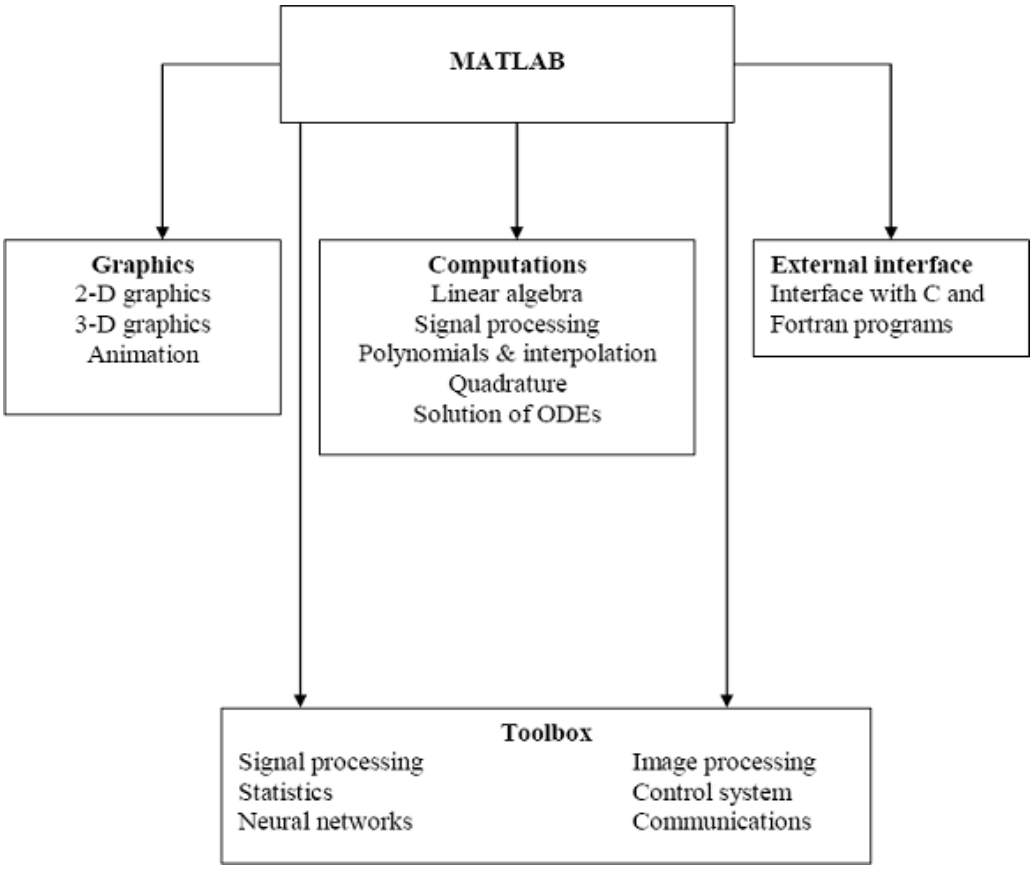
CONTENTS

INTRODUCTION

MATLAB stands for MATrix LABoratory. It is a technical computing environment for high performance numeric computation and visualisation. It integrates numerical analysis, matrix computation, signal processing and graphics in an easy-to-use environment, where problems and solutions are expressed just as they are written mathematically, without traditional programming. MATLAB allows us to express the entire algorithm in a few dozen lines, to compute the solution with great accuracy in a few minutes on a computer, and to readily manipulate a three-dimensional display of the result in colour.

MATLAB is an interactive system whose basic data element is a matrix that does not require dimensioning. It enables us to solve many numerical problems in a fraction of the time that it would take to write a program and execute in a language such as FORTRAN, BASIC, or C. It also features a family of application specific solutions, called ***toolboxes***. Areas in which toolboxes are available include signal processing, image processing, control systems design, dynamic systems simulation, systems identification, neural networks, wavelength communication and others. It can handle linear, non-linear, continuous-time, discrete-time, multivariable and multirate systems.



GENERATION OF CONTINUOUS TIME SIGNALS

AIM:

To generate a functional sequence of a signal (Sine, Cosine, triangular, Square, Saw tooth and sinc) using MATLAB function.

APPARATUS REQUIRED:

HARDWARE: Personal Computer

SOFTWARE :MATLABR2018a

PROCEDURE:

1. Start the MATLAB program.
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. If any error occurs in the program correct the error and run it again
7. For the output see command window\ Figure window
8. Stop the program.

PROGRAM: (Generation of Continuous Time Signals)

```
%Program for sine wave
t=0:0.1:10;
y=sin(2*pi*t);
subplot(3,3,1);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude') ;
title('Sine wave');

%Program for cosine wave
t=0:0.1:10;
y=cos(2*pi*t);
subplot(3,3,2);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Cosine wave');

%Program for square wave
t=0:0.001:10;
y=square(t);
subplot(3,3,3);
```

```
plot(t,y,'k');

xlabel('Time');
ylabel('Amplitude');
title('Square wave');
```

%Program for sawtooth wave

```
t=0:0.1:10;
y=sawtooth(t);
subplot(3,3,4);
plot(t,y,'k');
xlabel('Time');
ylabel('Amplitude');
title('Sawtoothwave');
```

%Program for Triangular wave

```
t=0:.0001:20;
y=sawtooth(t,.5); %sawtoothwith50%dutycycle (triangular)
subplot(3,3,5);
plot(t,y);
ylabel ('Amplitude');
xlabel ('TimeIndex');
title('Triangular waveform');
```

%Program for Sinc Pulse

```
t=-10:.01:10;
y=sinc(t);
axis([-10 10 -2 2]);
subplot(3,3,6)
plot(t,y)
ylabel ('Amplitude');
xlabel ('TimeIndex');
title('Sinc Pulse');
```

%Program for Exponential Growing signal

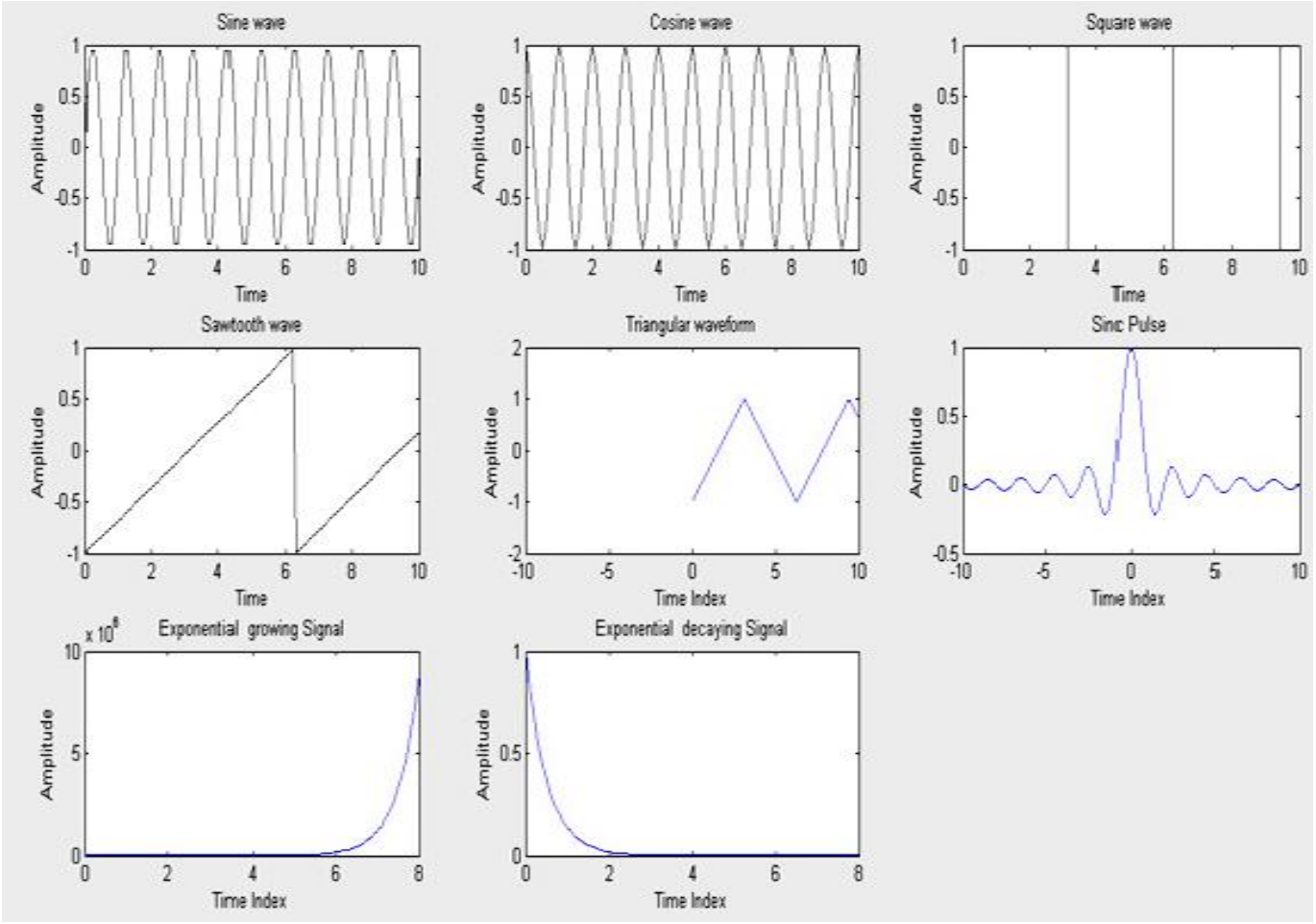
```
t=0:.1:8;
a=2;
y=exp(a*t);
subplot(3,3,7);
plot(t,y);
ylabel ('Amplitude');
xlabel('TimeIndex');
title('Exponential growingSignal');
```

%Program for Exponential Growing signal

```
t=0:.1:8;
a=2;
y=exp(-a*t);
subplot(3,3,8);
plot(t,y);
```

```
ylabel ('Amplitude');
xlabel('TimeIndex');
title('Exponential decaying Signal');
```

OUTPUT: (Generation of Continuous Time Signals)



RESULT:

Thus the MATLAB programs for functional sequence of a signal (Sine, Cosine, triangular, Square, Sawtooth and sinc) using MATLAB function written and the results were plotted.

GENERATION OF DISCRETE TIME SIGNALS

AIM:

To generate a discrete time signal sequence (Unit step, Unit ramp, Sine, Cosine, Exponential, Unit impulse) using MATLAB function.

APPARATUS REQUIRED:

HARDWARE: Personal Computer

SOFTWARE: MATLABR2018a

PROCEDURE:

1. Start the MATLAB program.
2. Open new M-file
3. Type the program
4. Save in current directory
5. Compile and Run the program
6. If any error occurs in the program correct the error and run it again
7. For the output see command window\ Figure window
8. Stop the program.

PROGRAM: (Generation of Discrete Time Signals)

%Program for unit step sequence

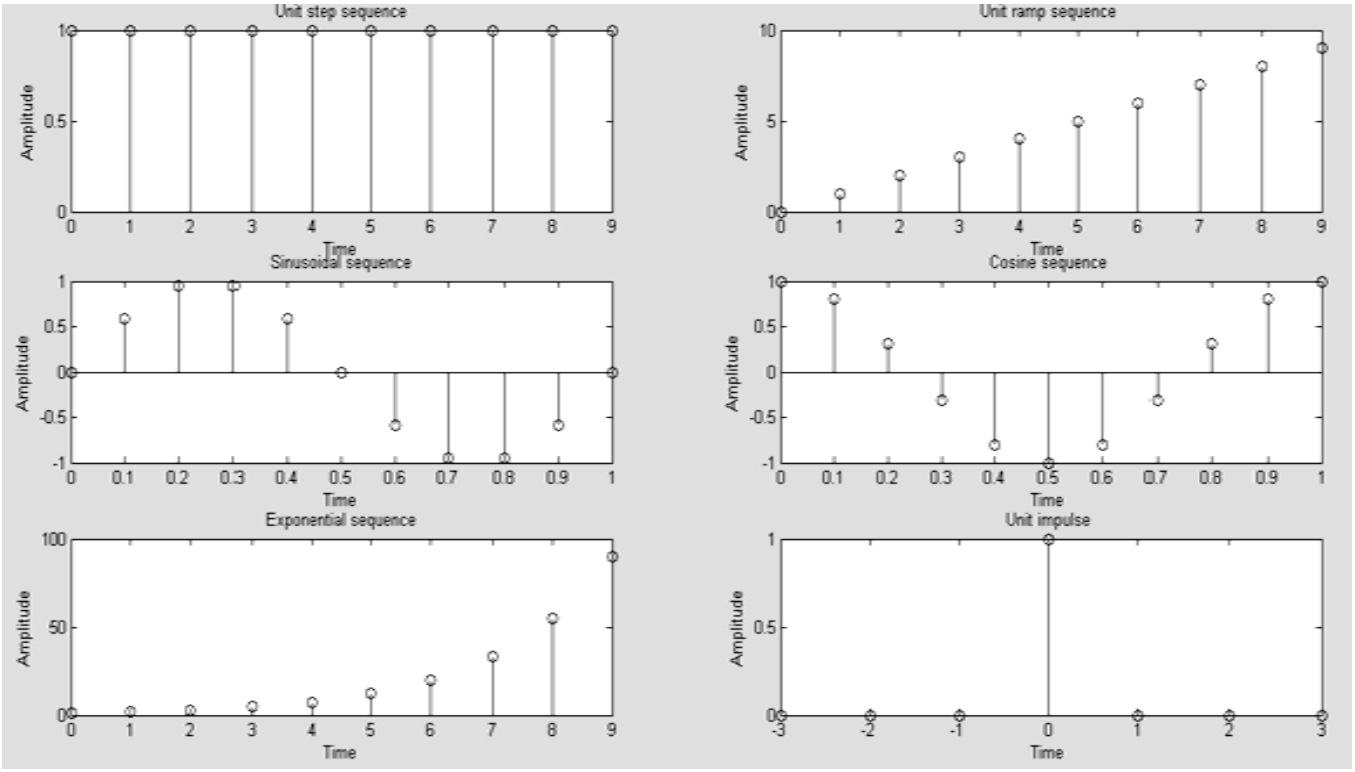
```
clc;
N=input('Enter the length of unit step sequence(N)=');
n=0:1:N-1;
y=ones(1,N);
subplot(3,2,1);
stem(n,y,'k');
xlabel('Time')
ylabel('Amplitude')
title('Unit step sequence');
```

%Program for unit ramp sequence

```
N1=input('Enter the length of unit ramp sequence(N1)=');
n1=0:1:N1-1;
y1=n1;
subplot(3,2,2);
stem(n1,y1,'k');
xlabel('Time');
```

```
ylabel('Amplitude');
title('Unit ramp sequence');
%Program for sinusoidal sequence
N2=input('Enter the length of sinusoidal sequence(N2)= ');
n2=0:0.1:N2-1;
y2=sin(2*pi*n2);
subplot(3,2,3);
stem(n2,y2,'k');
xlabel('Time');
ylabel('Amplitude');
title('Sinusoidal sequence');

%Program for cosine sequence
N3=input('Enter the length of the cosine sequence(N3)=');
n3=0:0.1:N3-1;
y3=cos(2*pi*n3);
subplot(3,2,4);
stem(n3,y3,'k');
xlabel('Time');
 ylabel('Amplitude');
title('Cosine sequence');

%Program for exponential sequence
N4=input('Enter the length of the exponential sequence(N4)=');
n4=0:1:N4-1;
a=input('Enter the value of the exponential sequence(a)= ');
y4=exp(a*n4);
subplot(3,2,5);
stem(n4,y4,'k');
xlabel('Time');
ylabel('Amplitude');
title('Exponential sequence');
```

OUTPUT: (Generation of Discrete Time Signals)



RESULT:

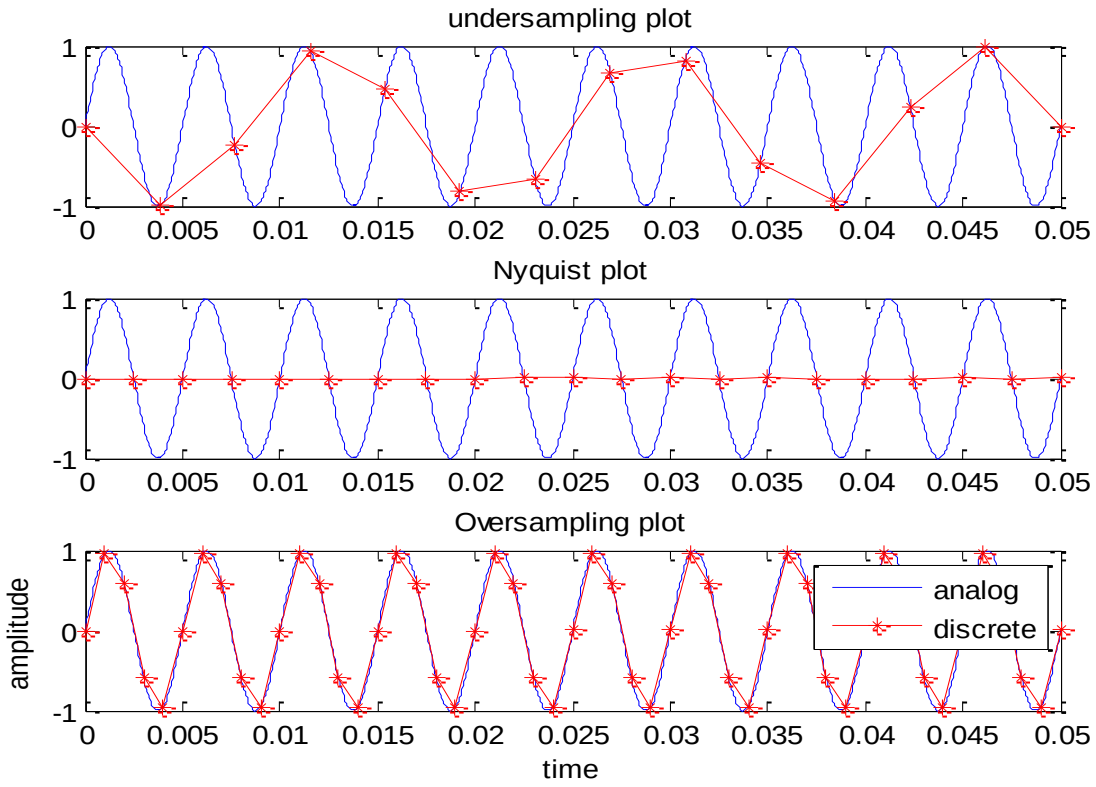
Thus the MATLAB programs for discrete time signal sequence (Unit step, Unit ramp, Sine, Cosine, Exponential, Unit impulse) using MATLAB function written and the results were plotted.

PROGRAM 1

VERIFICATION OF SAMPLING THEOREM BOTH IN TIME AND FREQUENCY DOMAINS

```
tfinal=0.05;
t=0:0.00005:tfinal;     % define time vector 0.05/.00005=1000 sample points
fd=input('Enter desired analog frequency');  %200 Hz

%Generate  analog signal
xt=sin(2*pi*fd*t);
%simulate condition for under sampling i.e., fs1<2*fd
fs1=1.3*fd;
%define the time vector for under sampling
n1=0:1/fs1:tfinal;  % define time vector .05/.00384=13 sample points
xn=sin(2*pi*n1*fd);     %Signal reconstruction
%plot the analog & reconstructed signals
subplot(3,1,1);
plot(t,xt,'b',n1,xn,'r*-');
title('under sampling plot');

%condition for Nyquist sampling fs2=2*fd
fs2=2*fd;
n2=0:1/fs2:tfinal;          %define the time vector 0.05/0.0025=20 sample points
xn=sin(2*pi*fd*n2);    %Signal reconstruction
% plot the analog & reconstructed signals
subplot(3,1,2);
plot(t,xt,'b',n2,xn,'r*-');
title('Nyquist plot');

% condition for oversampling
fs3=5*fd;
n3=0:1/fs3:tfinal ;          %define the time vector 05/.001=50 sample points
xn=sin(2*pi*fd*n3);     %Signal reconstruction
%plot the analog & reconstructed signals
subplot(3,1,3);
plot(t,xt,'b',n3,xn,'r*-');
title('Oversampling plot');
xlabel('time');
ylabel('amplitude');
legend('analog','discrete')
```

Plot:


undersampling plot
Nyquist plot
Oversampling plot
amplitude
time
analog
discrete
1
0
-1
0
0.005
0.01
0.015
0.02
0.025
0.03
0.035
0.04
0.045
0.05

b) Sampling theorem in frequency domain

Scope: To verify the sampling theorem in frequency domain

Sampling in time domain results in periodic components in frequency domain centered at nfs, where n=.... -3,-2,-1, 1,2 3,.....sampling in frequency domain can also be used to analyze aliasing effect

Note :that samples will repeat after 13/fs, hence its sufficient if we consider first 14 samples (0 to 13) of analog %signal (observe the figures given previously)

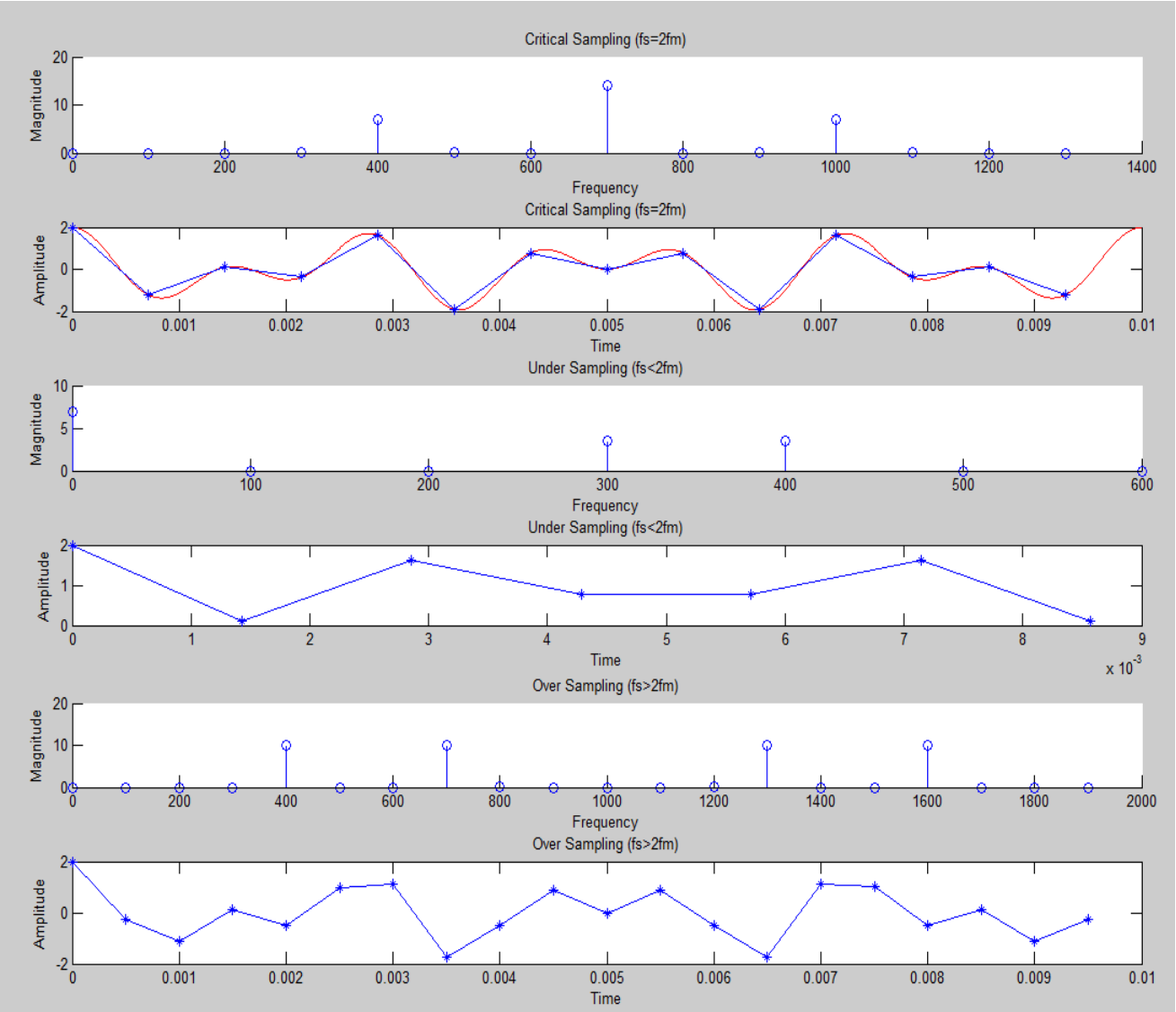
calculate 14 point DFT of sampled signal. abs(...) is required because plot function displays only real values

% Program:

```
tfinal=0.01;
t=0:0.00001: tfinal;
xanalog =cos (2*pi*400*t) +cos(2*pi*700*t); %Original signal

% critical sampling fs=2fm;  where (fm=700)
fs=1400;
tsamp=0:1/fs:13/fs;
xsampled=cos(2*pi*400*tsamp)+cos(2*pi*700*tsamp);
xsampled_DFT=abs(fft(xsampled));
xsampled_length=0:length(xsampled_DFT)-1;
subplot(6,1,1);
stem(100*xsampled_length,xsampled_DFT);
xlabel('Frequency');
ylabel('Magnitude');
title('Critical Sampling (fs=2fm)');

xreconstructed=ifft(fft(xsampled));
subplot(6,1,2);
plot(t,xanalog,'r',tsamp,xreconstructed,'b*-');
xlabel('Time');
ylabel('Amplitude');
title('Critical Sampling (fs=2fm)');

%Under Sampling(fs<2fm)
fs=700;

%Note: that samples will repeat after 6/fs, hence it's sufficient if we consider first 7 samples (0 to 6) of analog signal

tsamp=0:1/fs:6/fs;
xsampled=cos(2*pi*400*tsamp)+cos(2*pi*700*tsamp);
```

```
%calculate 7 point DFT of sampled signal.
xsampled_DFT=abs(fft(xsampled));
xsampled_length=0:length(xsampled_DFT)-1;
subplot(6,1,3);
stem(100*xsampled_length,xsampled_DFT)
xlabel('Frequency');
ylabel('Magnitude');
title('Under Sampling (fs<2fm)');

xreconstructed=ifft(fft(xsampled));
subplot(6,1,4);
plot(tsamp,xreconstructed,'b*-');
xlabel('Time');
ylabel('Amplitude');
title('Under Sampling (fs<2fm)');

%over Sampling(fs>2fm)
fs=2000;

% Note :that samples will repeat after 19/fs,hence its sufficient if we consider first 20 samples(0 to 19) of analog %signal

tsamp=0:1/fs:19/fs;
xsampled=cos(2*pi*400*tsamp)+cos(2*pi*700*tsamp); %calculate 20 point DFT of sampled signal.
xsampled_DFT=abs(fft(xsampled));
xsampled_length=0:length(xsampled_DFT)-1;
subplot(6,1,5);
stem(100*xsampled_length,xsampled_DFT)
xlabel('Frequency');
ylabel('Magnitude');
title('Over Sampling (fs>2fm)');


xreconstructed=ifft(fft(xsampled));
subplot(6,1,6);
plot(tsamp,xreconstructed,'b*-');
xlabel('Time');
ylabel('Amplitude');
title('Over Sampling (fs>2fm)');

%To reconstruct the signal, ifft(...) can be  used.
```

Plot:


Critical Sampling (fs=2fm)
Magnitude
Frequency
Critical Sampling (fs=2fm)
Amplitude
Time
Under Sampling (fs<2fm)
Magnitude
Frequency
Under Sampling (fs<2fm)
Amplitude
Time
x 10^-3
Over Sampling (fs>2fm)
Magnitude
Frequency
Over Sampling (fs>2fm)
Amplitude
Time

PROGRAM 2

EVALUATION OF IMPULSE RESPONSE OF A GIVEN SYSTEM

Aim: To find the impulse response h(n) of the given LTI system whose response y(n) to an input x(n) is known.

```
clc;
clear all;
close all;
b =[1 2 3];       % input('Enter the coefficients of  x: ');
a =[1 2];         % input('Enter the coefficients of y: ');
N=5;              % No of input samples
% Impulse Input
xi=[1,zeros(1,N-1)];       % Impulse with length 50
yimpulse = filter(b,a,xi); % Impulse Response
n=0:length(xi)-1;
disp('Impulse response ');
disp(yimpulse)
stem(n,yimpulse)
```

O/P : Impulse response

1 0 3 -6 12

OR

Experiment 2 : Impulse Response of a Given Second-Order System

```
clear all;
close all;
clc;
% Accept Input and Output signal Co-efficients:
b = input('Enter the coefficients of x(n) in 1-D Matrix Form: ');
a = input('Enter the coefficients of y(n) in 1-D Matrix Form: ');
N = input('Enter the number of samples of impulse response desired: ');

% Calculate Impulse Response using IMPZ function:
[h,t] = impz(b,a,N);
%Plot and Display impulse response co-efficients:
stem(t,h);
title('Impulse Response Plot');
ylabel('h(n)');
xlabel('n');
disp('Impulse Response Coefficients:');
```

```
disp(h);
```

Note: % [H,T] = IMPZ(B,A,N) computes N samples of the impulse response, using
% coefficients B and A from difference equation representation.

OUTPUT:

Example: Find the impulse response of the given difference equation y(n)+ 1/2 y(n-1)=2x(n)

Enter the coefficients of x(n) in 1-D Matrix Form: [2]

Enter the coefficients of y(n) in 1-D Matrix Form: [1 1/2]

Enter the number of samples of impulse response desired: 5

Impulse Response Coefficients:

2.0000

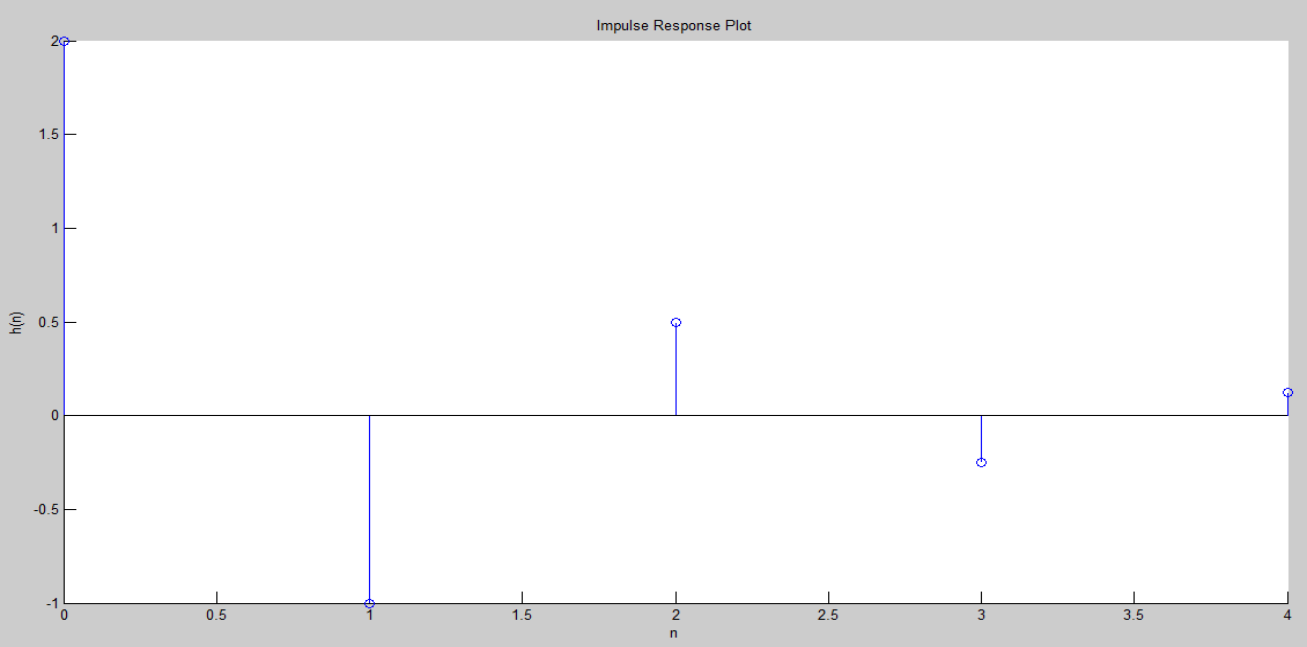
-1.0000

0.5000

-0.2500

0.1250

Plot:



PROGRAM 03

To Perform Linear Convolution of Given Sequences

Aim: To find the linear convolution of given two sequences using inbuilt conv(..) function

Algorithm:

1. Get two signals $x(n)$and $h(n)$in matrix form
2. The convolved signal is denoted as $y(n)$
3. $y(n)$ is given by the formula

$$y(n) = \sum_{k=-\infty}^{\infty} [x(k)\, h(n-k)] \text{ where } n=0 \text{ to } m+p-1$$

4.Stop

MATLAB CODE:

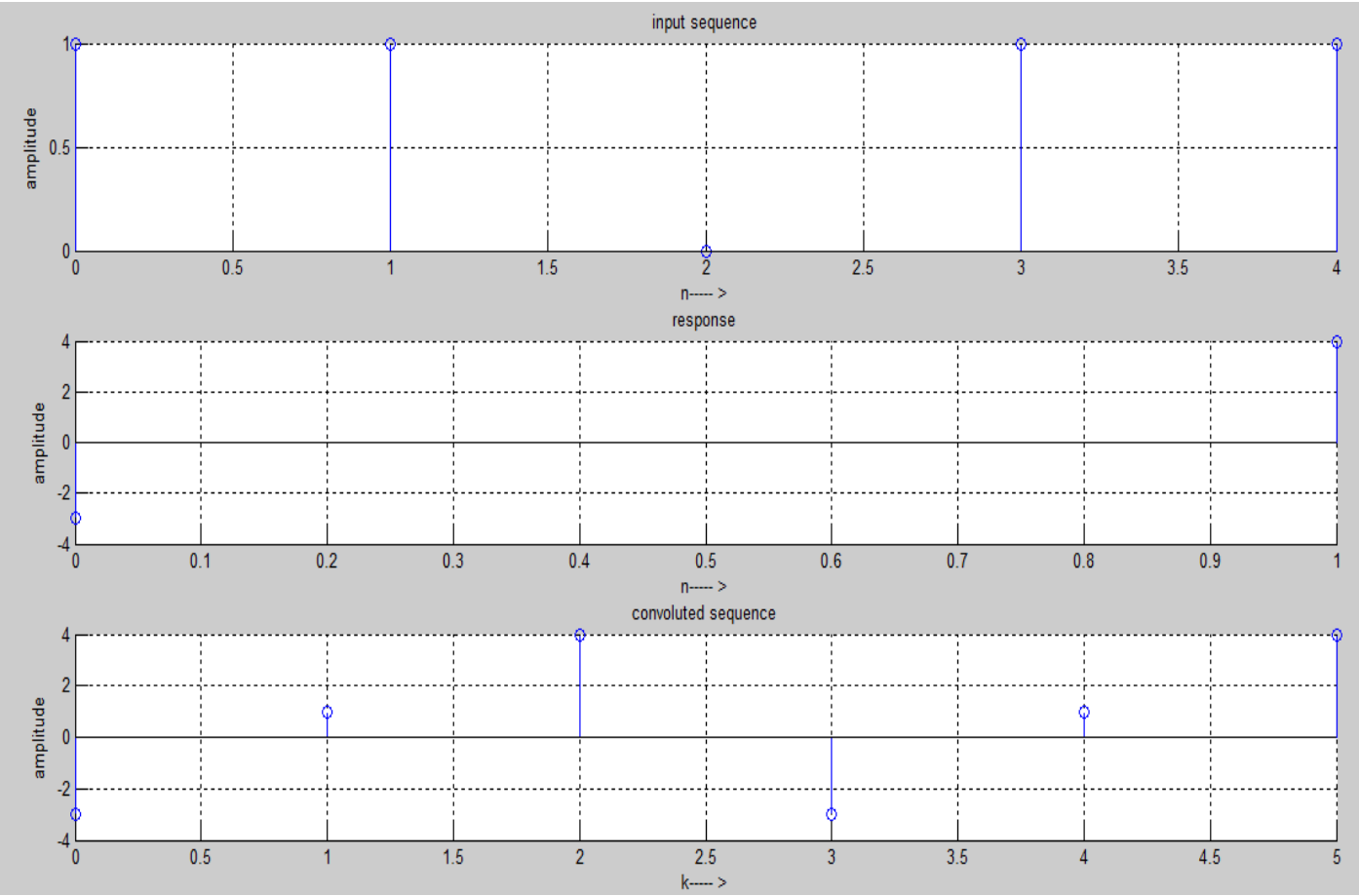
```
clc;
clear all;
close all;
xn=input ('Enter the input sequence, x(n):');
hn=input ('Enter the sequence of response, h(n):');
M=length(xn)
N=length(hn)
len_out=M+N-1
l=0:1:M-1
k=0:1:N-1
Yk=conv(xn, hn)
P=length(Yk)          %Length of convoluted sequence
S=0:1:P-1
disp('Yk=')
disp(Yk);
subplot (3, 1, 1);
stem(l,xn);
Grid on;
xlabel ('n----- >');
ylabel ('amplitude');
TITLE ('input sequence');
subplot (3, 1, 2);
stem(k,hn);
Grid on;
xlabel ('n----- >');
ylabel ('amplitude');
TITLE('response');
subplot (3, 1, 3);
stem(S, Yk);
Grid on;
xlabel('k----- >');
```

```
ylabel('amplitude');
TITLE ('convoluted sequence');
```

OUTPUT:

```
Enter the input sequence, x(n):[1 1 0 1 1]
Enter the sequence of response, h(n):[-3 4 ]
M =
     5
N =
     2
len_out = 6

l =
     0     1     2     3     4
k =
     0     1
Yk =
    -3     1     4    -3     1     4
P =
     6
S =
     0     1     2     3     4     5
Yk=
    -3     1     4    -3     1     4
```

Plot::



PROGRAM 04

CIRCULAR CONVOLUTION OF TWO GIVEN SEQUENCES

Aim: To perform Circular convolution of two given sequences using
a) Convolution summation formula
b) Matrix method
c) Linear convolution from circular convolution with zero padding.

Algorithm:

1. Input the two sequences as x and h.
2. Circularly convolve both to get output y.
3. Plot the sequences.

MATLab program:

a) Convolution summation formula

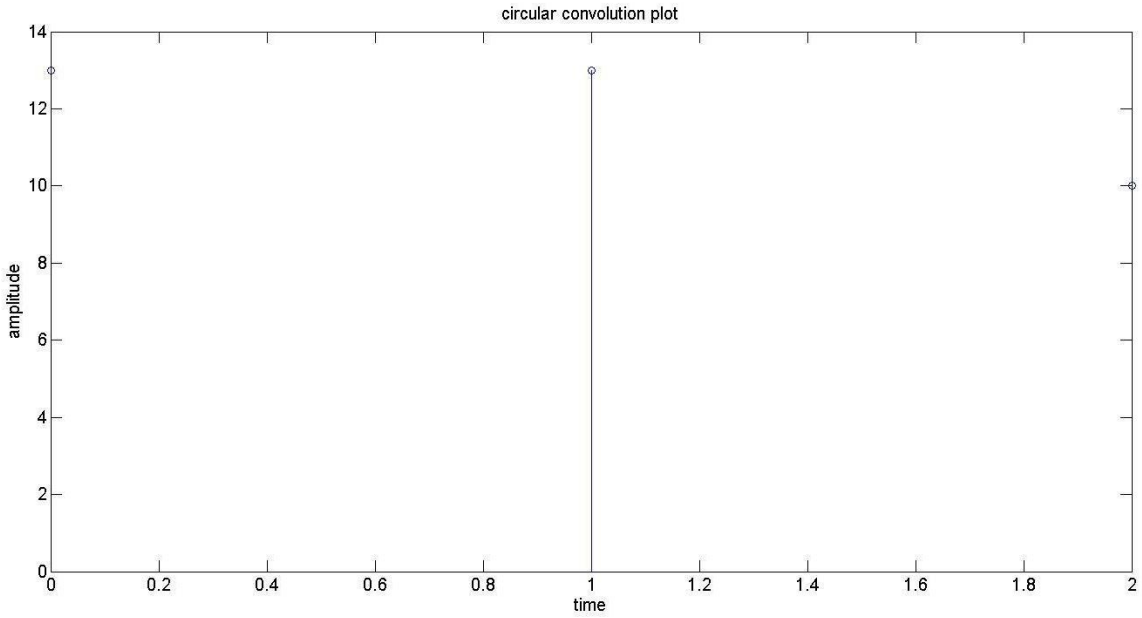
```
%circular convolution%
    clc;
    close all
    clear all
    x=input('enter the 1st sequence');
    h=input('enter the 2nd sequence ');
    N1=length(x);
    N2=length(h);
  N=max(N1,N2);

     for n=0:1:N-1
         y(n+1)=0;
         for M=0:1:N-1
                 i=n-M;
                 if(i<0)
                         i=i+N;
                 end
                 y(n+1)=y(n+1)+x(M+1)*h(i+1);
         end
    end

    disp(y)
    n=0:1:N-1;
    stem(n,y)
    title('circular convolution plot');
    xlabel('time');
    ylabel('amplitude');
```

Result:

enter the 1st sequence [1 2 3]
enter the 2nd sequence [1 2 3]

13 13 10

Plot:



b)Matrix Method

1st method

%matrix convolution method%

```
clc;
clear all;
close all;
x=input('enter the 1st sequence');
h=input('enter the 2nd sequence');
y=h*x;
disp('the circular conv o/p is');
disp(y);
subplot(3,1,1)
n=0:1:length(x)-1;
stem(n,x);
title('input of x');
xlabel('time');

ylabel('amplitude');
subplot(3,1,2)
n1=0:1:length(h)-1;

stem(n1,h);
title('input of h');
xlabel('time');
ylabel('amplitude');
subplot(3,1,3)
n2=0:1:length(y)-1;
stem(n2,y);
```

```
title('output of y');
xlabel('time');
ylabel('amplitude');
```

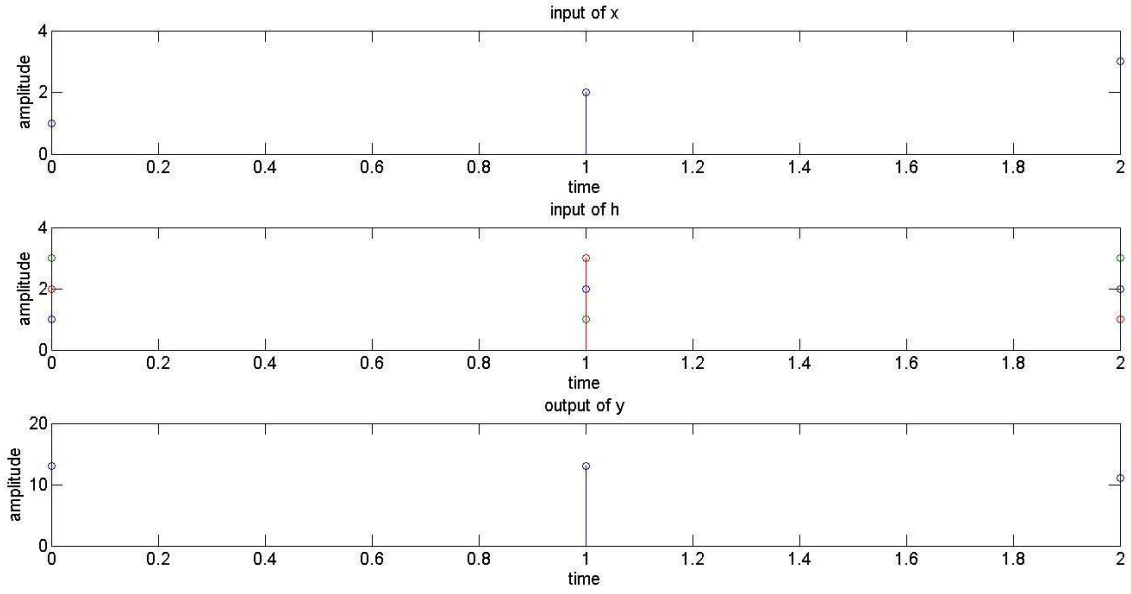
Result:

enter the 1st sequence [1;2;3]

enter the 2nd sequence[1 3 2;2 1 3;2 3 1]

the circular convolution o/p is 13
13
11

Plot:



2nd method

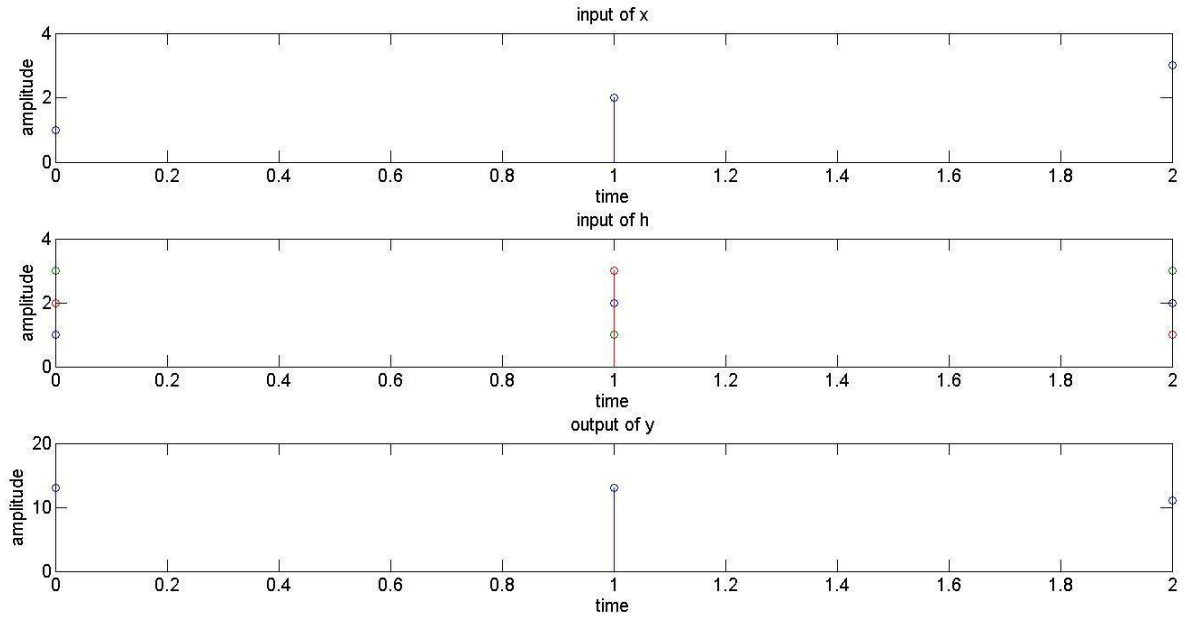
```
%circular matrix mutlipication%
clc;
clear all;
close all;
x=input('enter the 1st sequence');
h=input('enter the 2nd sequence');
a=h'
A=circshift(a,1);
B=circshift(a,2);
C=[a A B]
y=C*x';
disp('the circular conv o/p is');
disp(y);
subplot(3,1,1)
n=0:1:length(x)-1;
stem(n,x);
```

```
title('input of x');
xlabel('time');
ylabel('amplitude');
subplot(3,1,2)
n1=0:1:length(h)-1;
stem(n1,h);
title('input of h');
xlabel('time');
ylabel('amplitude');
subplot(3,1,3)
n2=0:1:length(y)-1;
stem(n2,y);
title('input of y');
xlabel('time');
ylabel('amplitude');
```

Result:

```
enter the 1st sequence [1 2 3]
enter the 2nd sequence [1 2 3]

a =
    1
    2
    3

  C =
      1    3   2
      2    1   3
      3    2   1


the circular convolution o/p is
   13
   13
   10
```

Plot:



c) linear convolution from circular convolution with zero padding.

%linear convolution with zero padding%

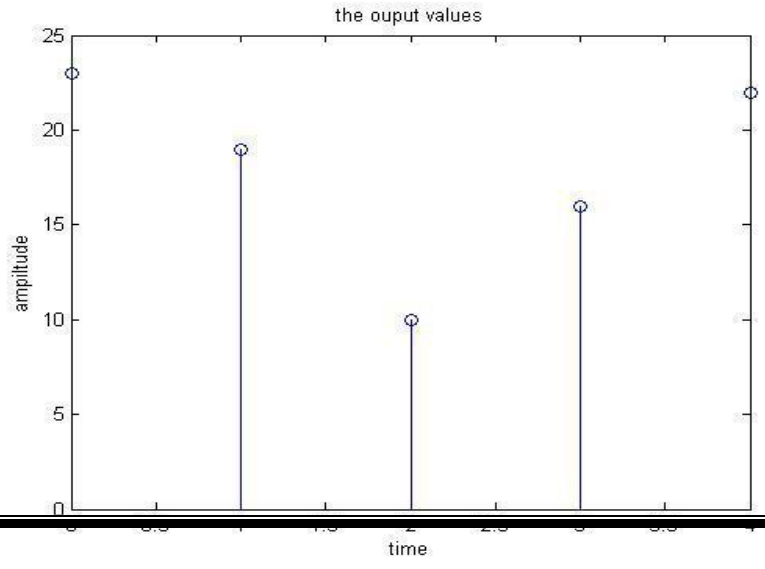
```
clc;
clear all;
close all;
x=input('enter the 1 sequence);
h=input('enter the 2 sequence);
N1=length(x);
N2=length(h);
N=max(N1,N2);
N3=N1-N2;
if (N3>0)
        h=[h,zeros(1,N3)]
else
        x=[x,zeros(1,abs(N3))]
end
for n=0:1:N-1
        y(n+1)=0;
        for M=0:1:N-1

                i=n-M;
                if(i<0)
                i=i+N;
                end
                y(n+1)=y(n+1)+x(M+1)*h(i+1);
        end
end
disp(y)
n=0:1:N-1;
stem(n,y)
title('the ouput values');
xlabel('time');
ylabel('ampiltude');
```

Result:

enter the 1 sequence [1 2 3]
enter the 2 sequence [1 2 3 4 5]

y =23 19 10 16 22

Plot:



ERE

PROGRAM 05

N-Point DFT

Aim: Computation of N-point DFT & to plot the magnitude & phase spectrum.

Algorithm:

1)Enter the value of N in N-point DFT analysis
2)Enter the input sequence
3)Calculate the DFT using the equation of DFT
4)Calculate & plot the phase and magnitude

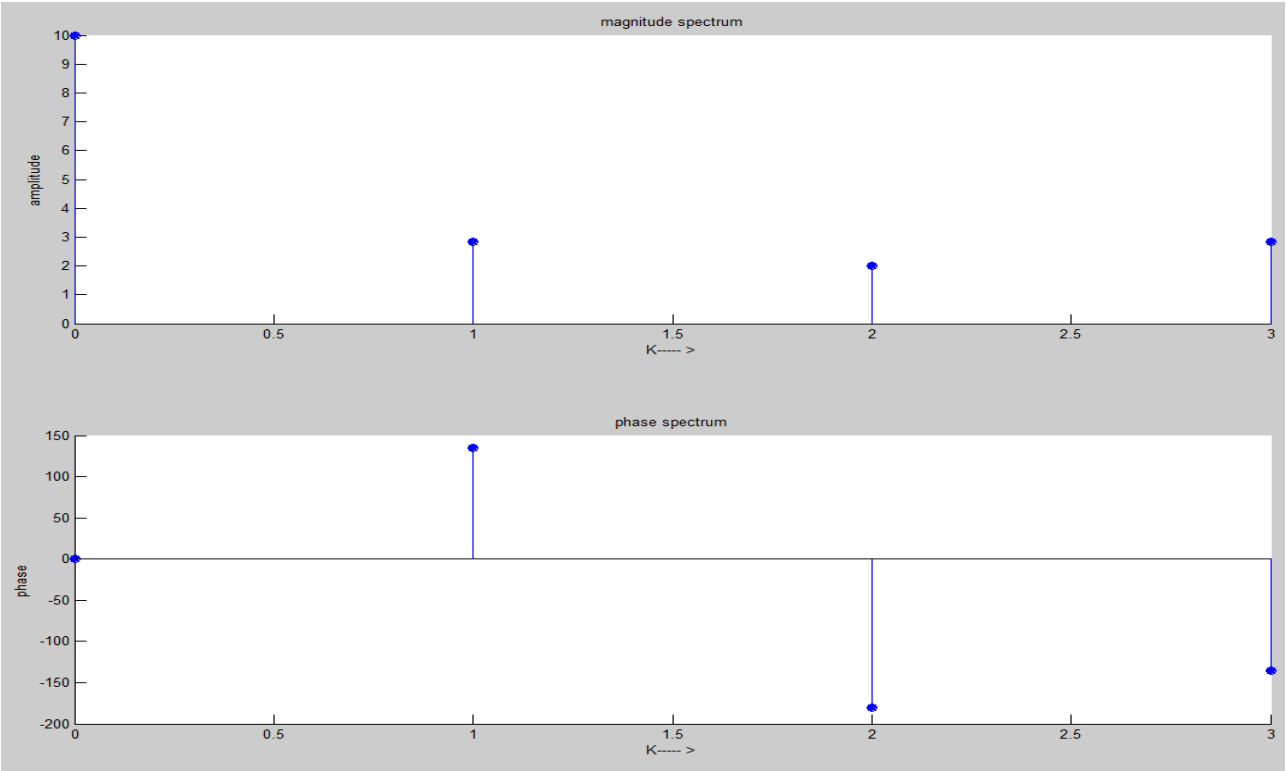
MATLAB CODE:

```
clc;
clear all;
close all;
%DFT computation
x=input ('enter the input sequence');
N=input ('enter the value of  N point DFT');
n=[0:1:N-1]
k=[0:1:N-1]
WN=exp(-j*2*pi/N)
nk=n'*k
WNnk=WN.^nk
Xk=x*WNnk
disp('The DFT of the sequence is=')
disp(Xk);
MagX=abs(Xk);
PhaseX=angle(Xk)*180/pi
disp(Xk);
Subplot(2, 1, 1);
stem (k,MagX,'filled' );
title ('magnitude spectrum');
xlabel('k----- >');
ylabel('amplitude');
Subplot(2, 1, 2);
stem(k, PhaseX, 'filled' );
title ('phase spectrum');
xlabel('k----- >');
ylabel('phase');
```

OUTPUT:

```
enter the input sequence      [1 2 3 4]
enter the value of  N point DFT   4

n =

    0     1     2     3

k =

    0     1     2     3


WN =

  0.0000 - 1.0000i

nk =

    0     0     0     0
    0     1     2     3
    0     2     4     6
    0     3     6     9


WNnk =

  1.0000            1.0000                  1.0000                  1.0000
  1.0000            0.0000 - 1.0000i       -1.0000 - 0.0000i        -0.0000 + 1.0000i
  1.0000           -1.0000 - 0.0000i        1.0000 + 0.0000i       -1.0000 - 0.0000i
  1.0000           -0.0000 + 1.0000i       -1.0000 - 0.0000i        0.0000 - 1.0000i


Xk =

  10.0000           -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 - 2.0000i

The DFT of the sequence is=
  10.0000           -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 - 2.0000i


PhaseX =

         0       135.0000       -180.0000       -135.0000

  10.0000           -2.0000 + 2.0000i  -2.0000 - 0.0000i  -2.0000 - 2.000
```

Plot :


magnitude spectrum
amplitude
K----- >
phase spectrum
phase
K----- >

PROGRAM 6

Linear and Circular convolution using DFT& IDFT

Aim: To compute linear and circular convolution by DFT and IDFT method.

Algorithm:

1) **Enter the sequence x(n) and h(n).**
2) **Compute the DFT of both the sequences using the command fft.**
3) **Multiply both the DFT's.**
4) **Compute the IDFT of the previous result using the command ifft**
5) **Plot x(n), h(n),y(n).**

MATLAB CODE:

a. Linear convolution using DFT and IDFT method

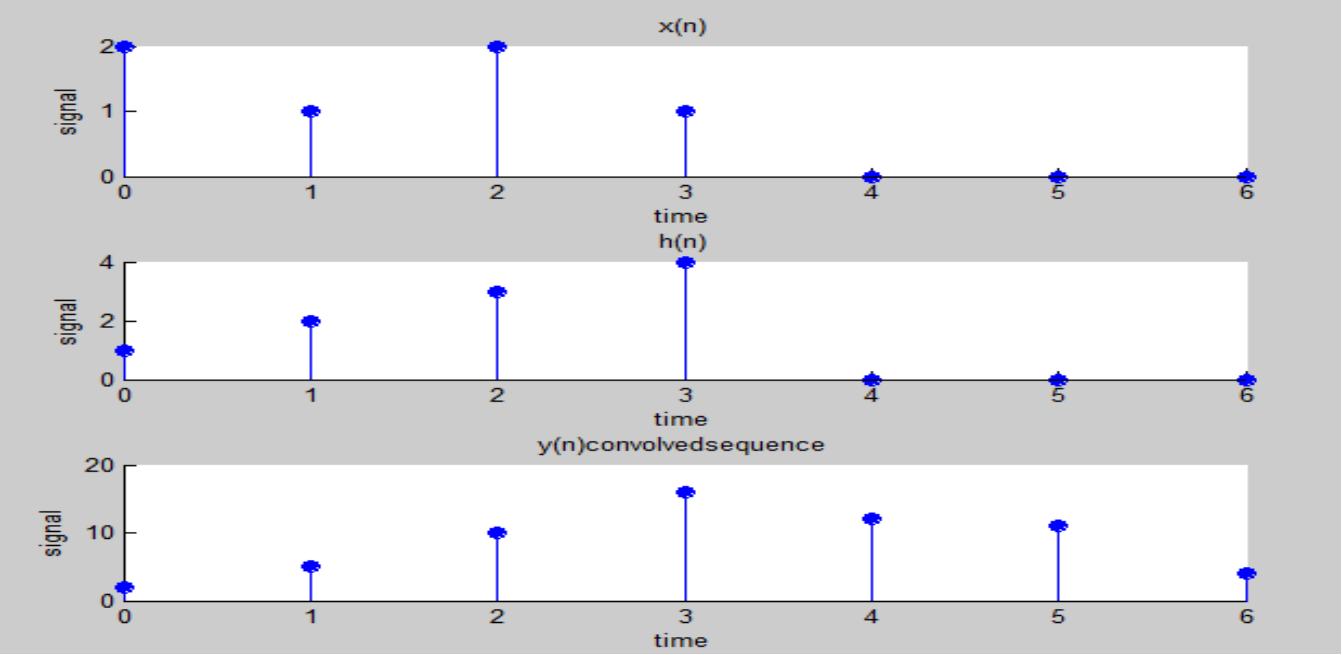
```
clc;
clear all;
x=input('Enter the sequence x(n)');
h=input('Enter the sequence h(n)');
n1=length(x);
n2=length(h);
N=n1+n2-1;
 if (N>n2)
   h=[h zeros(1,N-n2)]
end
if (N>n1)
   x=[x zeros(1,N-n1)]
end
% x1=[x zeros(1,N-n1)];
% x2=[h zeros(1,N-n2)];
X=fft(x,N)
H=fft(h,N)
Y=X.*H
y=ifft(Y,N)
N1=0:1:length(x)-1;
N2=0:1:length(h)-1;
disp('x(n)')
disp(x);
disp('h(n)');
```

```
disp(h);
disp('convolved sequence');
disp(y);
n=0:N-1
subplot(3,1,1);
stem(N1,x,'filled' );
title('x(n)');
ylabel('signal');
xlabel('time');
subplot(3,1,2);
stem(N2,h,'filled' );
title('h(n)');
ylabel( 'signal' );
xlabel('time');
subplot(3,1,3);
stem(n,y,'filled' );
title('y(n) convolved sequence');
ylabel( 'signal' );
xlabel('time');
```

OUTPUT

```
Enter the sequence x(n) [2 1 2 1]
Enter the sequence h(n) [1 2 3 4]

h =

    1    2    3    4    0    0    0


x =

    2    1    2    1    0    0    0


X =

  Columns 1 through 6

   6.0000            1.2775 - 3.1656i   0.5990 + 0.6747i   2.1235 + 0.1549i   2.1235 - 0.1549i   0.5990 - 0.6747i

  Column 7

   1.2775 + 3.1656i
```

```
H =

  Columns 1 through 6

  10.0000          -2.0245 - 6.2240i   0.3460 + 2.4791i   0.1784 - 2.4220i   0.1784 + 2.4220i   0.3460 - 2.4791i

  Column 7

  -2.0245 + 6.2240i


Y =

  Columns 1 through 6

  60.0000          -22.2887 - 1.5424i  -1.4653 + 1.7185i   0.7540 - 5.1154i   0.7540 + 5.1154i  -1.4653 - 1.7185i

  Column 7

 -22.2887 + 1.5424i


y =

    2.0000    5.0000   10.0000   16.0000   12.0000   11.0000    4.0000

x(n)
     2     1     2     1     0     0     0

h(n)
     1     2     3     4     0     0     0

convolved sequence
    2.0000    5.0000   10.0000   16.0000   12.0000   11.0000    4.0000


n =

     0     1     2     3     4     5
```



b.Circular convolution using DFT and IDFT method

```
clc;
clear all;
x=input('Enter the sequence x(n)');
h=input('Enter the sequence h(n)');
n1=length(x)
n2=length(h)
N=max(n1,n2)
if N>n1
   x1=[x zeros(1,N-n1)]
end
   if N>n2
      x2=[h zeros(1,N-n2)]
end
a=fft(x,N)
b=fft(h,N)
c=a.*b
d=ifft(c,N)
disp('x(n)');
disp(x);
disp('h(n)');
```

```
disp(h);
disp('y(n)');
disp(d);
n=0:N-1;
N1=0:1:length(x)-1
N2=0:1:length(h)-1
subplot(3,1,1);
stem(N1,x,'filled');
title('x(n)');
ylabel('signal');
xlabel('time');
subplot(3,1,2);
stem(N2,h,'filled');
title('h(n)');
ylabel('signal');
xlabel('time');
 subplot(3,1,3);
stem(n,d,'filled');
title('y(n)convolved sequence');
ylabel('signal');
xlabel('time');
```

OUTPUT

Ex 1: Enter the sequence x(n) [1 2 3 4]
Enter the sequence h(n) [4 3 2 1]

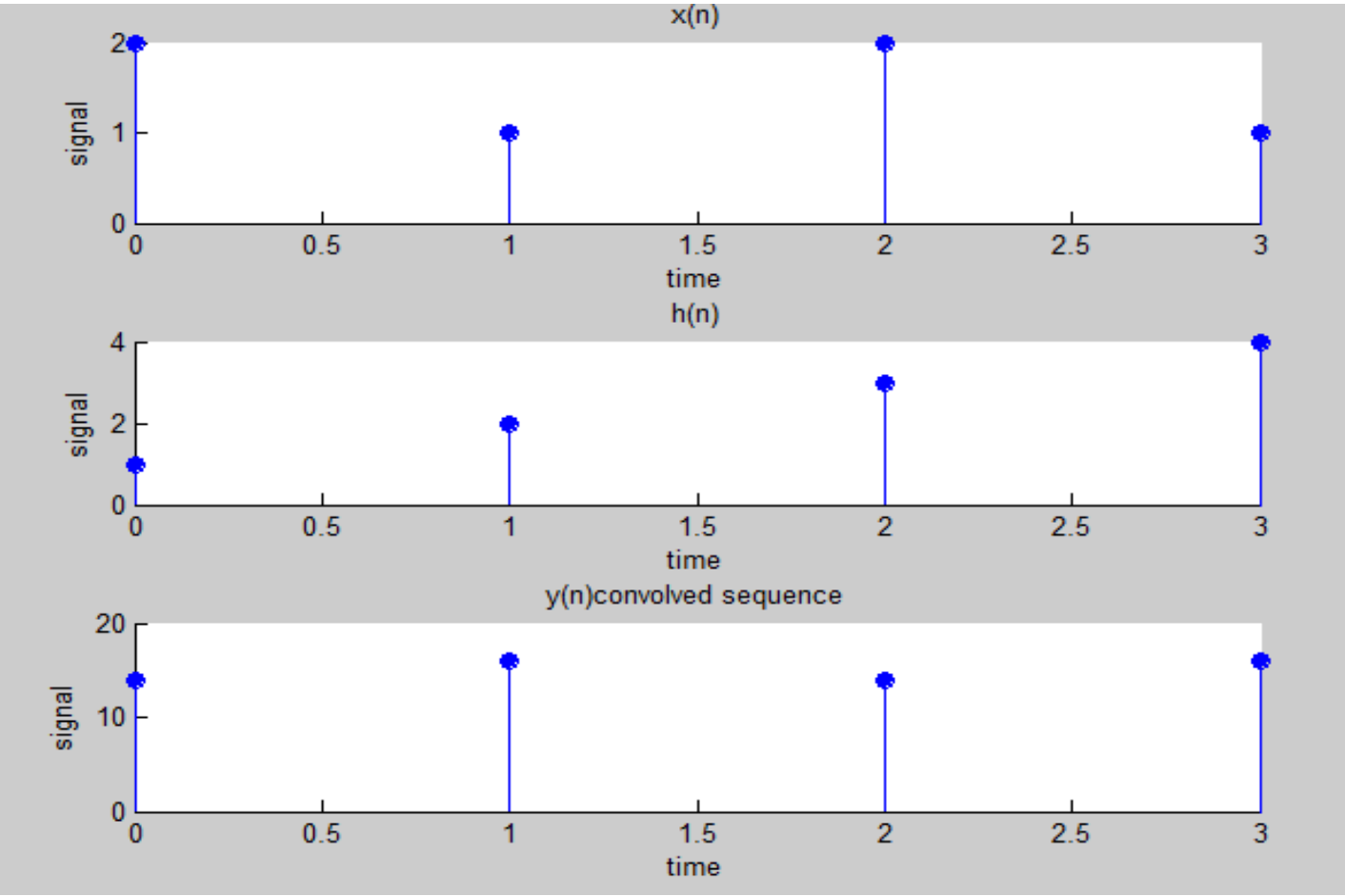
```
x(n)    1   2   3   4
h(n)    4   3   2   1

y(n)    24   22   24 30
```

Ex 2:
Enter the sequence x(n)[2 1 2 1]
Enter the sequence h(n)[1 2 3 4]

```
n1 =
    4
n2 =
```

```
     4
N =
     4
a =
     6     0     2     0
b =
  10.0000            -2.0000 + 2.0000i  -2.0000
-2.0000 - 2.0000i

c =
    60     0    -4     0

d =
    14    16    14    16
```

Plot:



PROGRAM 07

Solution of Given Difference Equation

Aim: To find the solution of a given difference equation

Algorithm:

1) **Enter the coefficients of x and y from the given difference equation**
2) **Use, command filter, filters the input data x using a rational transfer function defined by the numerator & denominator coefficients b & a.**
3) **The command filtic assumes that the input x is 0 in the past.(Only for finding forced response)**
4) **Plot x(n) & y(n).**

A. PROGRAM TO FIND NATURAL RESPONSE FOR GIVEN DIFFERENCE EQUATION

```
clc;
clear all;
close all;
b=input('enter the co-efficient of x(n) in the order x[n],x[n-1]….:');
a=input('enter the co-efficient of y(n) in the order y[n],y[n-1]….:');
i=input('enter the intial conditions in the order y[-1],y[-2]….:');
N=input('enter the number of samples:');
Zi=filtic(b,a,i);
n=0:1:N-1;
x=zeros(1,N);
Yn=filter(b,a,x,Zi);
Subplot(1,1,1);
Stem(n,Yn);
Xlabel('n');
Ylabel('amplitude');
title('natural response');
Grid on;
disp(Yn);
```

B. PROGRAM TO FIND FORCED RESPONSE

```
clc;
clear all;
close all;
b=input ('enter the co-efficient of x (n) in the order x[n], x [n-1]…. :');
a=input ('enter the co-efficient of y (n) in the order y[n], y [n-1]…. :');
x=input('enter the input sequence:');
Yf=filter (b,a,x);
subplot (1, 1, 1);
stem (Yf);
xlabel('n----- >');
ylabel('amplitude');
```

```
Title ('forced response');
Grid on;
disp (Yf);
```

C. PROGRAM TO FIND COMPELET RESPONSE

```
clc;
clear all;
close all;
b=input('enter the co-efficient of x(n) in the order x[n],x[n-1]….:');
a=input('enter the co-efficient of y(n) in the order y[n],y[n-1]….:');
x=input('enter the input sequence');
i=input('enter the intial conditions in the order y[-1],y[-2]….:');
N=input('enter the number of samples:');
n=0:N-1;
Zi=filtic(b,a,i);
Yc=filter(b,a,x,Zi);
stem(n,Yc);
Xlabel('n------- >');
Ylabel('amplitude');
Title('complete response');
Grid on;
```

Problem:

EX:1: Find the solution of given difference equation $y(n) - (1/9)[y(n-1)] = x(n-1)$ with initial conditions $y(-1)=1$, $y(-2)=0$ and $x(n)=u(n)$.

OUTPUT:

Natural Response: 0.1111 0.0123 0.00143 0.0002 0.0000

Forced Response: 0 1.0000 1.114 1.123

Complete Response: 0.111 1.0123 1.1125 1.1236 1.1248

PROGRAM 08

CALCULATION OF DFT & IDFT BY FFT

AIM: Write a MATLAB Script to compute Discrete Fourier Transform and Inverse Discrete Fourier Transform of the given sequence using FFT algorithms (DIT-FFT & DIF-FFT).

ALGORITHM:

1. Input the given sequence x[n]
2. Compute the Discrete Fourier Transform using FFT library function (dit fft or dif fft) and obtain X[k].
3. Compute the Inverse Discrete Fourier Transform using FFT library function (dit fft or dif fft) and obtain X[n] by following steps
   a. Take conjugate of X [k] and obtain X[k]*
   b. Compute the Discrete Fourier Transform using FFT library function (dit fft or dif fft) for X[k]* and obtain N.x[n]*
   c. Once again take conjugate for N.x[n]* and divide by N to obtain x[n]
4. Display the results.

MATLAB Program:

```
%dft and idft using fft%
clc;
clear all;
close all;
x=input('Enter the sequence')
N=length(x);
y=fft(x,N)
mag=abs(y)
phase=angle(y)
phase1=phase*(180/pi);

%verification%
yi=ifft(y)
yd=real(yi)
subplot(2,2,1)
n=0:1:length(x)-1;
stem(n,x)
title('the values ox x');
subplot(2,2,2)

stem(n,mag)
title('magnitude values of y');
subplot(2,2,3)

stem(n,phase1)
title('phase values of y');
subplot(2,2,4)
stem(n,yd)
title('real of ifft')
```

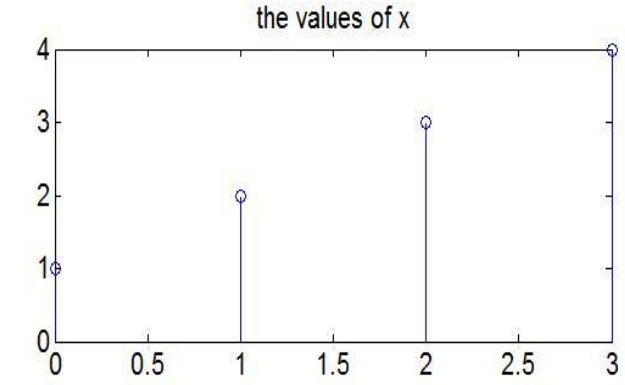
Result: Enter the sequence[1 2 3 4]

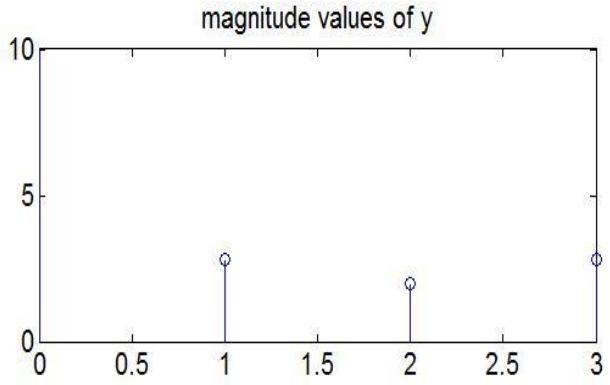
x = 1 2 3 4
y = 10.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i

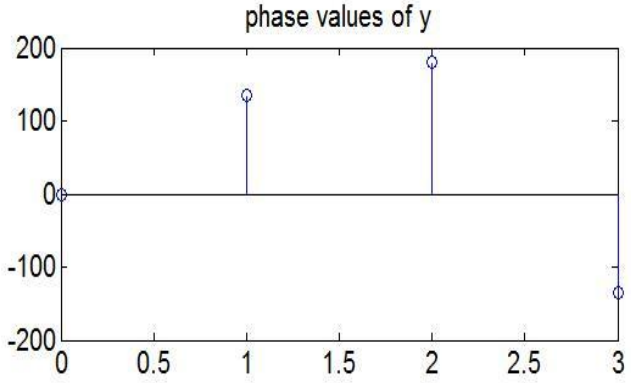
mag = 10.0000 2.8284 2.0000 2.8284
phase = 0 2.3562 3.1416 -2.3562

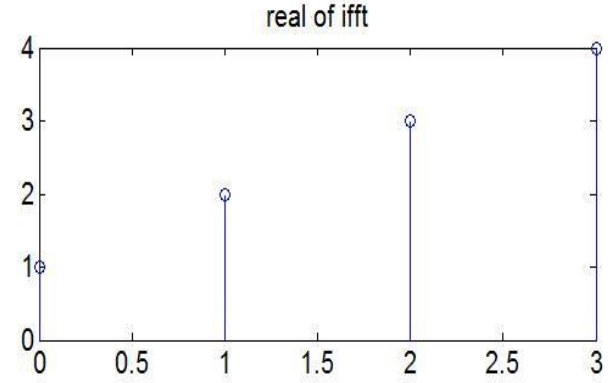
yi = 1 2 3 4
yd = 1 2 3 4

Plot:









PROGRAM 9

DESIGN OF IIR FILTERS

Aim: To design and implement IIR filters to meet the given specifications.

Algorithm:
1) Enter the pass band ripple (Ap in db) and stop band ripple (As in db).
2) Enter the pass band frequency (wp) and stop band frequency (ws).
3) Get the sampling time (T).
4) Calculate pass band edge analog frequency Ωp & stop band edge analog frequency Ωs.
5) **to calculate order of filter Butterworth filter order**
6) Calculate the magnitude of the frequency response in decibels (dB)
   mag=20*log10 (abs (H))
7) Plot the magnitude response [magnitude in dB Vs normalized frequency (Hz]]
8) Calculate the phase response using an = angle (H)
9) Plot the phase response [phase in radians Vs normalized frequency (Hz)].

MATLAB PROGRAM:

A) Design of Butterworth filters(Low Pass, High Pass)

Using Bilinear Transformation:

```
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
Ap=input('enter the passband ripple');
As=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
T=input('enter the sampling time');

[N,wn]=buttord(wp,ws,Ap,As,'s');% Find the order n and cutoff frequency
ch=input('give type of filter 1:LPF,2:HPF,3:BPF,4:BSF');

switch ch
case 1
disp('Frequency response of Butterworth IIR LPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=bilinear(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)
```

```
w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworth LPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
```

case 2

```
disp('Frequency response of Butterworth IIR LPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'high','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=bilinear(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworth LPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
end
```

B) Design of Butterworth filters(Bandpass and Bandstop)

Using Bilinear Transformation:

```
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
Ap=input('enter the passband ripple');
As=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
T=input('enter the sampling time');
```

```
[N,wc]=buttord(wp,ws,Ap,As,'s');% Find the order n and cutoff frequency
wc=[wp ws];
ch=input('give type of filter,1:BPF,2:BSF');
switch ch
```

case 1

```
disp('Frequency response of Butterworth IIR LPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'bandpass','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=bilinear(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh LPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
```

case 2

```
disp('Frequency response of Butterworth IIR HPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'STOP','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=bilinear(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
```

```
title('Magnitude Response of the desired Butterworh HPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
end
```

EXAMPLE 1: Design and implementation of butter worth LPF,HPF,BPF,BSF IIR filter by taking sampling time T=0.1 seconds to meet given specification using bilinear transformation.

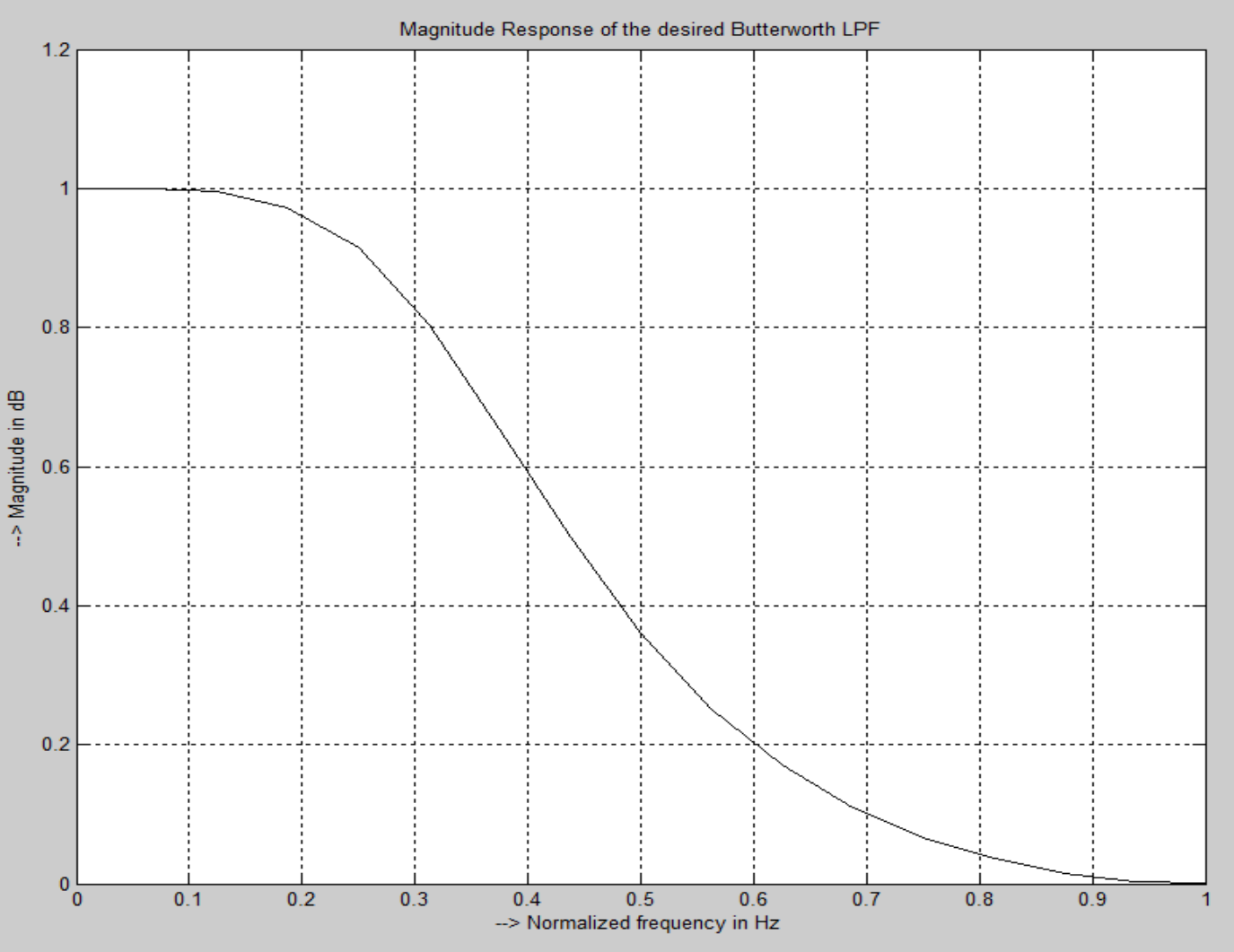
$$0.6 \le \left|H(e^{i\omega})\right| \le 1.0 \quad \text{for } 0 \le \omega \le 0.35\pi$$

$$\left|H(e^{i\omega})\right| \le 0.1 \quad \text{for } 0.7\pi \le \omega \le \pi$$

Output:

LPF

```
enter the IIR filter design specifications
enter the passband ripple 4.437
enter the stopband ripple 20
enter the passband freq 12.25
enter the stopband freq 39.25
enter the sampling time 0.1
give type of filter 1:LPF,2:HPF 1
Frequency response of Butterworth IIR LPF is:
normalized tf is,

Transfer function:
       1
-----------------
s^2 + 1.414 s + 1

unnormalized tf is,

Transfer function:
      154.8
--------------------
s^2 + 17.6 s + 154.8

digital tf is,

Transfer function:
0.1707 z^2 + 0.3415 z + 0.1707
------------------------------
  z^2 - 0.5407 z + 0.2237
```



HPF

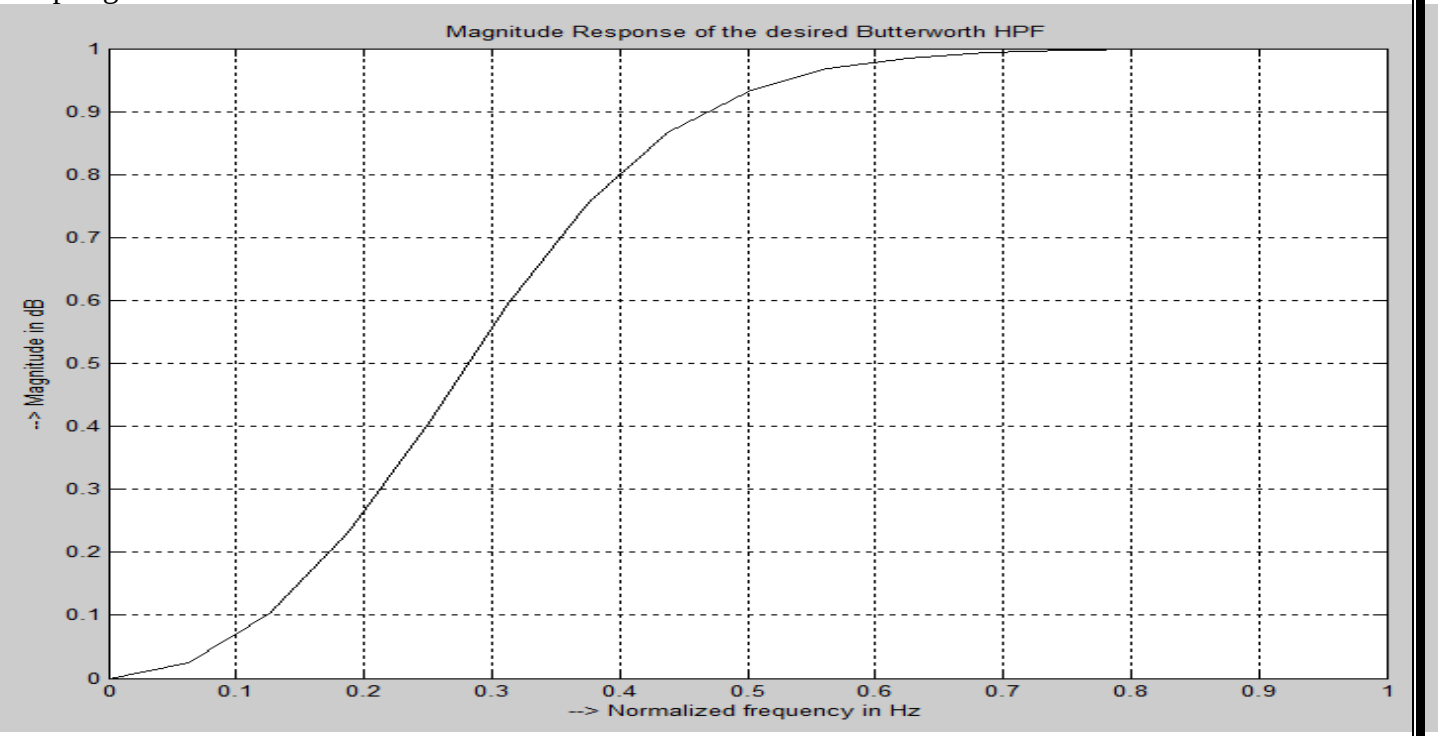
enter the IIR filter design specifications
enter the passband ripple 4.437
enter the stopband ripple 20
enter the passband freq 12.25
enter the stopband freq 39.25
enter the sampling time 0.1
give type of filter 1:LPF,2:HPF **2**
Frequency response of Butterworth IIR HPF is:
normalized tf is,
Transfer function:
1
---
s^2 + 1.414 s + 1

unnormalized tf is,

Transfer function:
s^2
---
s^2 + 17.6 s + 154.8
digital tf is,

Transfer function:
0.4411 z^2 - 0.8822 z + 0.4411
---
z^2 - 0.5407 z + 0.2237

Sampling time: 0.1

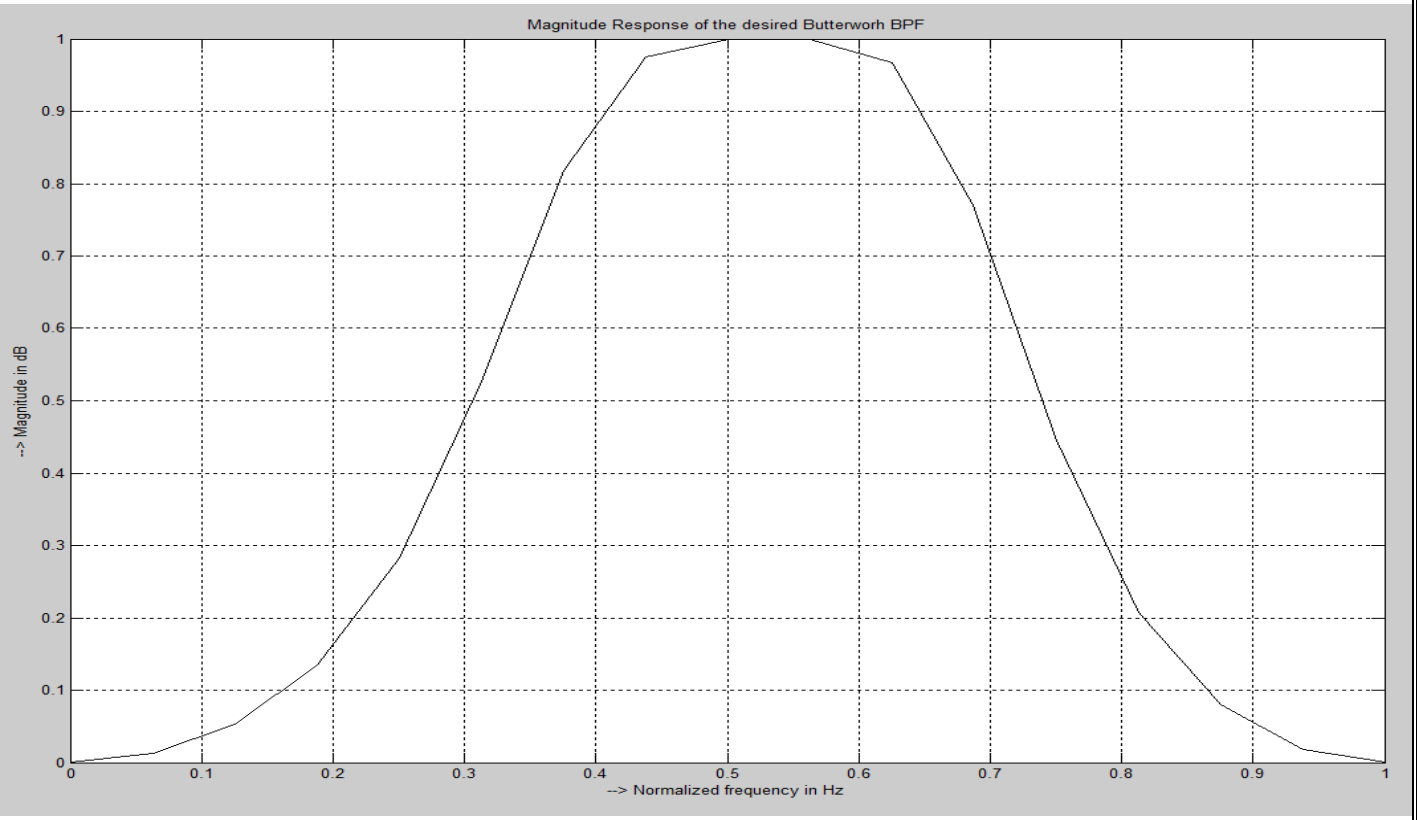


BPF
enter the IIR filter design specifications
enter the passband ripple4.437
enter the stopband ripple20
enter the passband freq12.25
enter the stopband freq39.25
enter the sampling time0.1
give type of filter,1:BPF,2:BSF 1
Frequency response of Butterworth IIR BPF is:
normalized tf is,

Transfer function:
1
---
s^2 + 1.414 s + 1
unnormalized tf is,

Transfer function:
729 s^2
---
s^4 + 38.18 s^3 + 1691 s^2 + 1.836e004 s + 2.312e005
digital tf is,

Transfer function:
0.1676 z^4 + 9.437e-016 z^3 - 0.3352 z^2 + 0.1676
---
z^4 + 0.2346 z^3 + 0.5716 z^2 + 0.09269 z + 0.2269



BSF

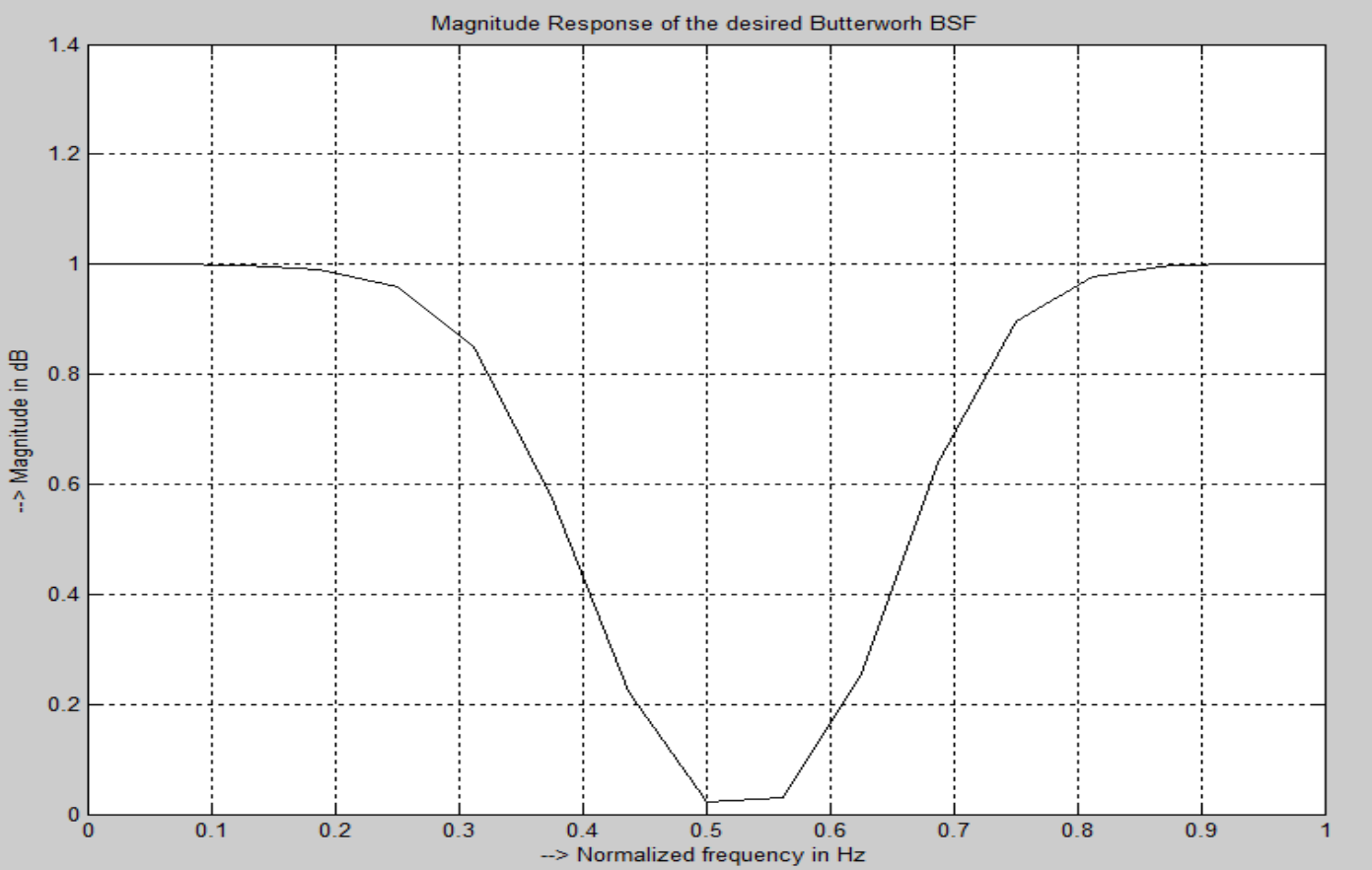
enter the IIR filter design specifications
enter the passband ripple4.437
enter the stopband ripple20
enter the passband freq12.25
enter the stopband freq39.25
enter the sampling time0.1
give type of filter,1:BPF,2:BSF 2
Frequency response of Butterworth IIR BSF is:
normalized tf is,

Transfer function:
1
---
s^2 + 1.414 s + 1

unnormalized tf is,

Transfer function:
s^4 + 961.6 s^2 + 2.312e005
---
s^4 + 38.18 s^3 + 1691 s^2 + 1.836e004 s + 2.312e005

digital tf is,
Transfer function:
0.4459 z^4 + 0.1636 z^3 + 0.9067 z^2 + 0.1636 z + 0.4459
---
z^4 + 0.2346 z^3 + 0.5716 z^2 + 0.09269 z + 0.2269


Magnitude Response of the desired Butterworh BSF
--> Magnitude in dB
--> Normalized frequency in Hz
0
0.2
0.4
0.6
0.8
1
1.2
1.4
0.1
0.3
0.5
0.7
0.9

A) Butterworth digital IIR Low Pass & High Pass using Impulse Invariant transformation

```
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
Ap=input('enter the passband ripple  in db');
As=input('enter the stopband ripple in db');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
T=input('enter the sampling time');

[N,wc]=buttord(wp,ws,Ap,As,'s');  % Find the order n and cutoff frequency
ch=input('give type of filter 1:LPF,2:HPF');
switch ch
case 1
disp('Frequency response of Butterworth IIR LPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=impinvar(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh LPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');

case 2

disp('Frequency response of Butterworth IIR HPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'high','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=impinvar(b,a,1/T);
disp('digital tf is,')
```

```
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh HPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
end
```

RESULTS:

EXAMPLE 1: Design and implementation of butterworth LPF,HPF,BPF,BSF IIR filter by taking sampling time T=0.1 seconds to meet given specification using Impulse Invariant transformation transformation.

$$0.9 \le \left|H(e^{i\omega})\right| \le 1.0 \quad \text{for } 0 \le \omega \le 0.35\pi$$

$$\left|H(e^{i\omega})\right| \le 0.275 \quad \text{for } 0.7\pi \le \omega \le \pi$$

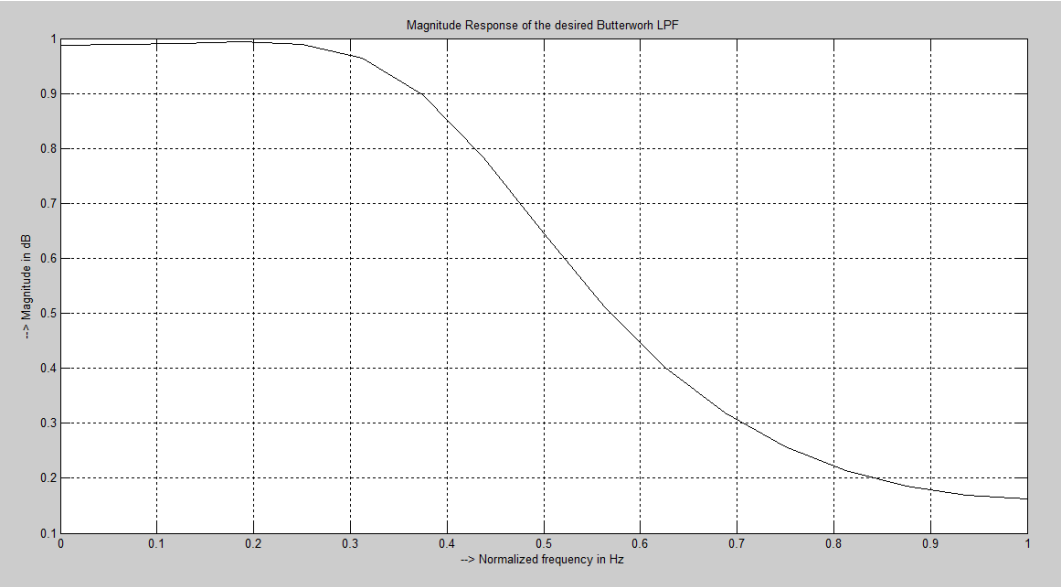
LPF:

```
enter the IIR filter design specifications
enter the passband ripple   0.9151
enter the stopband ripple 11.2133
enter the passband freq  1.0996
enter the stopband freq  2.1991
enter the sampling time  1
give type of filter 1:LPF,2:HPF   1
```

Frequency response of Butterworth IIR LPF is:

```
normalized tf is,

Transfer function:
          1
---------------------
s^3 + 2 s^2 + 2 s + 1

unnormalized tf is,

Transfer function:
             3.042
---------------------------------
s^3 + 2.898 s^2 + 4.199 s + 3.042

digital tf is,

Transfer function:
      0.5074 z^2 + 0.1985 z
-----------------------------------
z^3 - 0.536 z^2 + 0.3055 z - 0.05514
```

Sampling time: 1



HPF:

enter the IIR filter design specifications
enter the passband ripple 0.9151
enter the stopband ripple 11.2133
enter the passband freq 1.0996
enter the stopband freq 2.1991
enter the sampling time 1
give type of filter 1:LPF,2:HPF 2

Frequency response of Butterworth IIR HPF is:

normalized tf is,

Transfer function:

$$\frac{1}{s^3 + 2\,s^2 + 2\,s + 1}$$

unnormalized tf is,

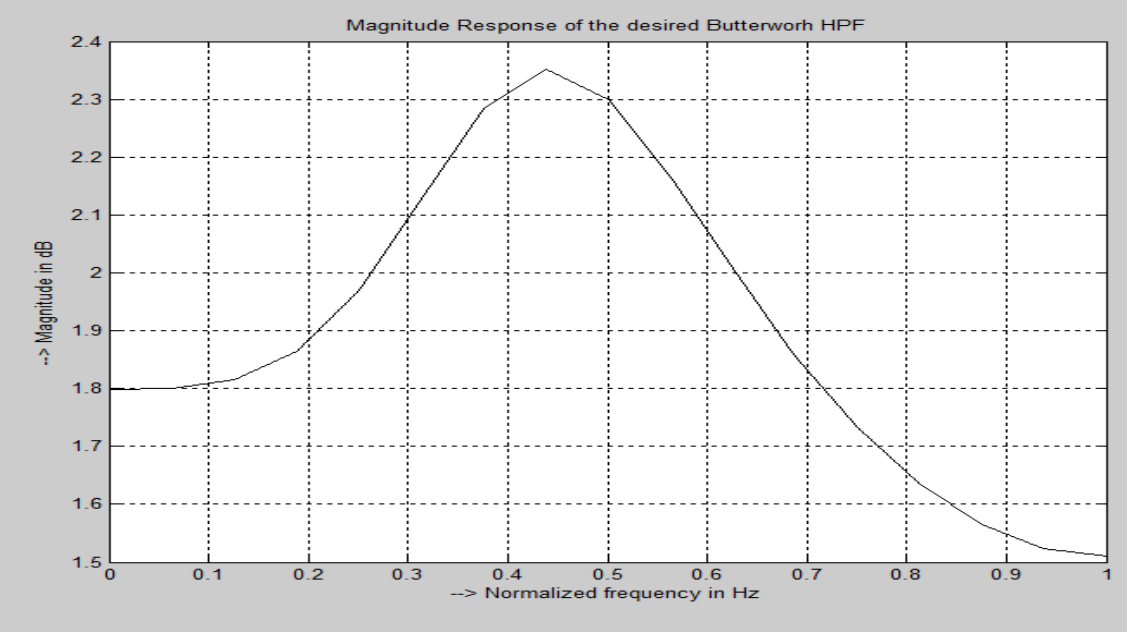
Transfer function:

$$\frac{s^3}{s^3 + 2.898\,s^2 + 4.199\,s + 3.042}$$

digital tf is,

Transfer function:

$$\frac{-1.898\,z^3 + 0.8441\,z^2 - 0.1764\,z - 0.05514}{z^3 - 0.536\,z^2 + 0.3055\,z - 0.05514}$$



B) Butterworth digital IIR BandPass & Bandstop using Impulse Invariant transformation

```
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
Ap=input('enter the passband ripple in db');
As=input('enter the stopband ripple in db ');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
T=input('enter the sampling time');

[N,wc]=buttord(wp,ws,Ap,As,'s');% Find the order n and cutoff frequency
wc=[wp ws];
ch=input('give type of filter 1:BPF,2:BSF');
switch ch
case 1
disp('Frequency response of Butterworth IIR BPF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'bandpass','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=impinvar(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
```

```
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh BPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
```

case 2

```
disp('Frequency response of Butterworth IIR BSF is:');
[bn,an]=butter(N,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=butter(N,wc,'STOP','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=impinvar(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/1000:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh BSF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
end
```

RESULTS:

BPF:

```
enter the IIR filter design specifications
enter the passband ripple 0.9151
enter the stopband ripple11.21
enter the passband freq 1.0996
enter the stopband freq2.1991
enter the sampling time1
give type of filter 1:BPF,2:BSF1
Frequency response of Butterworth IIR BPF is:
normalized tf is,

Transfer function:
          1
----------------------
s^3 + 2 s^2 + 2 s + 1
```

unnormalized tf is,

Transfer function:

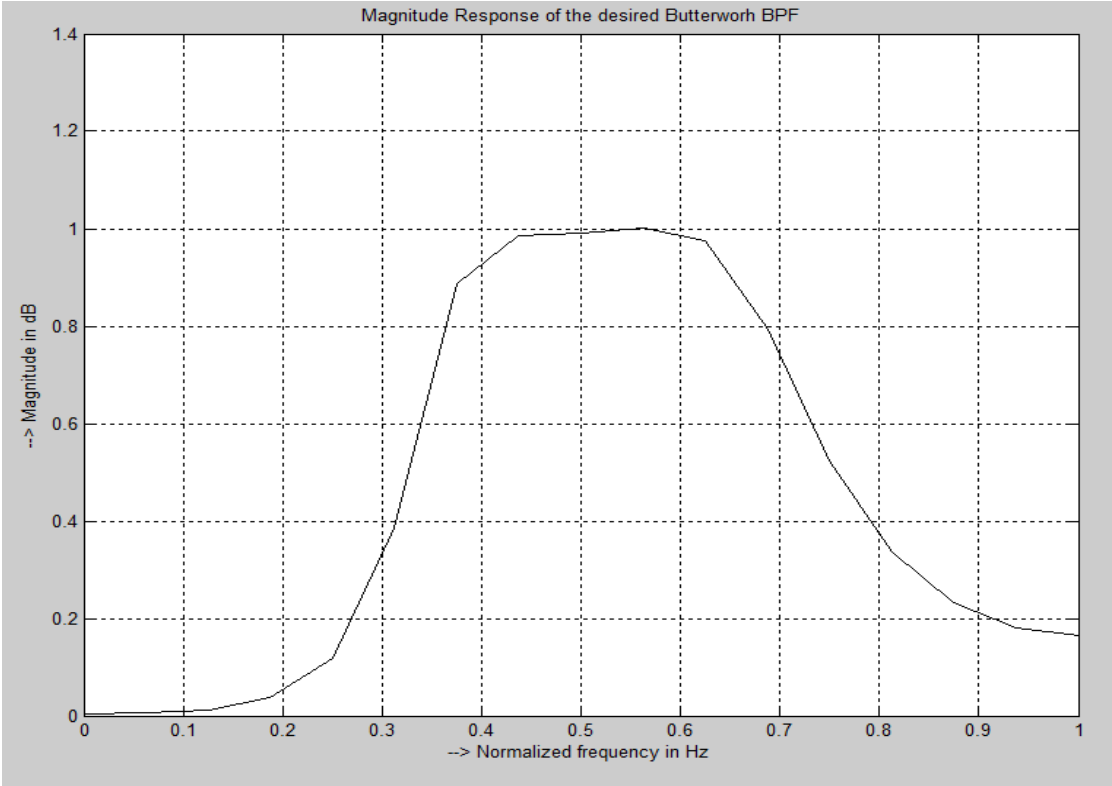
$$\frac{1.329\ s^3}{s^6 + 2.199\ s^5 + 9.672\ s^4 + 11.96\ s^3 + 23.39\ s^2 + 12.86\ s + 14.14}$$

digital tf is,

Transfer function:

$$\frac{8.327e\text{-}016\ z^6 + 0.1162\ z^5 - 0.3288\ z^4 + 0.1869\ z^3 + 0.09082\ z^2 - 0.07799\ z}{z^6 - 0.1549\ z^5 + 1.023\ z^4 + 0.02\ z^3 + 0.546\ z^2 - 0.004934\ z + 0.1109}$$

Sampling time: 1



BSF:

enter the IIR filter design specifications
enter the passband ripple0.9151
enter the stopband ripple11.213
enter the passband freq1.0996
enter the stopband freq2.1991
enter the sampling time1
give type of filter 1:BPF,2:BSF2
Frequency response of Butterworth IIR BSF is:
normalized tf is,

Transfer function:

$$\frac{1}{}$$

---
s^3 + 2 s^2 + 2 s + 1

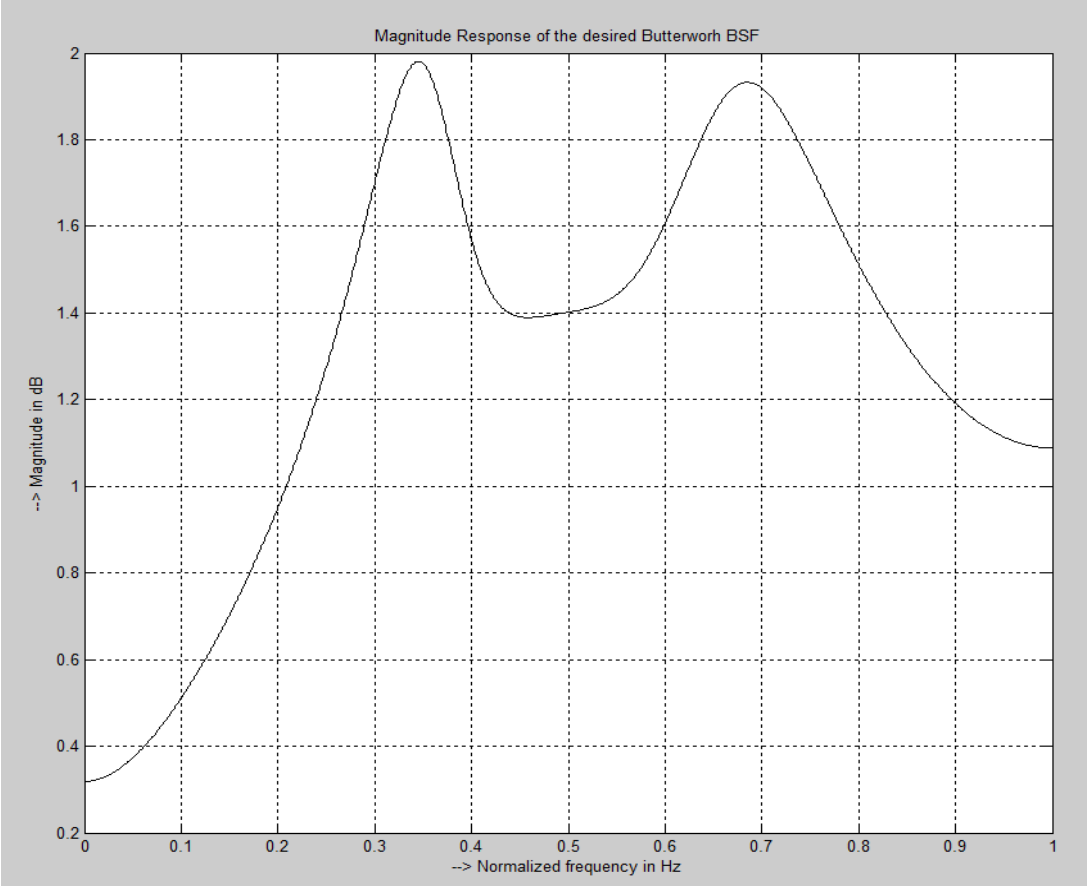
unnormalized tf is,

Transfer function:
s^6 + 7.254 s^4 + 17.54 s^2 + 14.14
---
s^6 + 2.199 s^5 + 9.672 s^4 + 11.96 s^3 + 23.39 s^2 + 12.86 s + 14.14

digital tf is,

Transfer function:
-1.199 z^6 + 0.6311 z^5 - 0.8302 z^4 + 0.3738 z^3 - 0.02131 z^2 + 0.1247 z + 0.1109
---
z^6 - 0.1549 z^5 + 1.023 z^4 + 0.02 z^3 + 0.546 z^2 - 0.004934 z + 0.1109

Sampling time: 1



A) Design of chebyshev-1 filters(Low Pass, High Pass)

Using Bilinear Transformation:

```
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
Ap=input('enter the passband ripple');
As=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
T=input('enter the sampling time');

[N,wc]=cheb1ord(wp,ws,Ap,As,'s');% Find the order n and cutoff frequency
ch=input('give type of filter 1:LPF,2:HPF');
switch ch
case 1
disp('Frequency response of C  IIR LPF is:');
[bn,an]=cheby1(N,Ap,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=cheby1(N,Ap,wc,'S');
disp('unnormalized tf is,')
Hs=tf(b,a)
[num,den]=bilinear(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired chebyshev-1   LPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');

case 2

disp('Frequency response of chebyshev-1   IIR HPF is:');
[bn,an]=cheby1(N,Ap,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=cheby1(N,wc,Ap,'high','S');
disp('unnormalized tf is,')
Hs=tf(b,a)
```

```
[num,den]=bilinear(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired chebyshev-1    HPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
 end
```

RESULTS:

EXAMPLE 1: Design and implementation of chebyshev-1 LPF,HPF IIR filter by taking sampling time T=0.1 seconds to meet given specification using Bilinear Transformation transformation.

$$0.8 \le \left|H(e^{i\omega})\right| \le 1.0 \quad \textbf{for } 0 \le \omega \le 0.2\pi$$

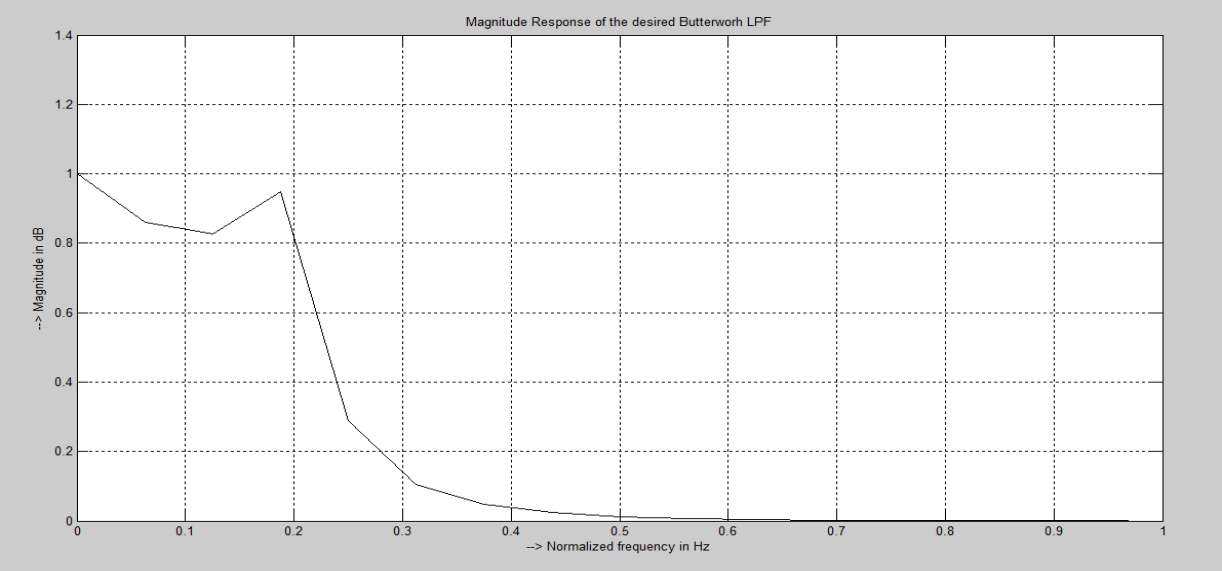
$$\left|H(e^{i\omega})\right| \le 0.2 \quad \textbf{for } 0.32\pi \le \omega \le \pi$$

LPF:

```
enter the IIR filter design specifications
enter the passband ripple1.9
enter the stopband ripple13.97
enter the passband freq0.6498
enter the stopband freq1.0995
enter the sampling time1
give type of filter 1:LPF,2:HPF1

Frequency response of Butterworth IIR LPF is:

normalized tf is,
Transfer function:
          0.3375
-----------------------------------
s^3 + 0.7559 s^2 + 1.036 s + 0.3375

unnormalized tf is,

Transfer function:
           0.09259
-------------------------------------
s^3 + 0.4912 s^2 + 0.4373 s + 0.09259

digital tf is,
```

Transfer function:
0.00847 z^3 + 0.02541 z^2 + 0.02541 z + 0.00847
---
z^3 - 2.27 z^2 + 1.961 z - 0.6236

Sampling time: 1



HPF:

enter the IIR filter design specifications
enter the passband ripple1.9
enter the stopband ripple13.97
enter the passband freq0.6498
enter the stopband freq1.0995
enter the sampling time1
give type of filter 1:LPF,2:HPF2

Frequency response of Butterworth IIR HPF is:
normalized tf is,

Transfer function:
0.3375
---
s^3 + 0.7559 s^2 + 1.036 s + 0.3375

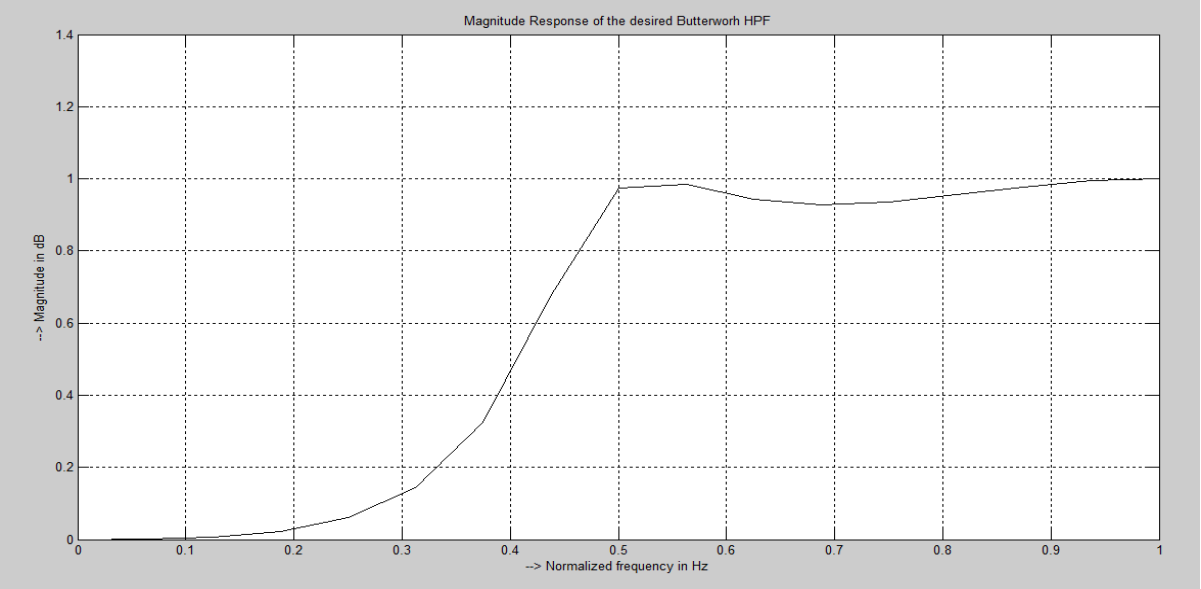
unnormalized tf is,

Transfer function:
s^3
---
s^3 + 4.313 s^2 + 6.678 s + 11.02

digital tf is,

Transfer function:
0.1612 z^3 - 0.4836 z^2 + 0.4836 z - 0.1612
---
z^3 + 0.1042 z^2 + 0.5332 z + 0.1394

Sampling time: 1



B) Design of chebyshev-1 filters(Low Pass, High Pass)

% Using IMPULSE INVARIENT Transformation:

```
clc;
clear all;
close all;
warning off;
disp('enter the IIR filter design specifications');
Ap=input('enter the passband ripple');
As=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
T=input('enter the sampling time');

[N,wc]=cheb1ord(wp,ws,Ap,As,'s');% Find the order n and cutoff frequency
ch=input('give type of filter 1:LPF,2:HPF');
switch ch
case 1
disp('Frequency response of Butterworth IIR LPF is:');
[bn,an]=cheby1(N,Ap,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)
[b,a]=cheby1(N,Ap,wc,'S');
```

```
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=impinvar(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh LPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
```

case 2

```
disp('Frequency response of Butterworth IIR HPF is:');
[bn,an]=cheby1(N,Ap,1,'S');
disp('normalized tf is,')
Hsn=tf(bn,an)

[b,a]=cheby1(N,wc,Ap,'high','S');
disp('unnormalized tf is,')
Hs=tf(b,a)

[num,den]=impinvar(b,a,1/T);
disp('digital tf is,')
Hz=tf(num,den,T)

w=0:pi/16:pi;
disp('freq response is,')
Hw=freqz(num,den,w)
disp('magnitude response is,')
Hw_mag=abs(Hw)
plot(w/pi,Hw_mag,'k');
grid;
title('Magnitude Response of the desired Butterworh HPF')
xlabel('--> Normalized frequency in Hz');
ylabel('--> Magnitude in dB');
end
```

RESULTS:

EXAMPLE 1: Design and implementation of chebyshev-1 LPF,HPF IIR filter by taking sampling time T=0.1 seconds to meet given specification using IMPULSE INVARIENT transformation.

$$0.9 \le \left|H(e^{i\omega})\right| \le 1.0 \quad \textbf{for} \quad 0 \le \omega \le 0.25\pi$$

$$\left|H(e^{i\omega})\right| \le 0.24 \quad \textbf{for} \quad 0.5\pi \le \omega \le \pi$$

LPF:

enter the IIR filter design specifications
enter the passband ripple0.9151
enter the stopband ripple12.395
enter the passband freq0.7854
enter the stopband freq1.5708
enter the sampling time1
give type of filter 1:LPF,2:HPF1

Frequency response of Butterworth IIR LPF is:
normalized tf is,

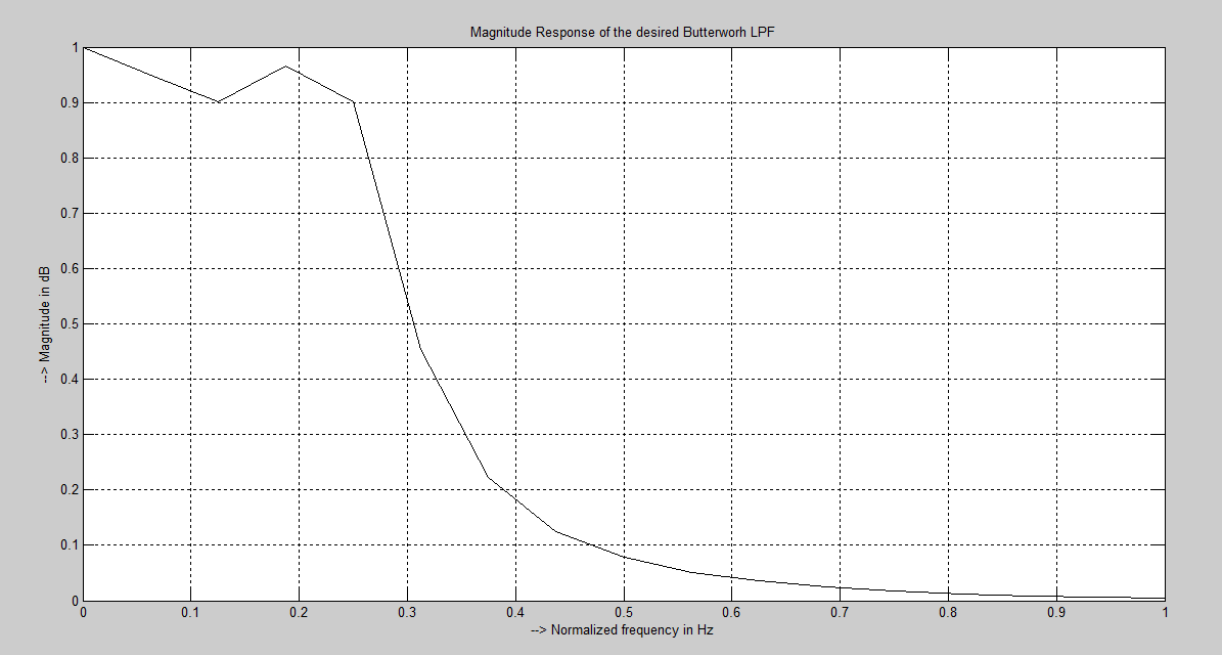
Transfer function:
0.5162
---
s^3 + 1.021 s^2 + 1.272 s + 0.5162

unnormalized tf is,

Transfer function:
0.2501
---
s^3 + 0.8022 s^2 + 0.7844 s + 0.2501

digital tf is,
Transfer function:
0.09112 z^2 + 0.06993 z
---
z^3 - 1.852 z^2 + 1.461 z - 0.4483

Sampling time: 1



HPF:

enter the IIR filter design specifications
enter the passband ripple0.9151
enter the stopband ripple12.39
enter the passband freq0.7854
enter the stopband freq1.5708
enter the sampling time1
give type of filter 1:LPF,2:HPF2

Frequency response of Butterworth IIR HPF is:
normalized tf is,

Transfer function:
0.5162
---
s^3 + 1.021 s^2 + 1.272 s + 0.5162

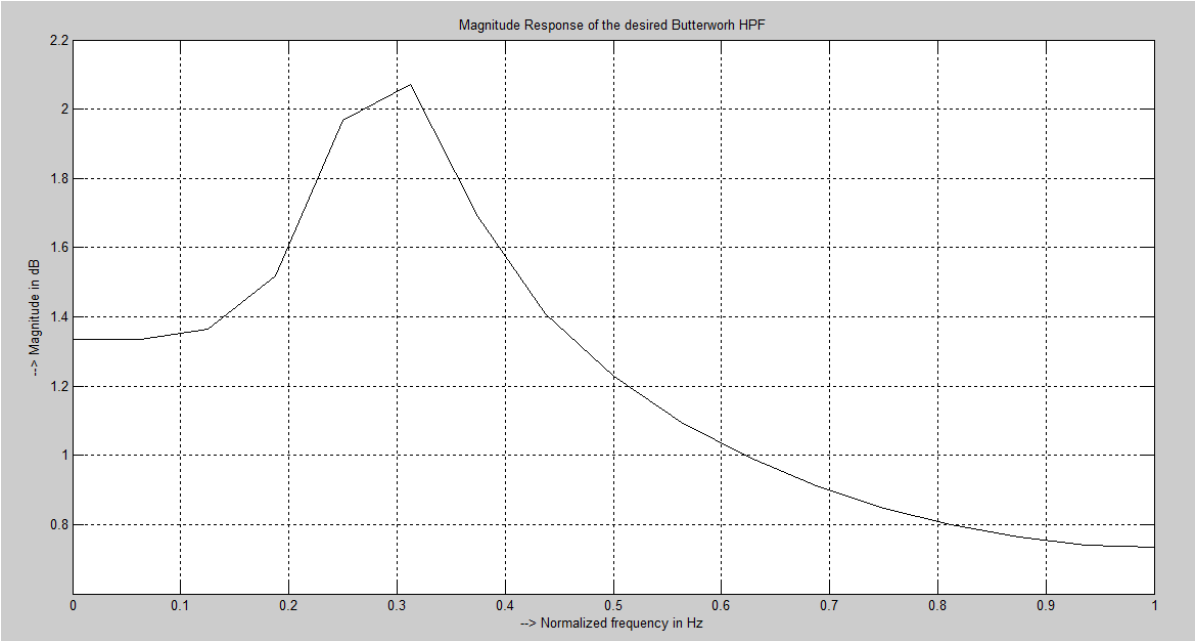
unnormalized tf is,

Transfer function:
s^3
---
s^3 + 2.171 s^2 + 1.609 s + 1.365

digital tf is,

Transfer function:
-1.171 z^3 + 0.9371 z^2 - 0.3046 z - 0.1141
---
z^3 - 1.207 z^2 + 0.81 z - 0.1141

Sampling time: 1



PROGRAM 10

DESIGN AND IMPLEMENTATION OF FIR FILTERS TO MEET GIVEN SPECIFICATIONS (LOW PASS, HIGH PASS, BAND PASS & BAND REJECT FILTERS) USING DIFFERENT WINDOW FUNCTIONS.

Aim: To write the program for design and implementation of FIR filters using different window functions.

MATLAB PROGRAM:

```
%FIR filter using Rectangular Window

clc;
clear all;
wc=0.5*pi;
N=input('enter the  length  of the filter');
alpha=(N-1)/2;
eps=0.001;
n=0:1:N-1;
%LPF
hd=sin(wc*(n-alpha+eps))./(pi*(n-alpha+eps));
wh=RECTWIN(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,1)
plot(w/pi,abs(h),'r')
xlabel ('normalized freq')
ylabel ('magnitude')
title('LP Rectangular')
disp('LP filter co-efficient');
disp(hn);
hold on

%HPF
hd=(sin(pi*(n-alpha+eps))-sin((n-alpha+eps)*wc))./(pi*(n-alpha+eps));
wh=RECTWIN(N);
hn=hd.*wh';
```

```
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,2)
plot(w/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('HP Rectangular')
disp('HP filter co-efficient');
disp(hn)
hold on
```

% BPF

```
Wc1=0.25*pi;
Wc2=0.75*pi;
hd=(sin(Wc2*(n-alpha+eps))-sin(Wc1*(n-alpha+eps))).//((n-alpha+eps)*pi);
wh=RECTWIN(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,3)
plot(W/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BPF Rectangular')
disp('BPF filter co-efficient');
disp(hn)
hold on;
```

% BSF

```
hd=(sin(Wc1*(n-alpha+eps))-sin(Wc2*(n-alpha+eps))+sin(pi*(n-alpha+eps))).//((n-alpha+eps)*pi);
wh=RECTWIN(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,4)
```

```
plot(W/pi,abs(h),'m');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BSF Rectangular')
disp('BSF filter co-efficient');
disp(hn)
hold on;
```

RESULTS:

enter the length of the filter7

LP filter co-efficient

-0.1061 0.0003 0.3186 0.5000 0.3180 -0.0002 -0.1061

HP filter co-efficient

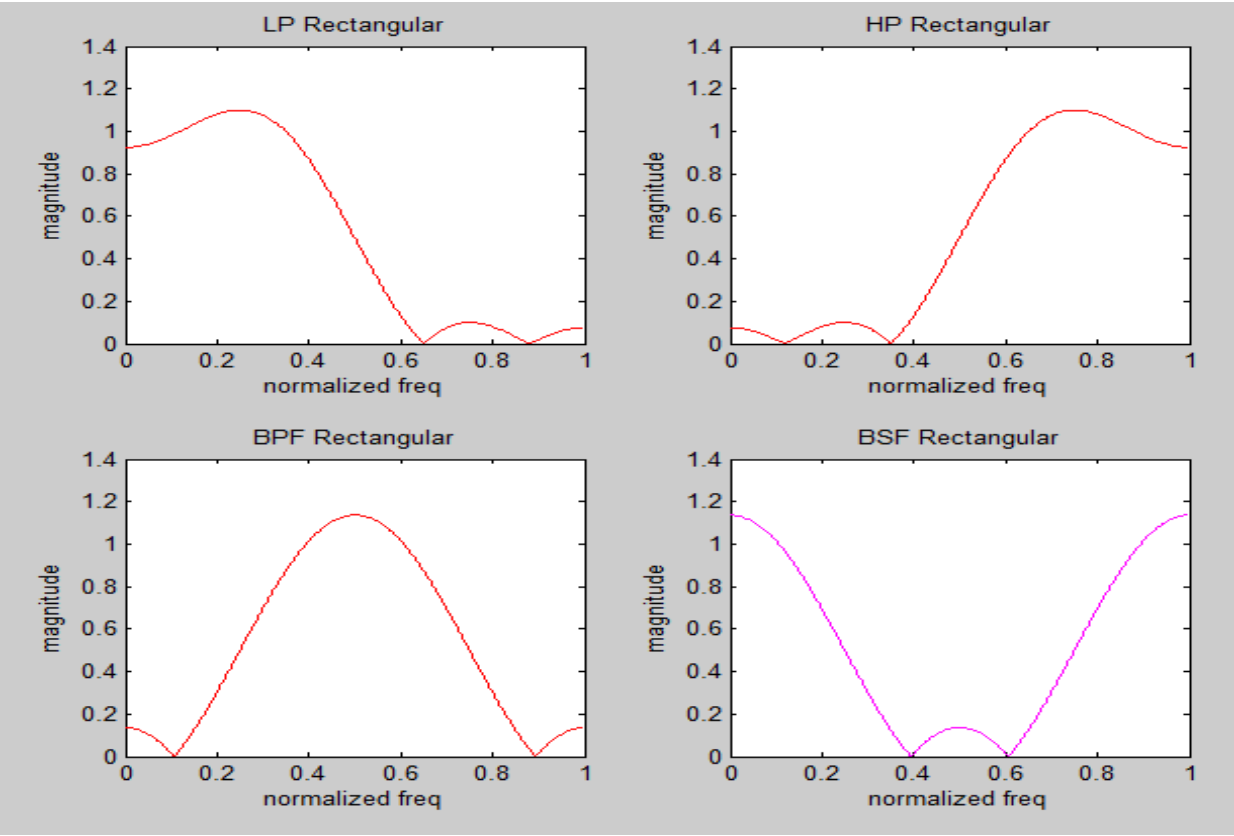
0.1065 -0.0008 -0.3176 0.5000 -0.3190 0.0007 0.1057

BPF filter co-efficient

-0.0002 -0.3185 0.0007 0.5000 -0.0007 -0.3182 0.0002

BSF filter co-efficient

0.0006 0.3180 0.0003 0.5000 -0.0003 0.3187 -0.0006



```
%FIR filter using HAMMING Window
clc;
clear all;
close all;
wc=0.5*pi;

N=input('enter the  length  of the filter=');
alpha=(N-1)/2;
eps=0.001;
n=0:1:N-1;

%LPF
hd=sin(wc*(n-alpha+eps))./(pi*(n-alpha+eps));
wh=hamming(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,1)
plot(w/pi,abs(h),'r')
xlabel ('normalized freq')
ylabel ('magnitude')
title('LPF HAMMING')
disp('LP filter co-efficient');
disp(hn)
hold on;

%HPF

hd=(sin(pi*(n-alpha+eps))-sin((n-alpha+eps)*wc))./(pi*(n-alpha+eps));
wh=hamming(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,2)
```

```
plot(w/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('HP HAMMING')
disp('HPF filter co-efficient');
disp(hn)
hold on;
```

% BPF

```
Wc1=0.25*pi;
Wc2=0.75*pi;
hd=(sin(Wc2*(n-alpha+eps))-sin(Wc1*(n-alpha+eps))).//((n-alpha+eps)*pi);
wh=hamming(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,3)
plot(W/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BPF HAMMING')
disp('BPF filter co-efficient');
disp(hn)ss
hold on;
```

% BSF

```
hd=(sin(Wc1*(n-alpha+eps))-sin(Wc2*(n-alpha+eps))+sin(pi*(n-alpha+eps))).//((n-alpha+eps)*pi);
wh=hamming(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,4)
```

```
plot(W/pi,abs(h),'m');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BSF HAMMING')
disp('BSF filter co-efficient');
disp(hn)
hold on;
```

RESULTS:

enter the length of the filter=7

LP filter co-efficient

-0.0085 0.0001 0.2453 0.5000 0.2449 -0.0001 -0.0085

HPF filter co-efficient

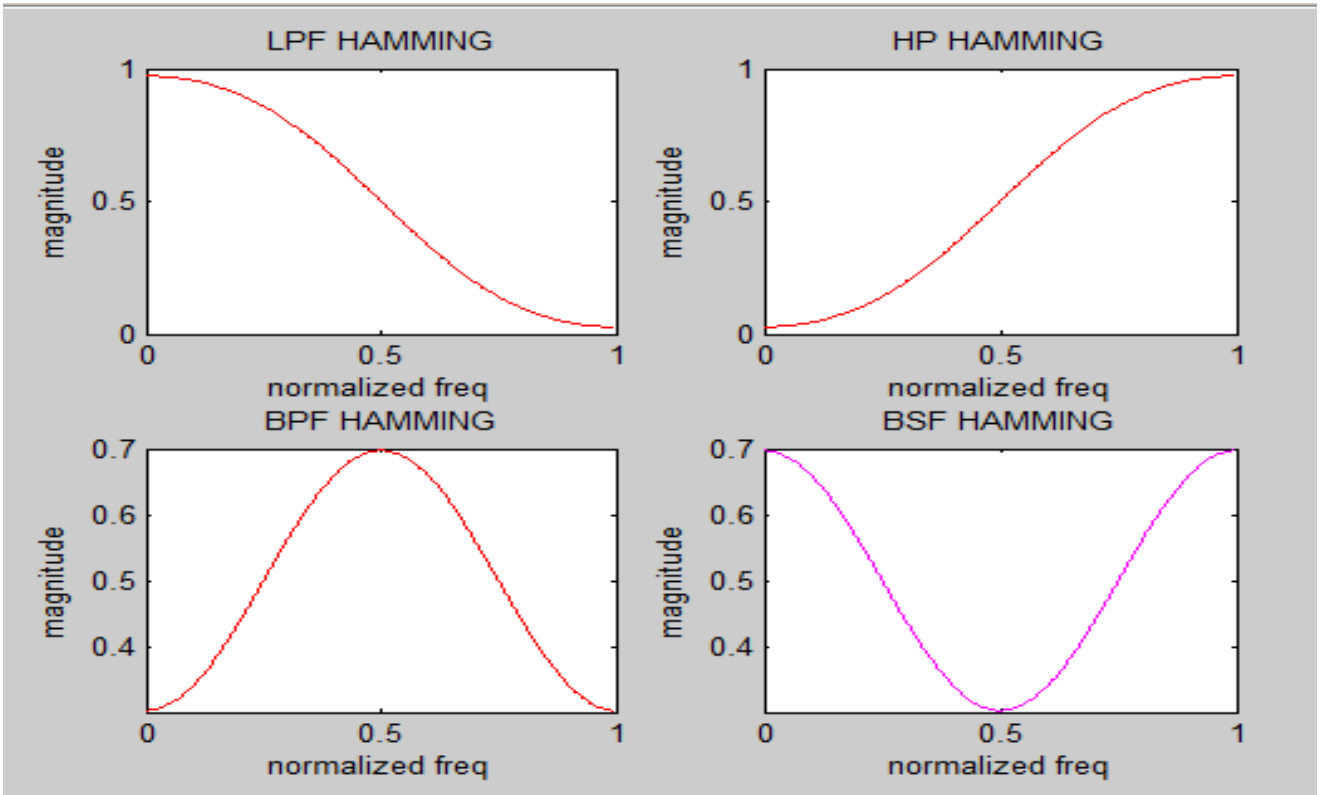
0.0085 -0.0002 -0.2446 0.5000 -0.2456 0.0002 0.0085

BPF filter co-efficient

-0.0000 -0.0987 0.0005 0.5000 -0.0005 -0.0986 0.0000

BSF filter co-efficient

0.0000 0.0986 0.0002 0.5000 -0.0002 0.0988 -0.0000



%FIR filter using HANNING Window

```
clc;
clear all;
close all;
wc=0.5*pi;
N=input('enter the  length  of the filter=');
alpha=(N-1)/2;
eps=0.001;
n=0:1:N-1;
```

%LPF

```
hd=sin(wc*(n-alpha+eps))./(pi*(n-alpha+eps));
wh=hanning(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,1)
plot(w/pi,abs(h),'r')
xlabel ('normalized freq')
ylabel ('magnitude')
```

```
title('LPF HANNING')
disp('LP filter co-efficient');
disp(hn)
hold on;
```

%HPF

```
hd=(sin(pi*(n-alpha+eps))-sin((n-alpha+eps)*wc))./(pi*(n-alpha+eps));
wh=hanning(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,2)
plot(w/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('HP HANNING')
disp('HPF filter co-efficient');
disp(hn)
hold on;
```

% BPF

```
Wc1=0.25*pi;
Wc2=0.75*pi;
hd=(sin(Wc2*(n-alpha+eps))-sin(Wc1*(n-alpha+eps)))./((n-alpha+eps)*pi);
wh=hanning(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,3)
plot(W/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BPF HANNING')
```

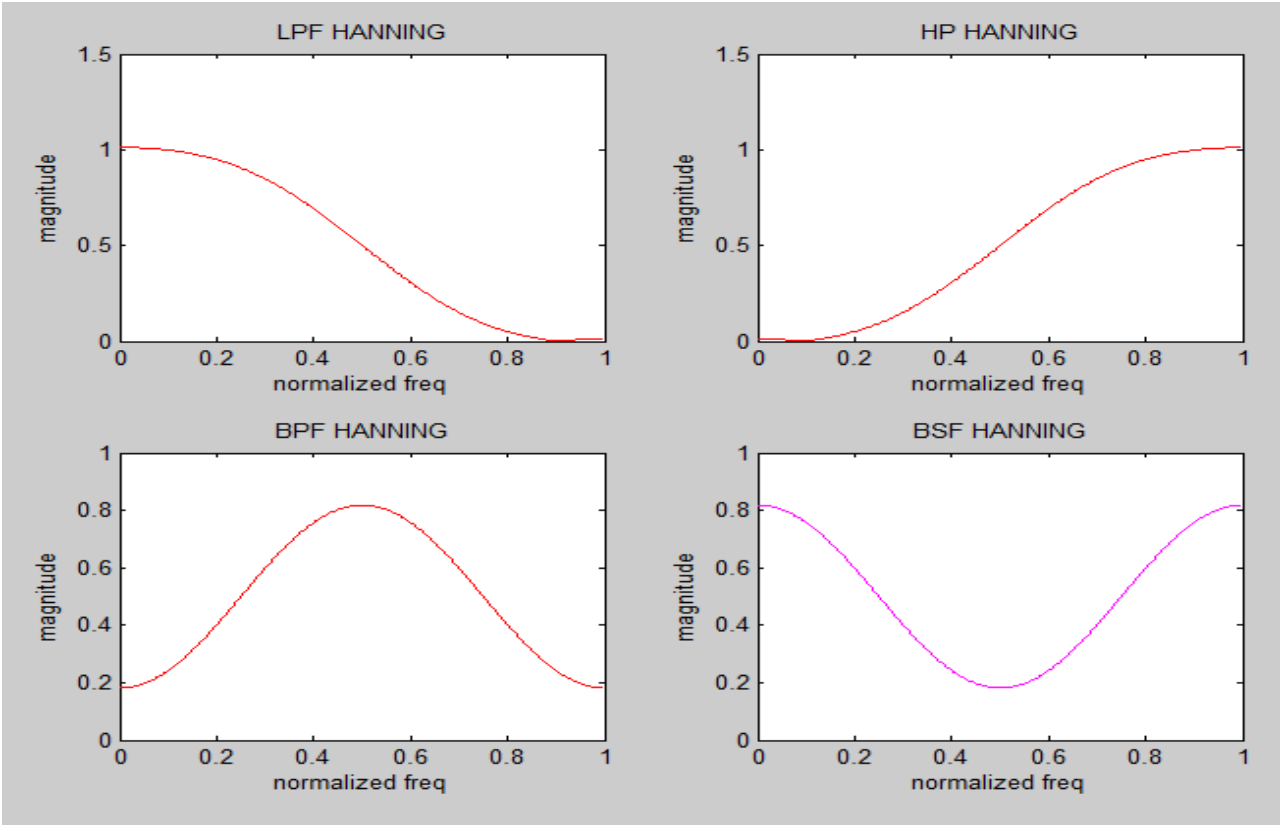
```
disp('BPF filter co-efficient');
disp(hn)
hold on;
```

% BSF

```
hd=(sin(Wc1*(n-alpha+eps))-sin(Wc2*(n-alpha+eps))+sin(pi*(n-alpha+eps))).((n-alpha+eps)*pi);
wh=hanning(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,4)
plot(W/pi,abs(h),'m');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BSF HANNING')
disp('BSF filter co-efficient');
disp(hn)
hold on;
```

RESULTS:

```
enter the  length  of the filter=7
LP filter co-efficient
  -0.0155    0.0001    0.2720    0.5000    0.2714   -0.0001   -0.0155

HPF filter co-efficient
   0.0156   -0.0004   -0.2711    0.5000   -0.2723    0.0004    0.0155

BPF filter co-efficient
  -0.0000   -0.1592    0.0006    0.5000   -0.0006   -0.1591    0.0000

BSF filter co-efficient
   0.0001    0.1590    0.0003    0.5000   -0.0002    0.1593   -0.0001
```



%FIR filter using BLACKMAN Window

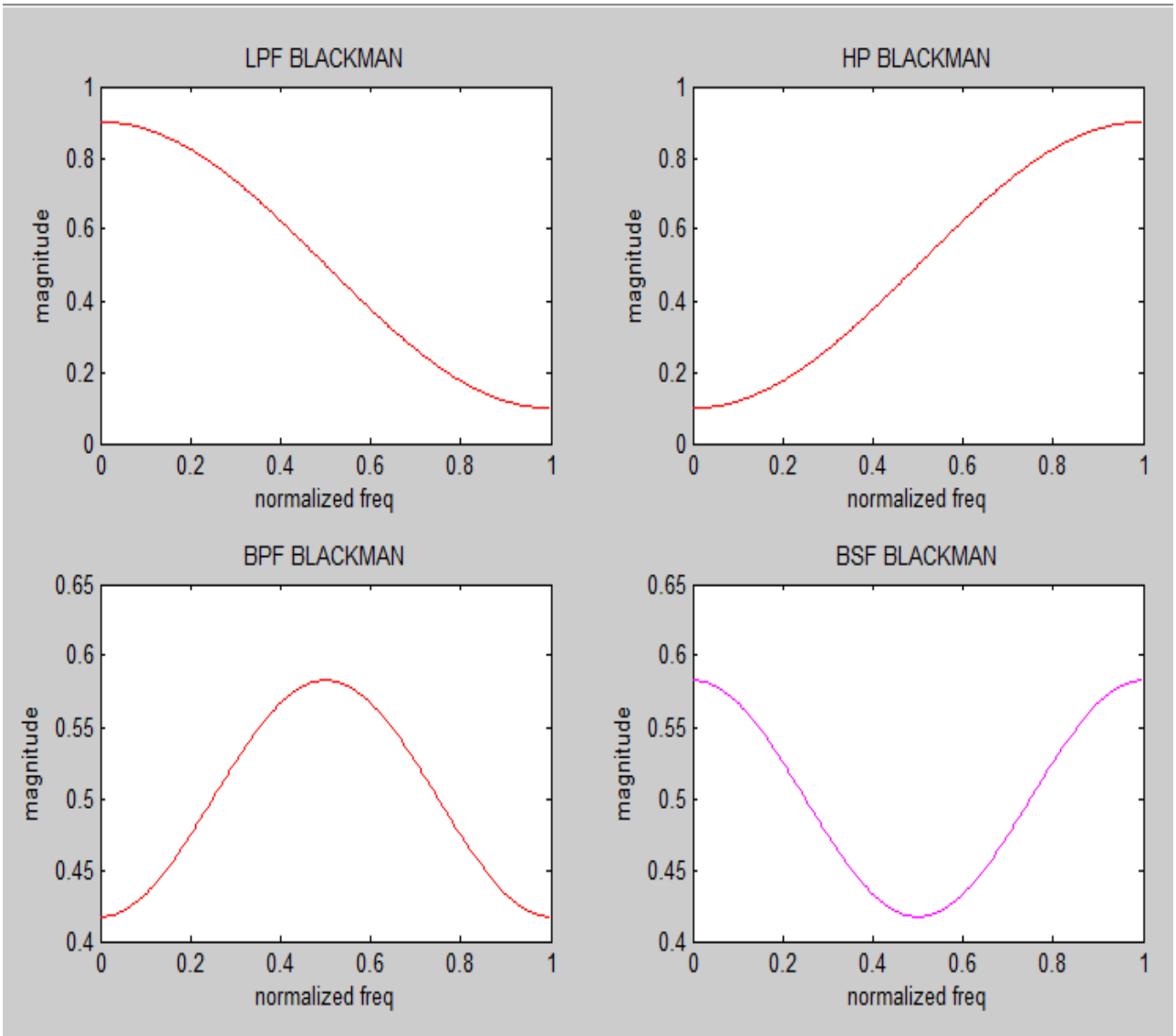
```
clc;
clear all;
close all;
wc=0.5*pi;
N=input('enter the  length  of the filter=');
alpha=(N-1)/2;
eps=0.001;
n=0:1:N-1;
%LPF
hd=sin(wc*(n-alpha+eps))./(pi*(n-alpha+eps));
wh=blackman(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,1)
plot(w/pi,abs(h),'r')
xlabel ('normalized freq')
ylabel ('magnitude')
```

```
title('LPF BLACKMAN')
disp('LP filter co-efficient');
disp(hn)
hold on;

%HPF
hd=(sin(pi*(n-alpha+eps))-sin((n-alpha+eps)*wc))./(pi*(n-alpha+eps));
wh=blackman(N);
hn=hd.*wh';
w=0:0.01:pi;
h=freqz(hn,1,w);
subplot(2,2,2)
plot(w/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('HP BLACKMAN')
disp('HPF filter co-efficient');
disp(hn)
hold on;

% BPF
Wc1=0.25*pi;
Wc2=0.75*pi;
hd=(sin(Wc2*(n-alpha+eps))-sin(Wc1*(n-alpha+eps)))./((n-alpha+eps)*pi);
wh=blackman(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,3)
plot(W/pi,abs(h),'r');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BPF BLACKMAN')
disp('BPF filter co-efficient');
disp(hn)
```

```
hold on;

% BSF
hd=(sin(Wc1*(n-alpha+eps))-sin(Wc2*(n-alpha+eps))+sin(pi*(n-alpha+eps))).((n-alpha+eps)*pi);
wh=blackman(N);
hn=hd.*wh';
W=0:0.01:pi;
h=freqz(hn,1,W);
subplot(2,2,4)
plot(W/pi,abs(h),'m');
xlabel ('normalized freq')
ylabel ('magnitude')
title('BSF BLACKMAN')
disp('BSF filter co-efficient');
disp(hn)
hold on;
```

RESULTS:

```
enter the  length  of the filter=7
LP filter co-efficient
   0.0000    0.0000    0.2007    0.5000    0.2003   -0.0000    0.0000

HPF filter co-efficient
  -0.0000   -0.0001   -0.2001    0.5000   -0.2010    0.0001   -0.0000

BPF filter co-efficient
   0.0000   -0.0414    0.0004    0.5000   -0.0004   -0.0414   -0.0000

BSF filter co-efficient
  -0.0000    0.0413    0.0002    0.5000   -0.0002    0.0414    0.0000
```


LPF BLACKMAN
magnitude
normalized freq
HP BLACKMAN
magnitude
normalized freq
BPF BLACKMAN
magnitude
normalized freq
BSF BLACKMAN
magnitude
normalized freq