cutoff frequency:'); %cutoff frequency
beta=input('Enter value of beta:');

% Low pass filter
b=fir1(n,wc,'low',kaiser(n+1,beta)); %filter using kaiser window
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
plot(o/pi,m);
title('Low pass filter');
xlabel('Gain in dB');
ylabel('Normalised frequency');
grid on;
```

OUTPUT :

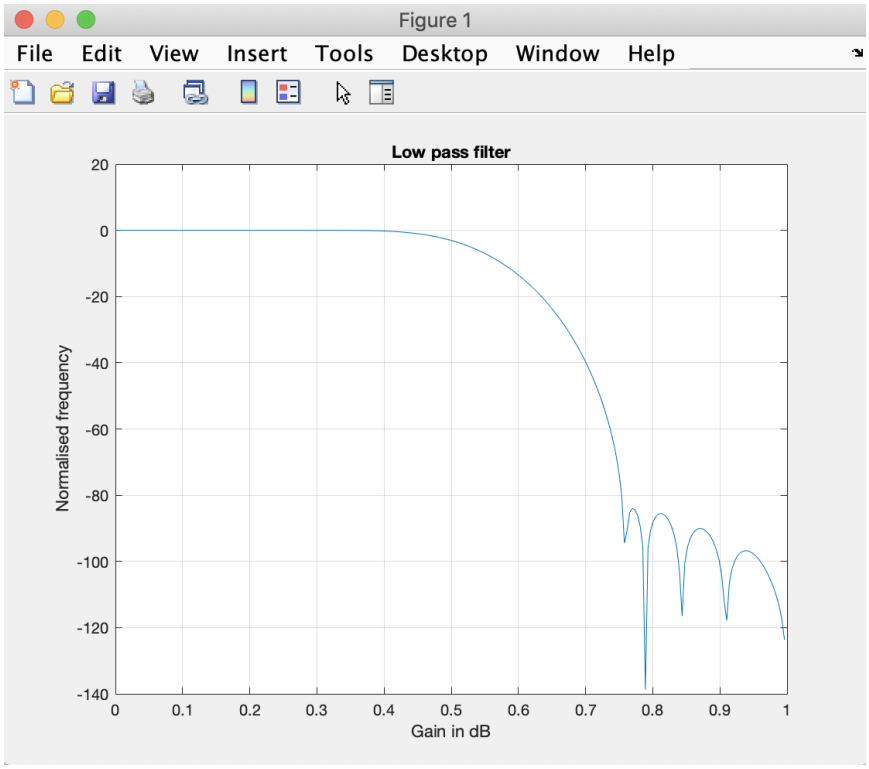
Enter order of filter:25
Enter cutoff frequency:0.54
Enter value of beta:0.5



Enter order of filter:25
Enter cutoff frequency:0.54
Enter value of beta:3.5



Enter order of filter:25
Enter cutoff frequency:0.54
Enter value of beta:8.5



CONCLUSION:

In this experiment we have performed and observed the windowing technique Keiser Window for different values of β. And we observed that a the value of Beta increased the LPF does not drop abruptly after the cut off frequency