Software Experiment - 2: Linear Convolution & Cross Correlation

Aim:

To perform linear convolution and cross correlation for the given discrete sequence x(n) & h(n).

To plot:

a) x(n)
   h(n)
   linear convolution of x(n) & h(n)
b) Discrete flipped x(n)
   h(n)
   cross correlation of x(n) & h(n)

With and without built-in functions.

Software required:

MATLAB

1. Linear Convolution:

a. With built-in function:

Code:

```
h = input('Enter the impulse sequence:\n');
x= input('Enter the input sequence:\n');
hlen=length(h);
xlen=length(x);
y=conv(x,h)
fprintf('Convolution x(m)*h(n-m):\n');
disp(y);
subplot(3,1,1);
stem(x,'color','b','linewidth',2);
set(gca,'fontsize',13, 'fontweight','bold');
xlabel('Number of samples,
n','fontsize',12,'fontweight','bold');
ylabel('Amplitude','fontsize',12,'fontweight','bold');
title('x(n)','fontsize',14);
grid on;
subplot(3,1,2);
stem(h,'color','r','linewidth',2);
set(gca,'fontsize',13, 'fontweight','bold');
xlabel('Number of
samples,n','fontsize',12,'fontweight','bold');
ylabel('Amplitude','fontsize',12,'fontweight','bold');
title('h(n)','fontsize',14);
grid on;
subplot(3,1,3);
stem(y,'color','g','linewidth',2);
set(gca,'fontsize',13, 'fontweight','bold');
xlabel('Number of
samples,n','fontsize',12,'fontweight','bold');
ylabel('Amplitude','fontsize',12,'fontweight','bold');
title('cross correlation result','fontsize',14);
grid on;
```

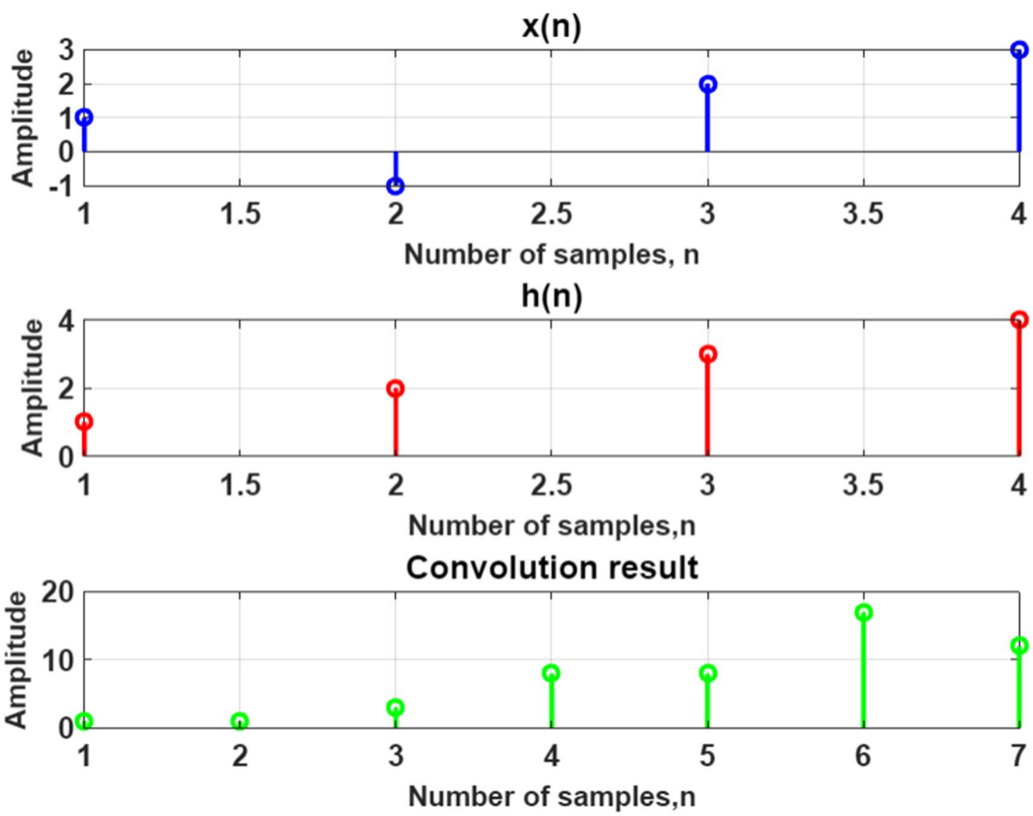
b. Without built-in function:

Code:

```
h = input('Enter the impulse sequence:\n');
x= input('Enter the input sequence:\n');
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
for i=1:n+m-1
    y(i)=0;
    for j=1:m
        if(i-j+1>0)
            y(i) = y(i)+X(j)*H(i-j+1);
        end
    end
end

fprintf('Convolution x(m)*h(n-m):\n');
disp(y);
subplot(3,1,1);
stem(x,'color','b','linewidth',2);
set(gca,'fontsize',13, 'fontweight','bold');
xlabel('Number of samples,
n','fontsize',12,'fontweight','bold');
ylabel('Amplitude','fontsize',12,'fontweight','bold');
title('x(n)','fontsize',14);
grid on;
subplot(3,1,2);
stem(h,'color','r','linewidth',2);
set(gca,'fontsize',13, 'fontweight','bold');
xlabel('Number of
samples,n','fontsize',12,'fontweight','bold');
ylabel('Amplitude','fontsize',12,'fontweight','bold');
title('h(n)','fontsize',14);
grid on;
subplot(3,1,3);
stem(y,'color','g','linewidth',2);
set(gca,'fontsize',13, 'fontweight','bold');
xlabel('Number of
samples,n','fontsize',12,'fontweight','bold');
ylabel('Amplitude','fontsize',12,'fontweight','bold');
title('Convolution result','fontsize',14);
grid on;
```

Output:

```
>> test
Enter the impulse sequence:
[1 -1 2 3]
Enter the input sequence:
[1 2 3 4]
Convolution x(m)*h(n-m):
     1     1     3     8     8    17    12
```



2. **Cross Correlation:**

a. With built-in function:

Code:

```
close all;
clear all;
% Enter the impulse sequence
h = input('Enter the impulse sequence: ');
% Enter the input sequence
x = input('Enter the input sequence: ');
hlen = length(h);
xlen = length(x);
y = xcorr(x,h);
fprintf('CROSS-CORRELATION x(m)*h(n-m):\n');
disp(y);
% Plotting
subplot(3, 1, 1);
stem(0:xlen-1, x, 'color', 'b', 'linewidth', 2);
set(gca, 'fontsize', 13, 'fontweight', 'bold');
xlabel('Number of samples, n', 'fontsize', 12,
'fontweight', 'bold');
ylabel('Amplitude', 'fontsize', 12, 'fontweight',
'bold');
title('Input Sequence x(n)', 'fontsize', 14);
grid on;
subplot(3, 1, 2);
stem(0:hlen-1, h, 'color', 'r', 'linewidth', 2);
set(gca, 'fontsize', 13, 'fontweight', 'bold');
xlabel('Number of samples, n', 'fontsize', 12,
'fontweight', 'bold');
ylabel('Amplitude', 'fontsize', 12, 'fontweight',
'bold');
title('Impulse Sequence h(n)', 'fontsize', 14);
grid on;
subplot(3, 1, 3);
stem(0:(xlen+hlen-2), y, 'color', 'g', 'linewidth', 2);
set(gca, 'fontsize', 13, 'fontweight', 'bold');
xlabel('Number of samples, n', 'fontsize', 12,
'fontweight', 'bold');
ylabel('Amplitude', 'fontsize', 12, 'fontweight',
'bold');
title('Cross-Correlation Result y(n)', 'fontsize', 14);
grid on;
```

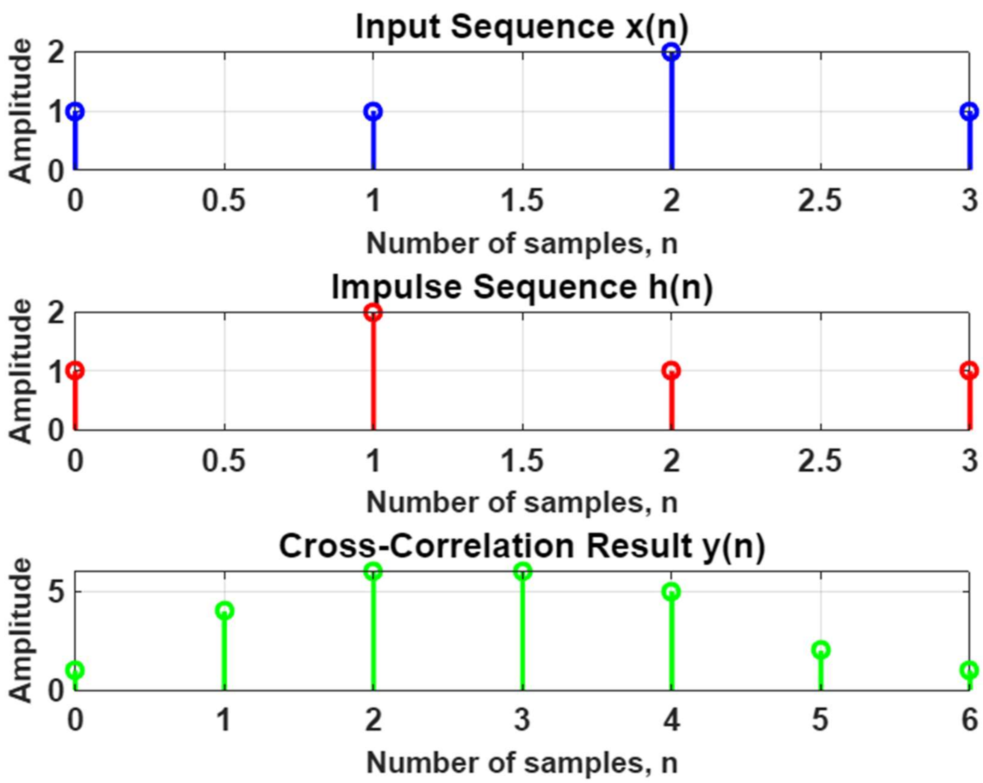
b. Without built-in function:

Code:

```
close all;
clear all;
% Enter the impulse sequence
h = input('Enter the impulse sequence: ');
% Enter the input sequence
x = input('Enter the input sequence: ');
hlen = length(h);
xlen = length(x);
x_reversed = fliplr(x);
y = zeros(1, xlen + hlen - 1);
for i = 1:xlen
for j = 1:hlen
y(1, i + j - 1) = y(1, i + j - 1) + h(j) * x_reversed(i);
end
end
fprintf('CROSS-CORRELATION x(m)*h(n-m):\n');
disp(y);
% Plotting
subplot(3, 1, 1);
stem(0:xlen-1, x, 'color', 'b', 'linewidth', 2);
set(gca, 'fontsize', 13, 'fontweight', 'bold');
xlabel('Number of samples, n', 'fontsize', 12,
'fontweight', 'bold');
ylabel('Amplitude', 'fontsize', 12, 'fontweight',
'bold');
title('Input Sequence x(n)', 'fontsize', 14);
grid on;
subplot(3, 1, 2);
stem(0:hlen-1, h, 'color', 'r', 'linewidth', 2);
set(gca, 'fontsize', 13, 'fontweight', 'bold');
xlabel('Number of samples, n', 'fontsize', 12,
'fontweight', 'bold');
ylabel('Amplitude', 'fontsize', 12, 'fontweight',
'bold');
title('Impulse Sequence h(n)', 'fontsize', 14);
grid on;
subplot(3, 1, 3);
stem(0:(xlen+hlen-2), y, 'color', 'g', 'linewidth', 2);
set(gca, 'fontsize', 13, 'fontweight', 'bold');
xlabel('Number of samples, n', 'fontsize', 12,
'fontweight', 'bold');
ylabel('Amplitude', 'fontsize', 12, 'fontweight',
'bold');
title('Cross-Correlation Result y(n)', 'fontsize', 14);
grid on;
```

Output:

```
test
Enter the impulse sequence:
[1 2 1 1]
Enter the input sequence:
[1 1 2 1]
CROSS-CORRELATION x(m)*h(n-m):
     1     4     6     6     5     2     1
```



Output Verification:

Date : 17-1-2024
Exp No : 02

Linear convolution + Cross correlation

Aim :

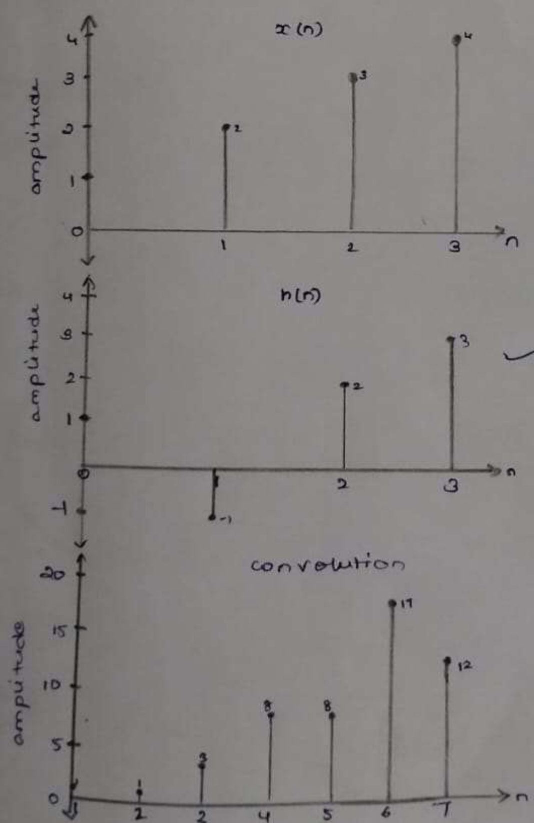
To perform linear convolution and cross correlation for the given discrete sequence x(n) + h(n)

To plot :

a) x(n)
h(n)
linear convolution of x(n) + h(n)

b) Discrete flipped x(n)
h(n)
cross correlation of x(n) + h(n)

$$y[n] = \sum_{k=-\infty}^{\infty} h[k]\, x[n-k]$$



17/01/24

Software Experiment - 3: System Response & Stability using Z-transform

Aim:

To perform the following tasks on Z-Transform, Poles and Zeros:

➢ Plotting Stable, Unstable and Marginally Stable Versions of Z-Transform graphs for the given system $x(z)$= $1-1.6180z^{-1}+z^{-2}$ /$1-1.5161z^{-1}+0.8781z^{-2}$

➢ Compute Poles, Zeros and infer the Stability of the given system:

$H(z)$= $1+3z^{-1}+2z^{-2}+3z^{-3}$ /$1+az^{-1}+bz^{-2}+cz^{-3}+dz^{-4}$ where a, b, c & d refer to the Register Number: 19BECabcd

➢ Determine the number of ROCs of the above done H(z) system and show all possible ROCs and infer if DTFT exists

➢ Identify an Unstable System and determine the Partial Fraction Expansion of a Rational Z-Transform, and Determine its Inverse Z-Transform.

Software required:

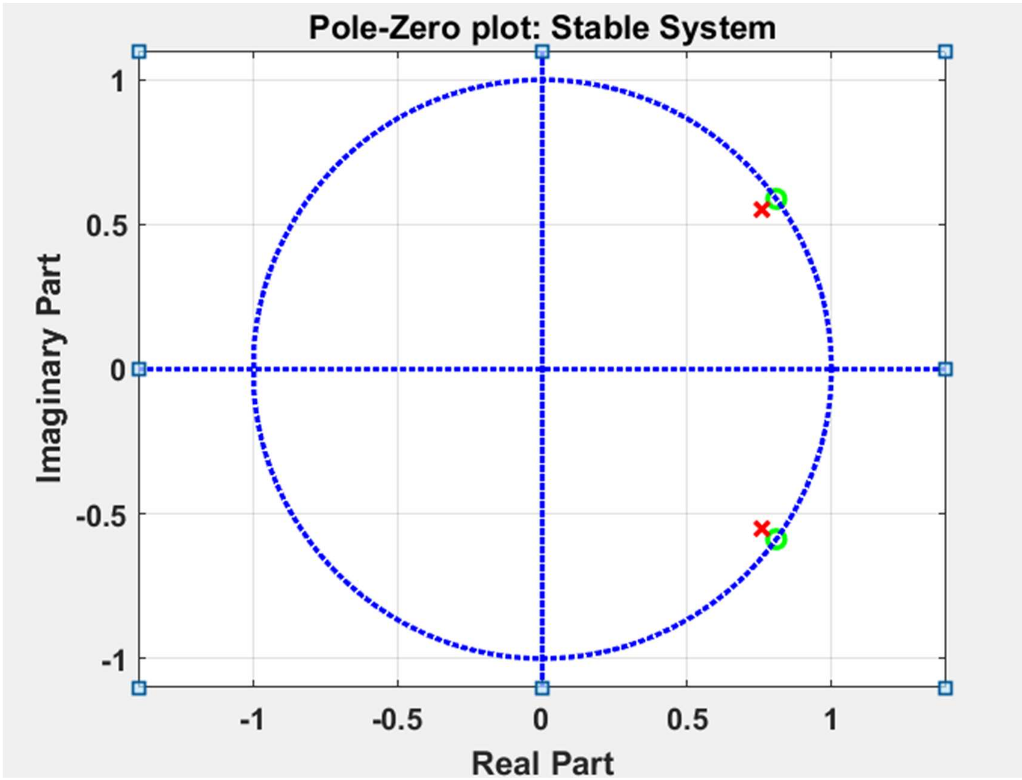
MATLAB

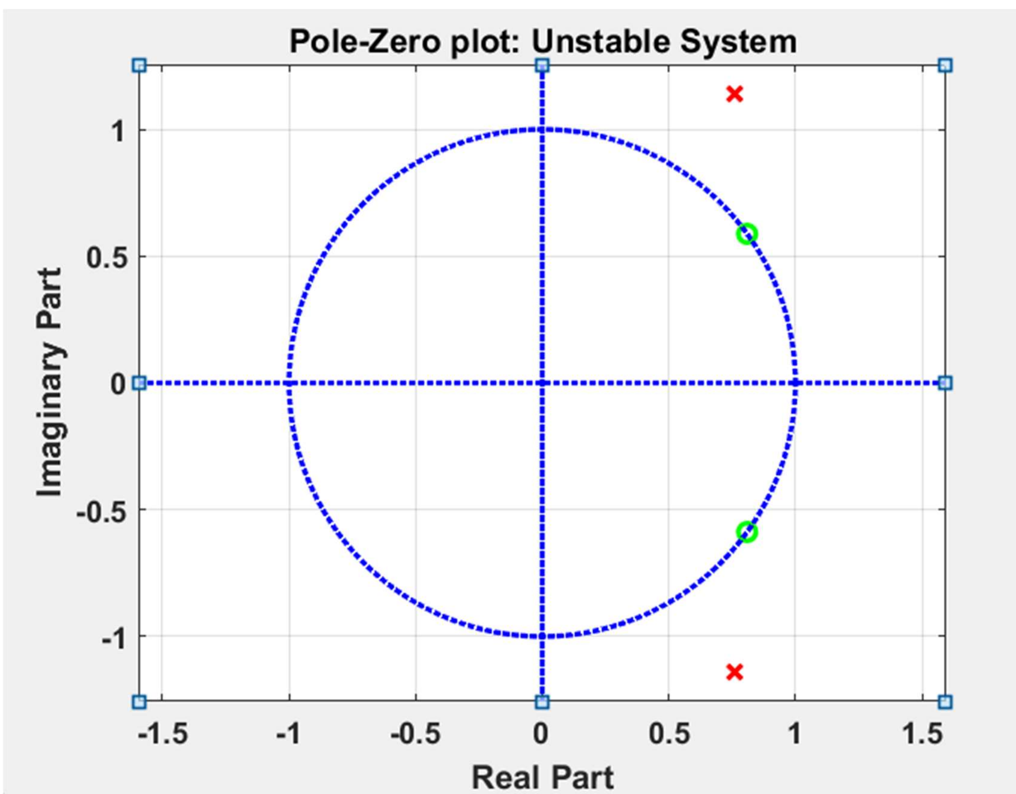
1. **Program 1 - Plotting stable, unstable & marginally stable given systems:**

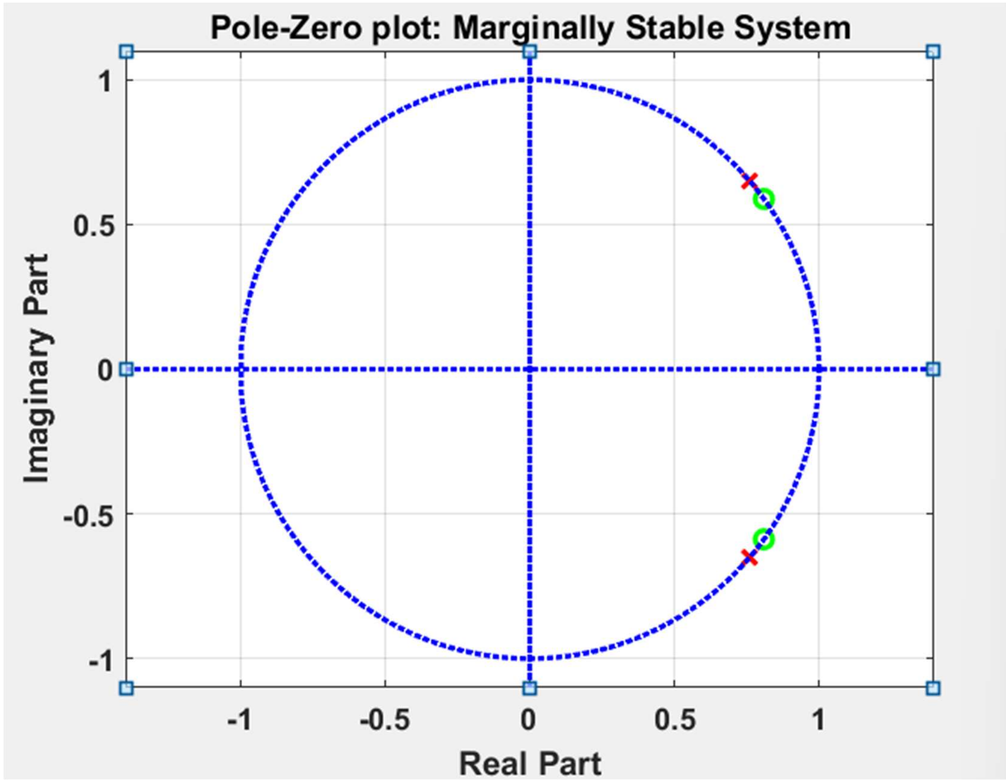
   Code:

```
clc
clear all
close all
b1=[1 -1.6180 1];
a1 = [1 -1.5161 0.8781]
[Z,P,K] = tf2zp(b1,a1);
figure(1);
zplane(b1,a1)
set(gca,'fontsize',13,'fontweight','bold');
[HZ1,HP1,HT1]=zplane(b1,a1);
set(findobj(HZ1,'Type','Line'),'Color','g','linewidth',2);
set(findobj(HT1,'Type','Line'),'Color','b','linewidth',2);
set(findobj(HP1,'Type','Line'),'Color','r','linewidth',2);
title('Pole-Zero plot: Stable System','fontsize',15);
grid on;

b1=[1 -1.6180 1];
a1 = [1 -1.5161 1.8781]
[Z1,P1,K1] = tf2zp(b1,a1);
figure(2);
zplane(b1,a1)
set(gca,'fontsize',13,'fontweight','bold');
[HZ1,HP1,HT1]=zplane(b1,a1);
set(findobj(HZ1,'Type','Line'),'Color','g','linewidth',2);
set(findobj(HT1,'Type','Line'),'Color','b','linewidth',2);
set(findobj(HP1,'Type','Line'),'Color','r','linewidth',2);
title('Pole-Zero plot: Unstable System','fontsize',15);
grid on;

b1=[1 -1.6180 1];
a1 = [1 -1.5161 1]
[Z2,P2,K2] = tf2zp(b1,a1);
figure(3);
zplane(b1,a1)
set(gca,'fontsize',13,'fontweight','bold');
[HZ1,HP1,HT1]=zplane(b1,a1);
set(findobj(HZ1,'Type','Line'),'Color','g','linewidth',2);
set(findobj(HT1,'Type','Line'),'Color','b','linewidth',2);
set(findobj(HP1,'Type','Line'),'Color','r','linewidth',2);
title('Pole-Zero plot: Marginally Stable
System','fontsize',15);
grid on;
```

Output:


Pole-Zero plot: Stable System
Imaginary Part
Real Part
1
0.5
0
-0.5
-1
-1
-0.5
0
0.5
1


Pole-Zero plot: Unstable System
Imaginary Part
Real Part
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

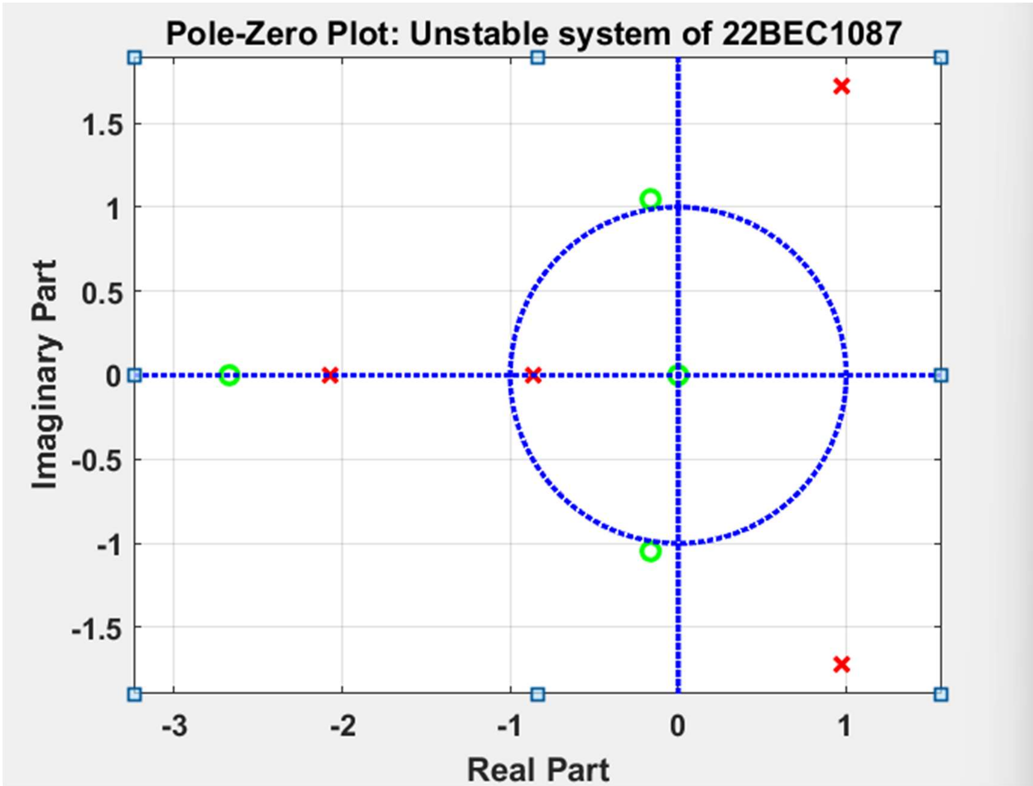


2. **Program 2 - Plotting the Unstable system of my registration number:**

   Code:

```
clc
clear all
close all
b1=[1 3 2 3];
a1 = [1 1 0 8 7] %Reg no = 22BEC1087
[Zeros,Poles,K] = tf2zp(b1,a1);
figure(1);
zplane(b1,a1)
set(gca,'fontsize',13,'fontweight','bold');
[HZ1,HP1,HT1]=zplane(b1,a1);
set(findobj(HZ1,'Type','Line'),'Color','g','linewidth',2);
set(findobj(HT1,'Type','Line'),'Color','b','linewidth',2);
set(findobj(HP1,'Type','Line'),'Color','r','linewidth',2);
title('Pole-Zero Plot: Unstable system of 22BEC1087
','fontsize',15);
grid on;
display(Zeros);
display(Poles);
display(K);
```

Output:




```
a1 =

     1     1     0     8     7


Zeros =

  -2.6717 + 0.0000i
  -0.1642 + 1.0469i
  -0.1642 - 1.0469i


Poles =

   0.9694 + 1.7220i
   0.9694 - 1.7220i
  -2.0747 + 0.0000i
  -0.8640 + 0.0000i


K =

     1
```

3. **Program 3 - Printing the ROCs of the given system and if DTFT exists:**

Code:

```
clc
clear all
close all
b1=[1 3 2 3];
a1 = [1 1 0 8 7] %Reg no = 22BEC1087
[Zeros,Poles,K] = tf2zp(b1,a1);
figure(1);
zplane(b1,a1)
set(gca,'fontsize',13,'fontweight','bold');
[HZ1,HP1,HT1]=zplane(b1,a1);
set(findobj(HZ1,'Type','Line'),'Color','g','linewidth',2);
set(findobj(HT1,'Type','Line'),'Color','b','linewidth',2);
set(findobj(HP1,'Type','Line'),'Color','r','linewidth',2);
title('Pole-Zero Plot: Unstable system of 22BEC1087
','fontsize',15);
grid on;
M=abs(Poles)
N=max(M);
if(N<1)
    fprintf('DTFT EXISTS');
else
    fprintf('DTFT NOT EXISTS');
end
```

Output:

```
a1 =

     1     1     0     8     7


M =

    1.9761
    1.9761
    2.0747
    0.8640

DTFT NOT EXISTS>>
```

4. **Program 4 – Finding the ZT and IZT of an unstable system:**

Code:

```
syms z
b1=[1 3 2 3];
a1 = [1 1 0 8 7];
[Z,P,K] = tf2zp(b1,a1);
[H,T]=impz(b1,a1);
[R,P1,K1]=residuez(b1,a1);
ZT = ztrans(P,z)
IZT = iztrans(ZT,z)

figure(1);
zplane(b1,a1)
set(gca,'fontsize',13,'fontweight','bold');
[HZ1,HP1,HT1]=zplane(b1,a1);
set(findobj(HZ1,'Type','Line'),'Color','g','linewidth',2);
set(findobj(HT1,'Type','Line'),'Color','b','linewidth',2);
set(findobj(HP1,'Type','Line'),'Color','r','linewidth',2);
title('Pole-Zero Plot: Unstable system of 22BEC1087
','fontsize',15);
grid on;
```

Output:

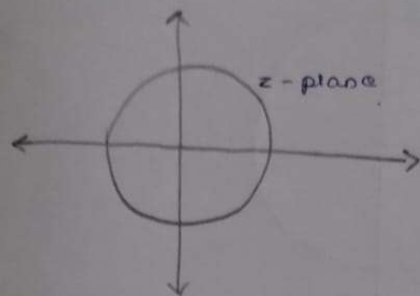
```
>> test

ZT =

(z*(4365679556904593/4503599627370496 + 7755073583733221i/4503599627370496))/(z - 1)
(z*(4365679556904593/4503599627370496 - 7755073583733221i/4503599627370496))/(z - 1)
                                         -(1167960229380269*z)/(562949953421312*(z - 1))
                                         -(972819226534377*z)/(1125899906842624*(z - 1))


IZT =

4365679556904593/4503599627370496 + 7755073583733221i/4503599627370496
4365679556904593/4503599627370496 - 7755073583733221i/4503599627370496
                                         -1167960229380269/562949953421312
                                         -972819226534377/1125899906842624
```

Output Verification:

Date: 24-1-24

Exp No: 03

System Response & Stability Analysis using Z-transform



Stable - All poles must lie inside O^{le}

Unstable - Any one pole lies outside O^{le}

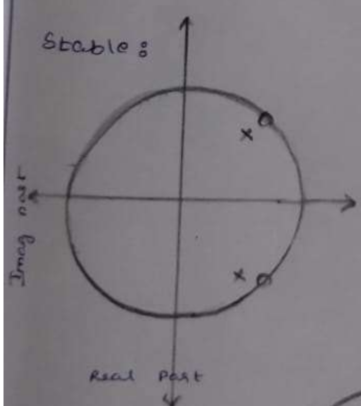
Marginally stable - If any one of the poles on the unit O^{le}

Program 1:-

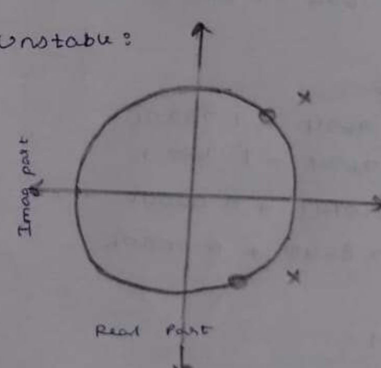
$$X(z) = \frac{1 - 1.6180z^{-1} + z^{-2}}{1 - 1.5161z^{-1} + 0.8781z^{-2}}$$

$$X(z) = \frac{1 - 1.6180z^{-1} + z^{-2}}{1 - 1.5161z^{-1} + 1.8781z^{-2}}$$

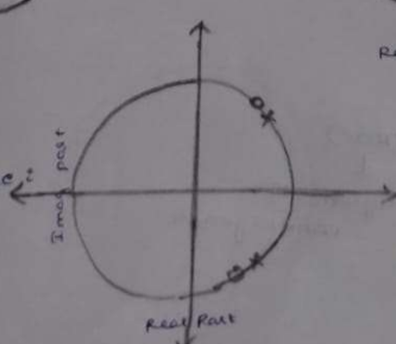
Stable:



Unstable:



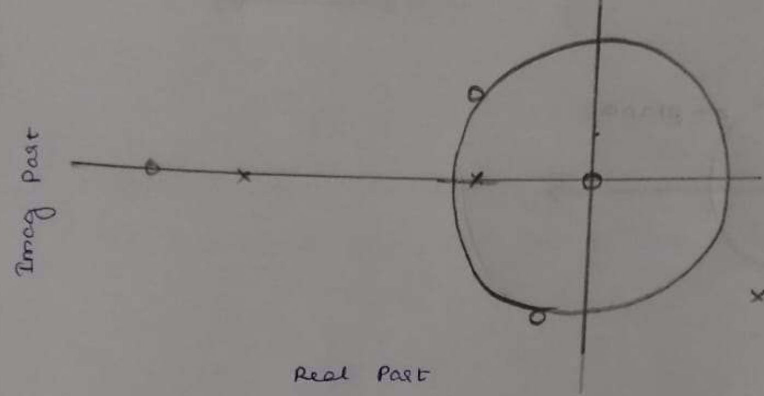
Marginally Stable:



$$X(z) = \frac{1 - 1.6180z^{-1} + z^{-2}}{1 - 1.5161z^{-1} + 1z^{-2}}$$

Program 2:-

Pole - zero Plot: Unstable System of 22BEC1087



Program 3:-

a1 =

1 1 0 8 7

zeros =

-2.6717 + 0.0000i
-0.1642 + 1.0469i
-0.1642 - 1.0469i

poles =

0.9694 + 1.7220i
0.9694 - 1.7220i
-2.0747 + 0.0000i
-0.8640 + 0.0000i

K =

1

H =

1.9761
1.9761
2.0747
0.8640

(ROC)
↓
Region of convergence.

Program 4:

ZT =

(z* (4365679556904587/ 4503599627370496 + 2423460494916663i /

140737488355328)) /(z-1)

IZT =

4365679556904587/4503599627370496 + 2423460494916663i/140737488355328

24/01/24

Software Experiment - 5: Linear Convolution in code composer studio

Aim:

To perform linear convolution of two functions using code composer studio (CCS).

Software required:

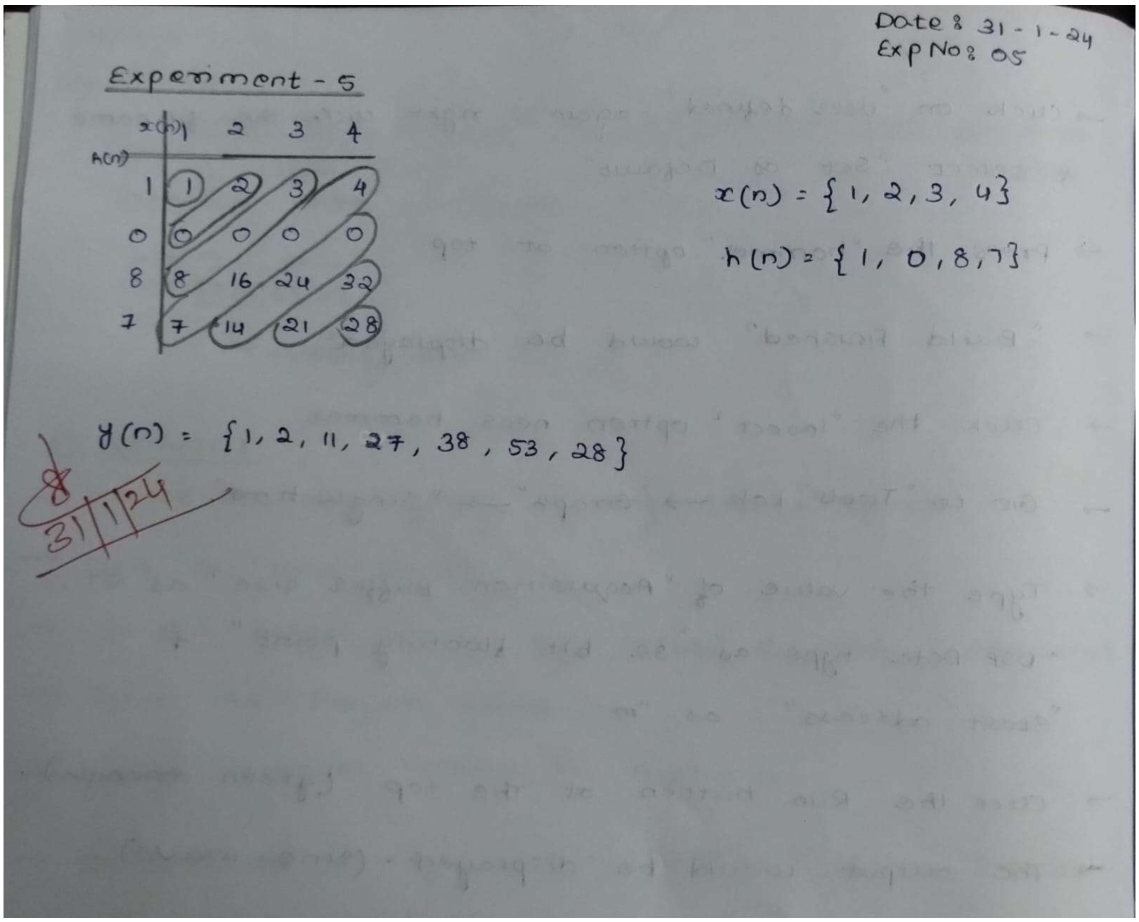
CCS (Code Composer Studio)

Code:

```
#include<stdio.h>
int x[15],h[15],y[15];
int main() {
  int i,j,m,n;
  printf("Enter value for m: ");
  scanf("%d",&m);
  printf("Enter value for n: ");
  scanf("%d",&n);

  printf("Enter values for i/p x(n):\n");
  for(i=0;i<m;i++)
  scanf("%d",&x[i]);

  printf("Enter Values for i/p h(n):\n");
  for(i=0;i<n; i++)
  scanf("%d",&h[i]);

  // padding of zeors
  for(i=m;i<=m+n-1;i++)
  x[i]=0;
  for(i=n;i<=m+n-1;i++)
  h[i]=0;

  /* convolution operation */
  for(i=0;i<m+n-1;i++) {
    y[i]=0;
    for(j=0;j<=i;j++) {
      y[i]=y[i]+(x[j]*h[i-j]);
    } }
```

```
    //displaying the o/p
    for(i=0;i<m+n-1;i++)
      printf("\n The Value of output y[%d]=%d",i,y[i]);
}
```

Output Verification:



Software Experiment - 6: Designing various IIR Filters

Aim:

➢ To design a Low Pass Butterworth Filter given the frequency and ripple parameters, plot its magnitude and phase responses and compute its order and cut-off frequency.

➢ To design a High Pass Butterworth Filter given the frequency and ripple parameters, plot its magnitude and phase responses and compute its order and cut-off frequency.

➢ To design a Band-Pass Butterworth Filter given the normalised frequency and ripple parameters, plot its magnitude and phase responses and compute its order and cut-off frequencies.

➢ To design a Band-Stop Butterworth Filter given the normalised frequency and ripple parameters, plot its magnitude and phase responses and compute its order and cut-off frequencies.

Software required:

MATLAB

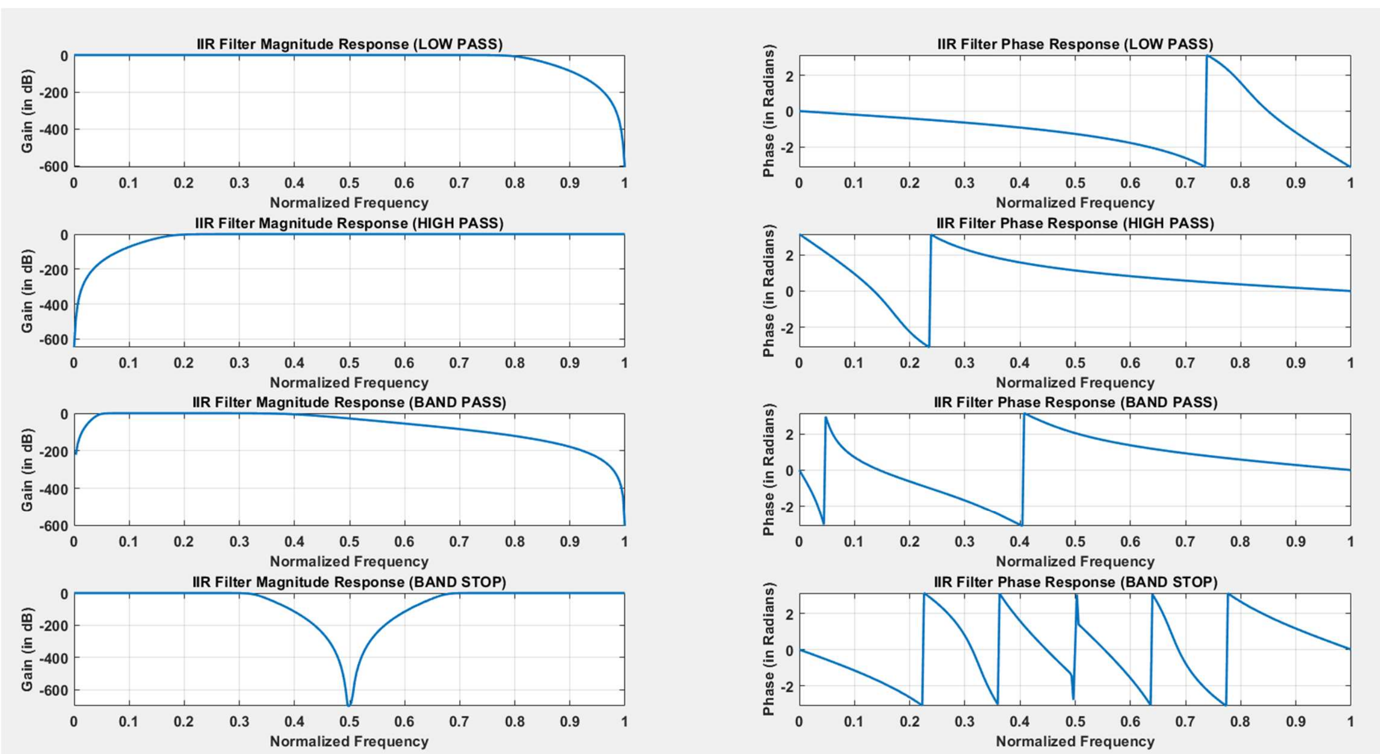
Code:

```
FC1 = 400;
FS1 = 1000;
[b1,a1] = butter(6,FC1/(FS1/2),'low');
w1 = 0:0.01:pi;
[h1,Om1] = freqz(b1,a1,w1);
M1 = 20*log(abs(h1));
A1 = angle(h1);

subplot(4,2,1);
plot(Om1/pi,M1,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Gain (in dB)','fontsize',13,'fontweight','bold');
title('IIR Filter Magnitude Response (LOW PASS)','fontsize',13);
grid on;

subplot(4,2,2);
plot(Om1/pi,A1,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Phase (in Radians)','fontsize',13,'fontweight','bold');
title('IIR Filter Phase Response (LOW PASS)','fontsize',13);
grid on;

FC2 = 90;
FS2 = 1000;
[b2,a2] = butter(6,FC2/(FS2/2),'high');
w2 = 0:0.01:pi;
[h2,Om2] = freqz(b2,a2,w2);
M2 = 20*log(abs(h2));
A2 = angle(h2);

subplot(4,2,3);
plot(Om2/pi,M2,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Gain (in dB)','fontsize',13,'fontweight','bold');
title('IIR Filter Magnitude Response (HIGH PASS)','fontsize',13);
grid on;
```

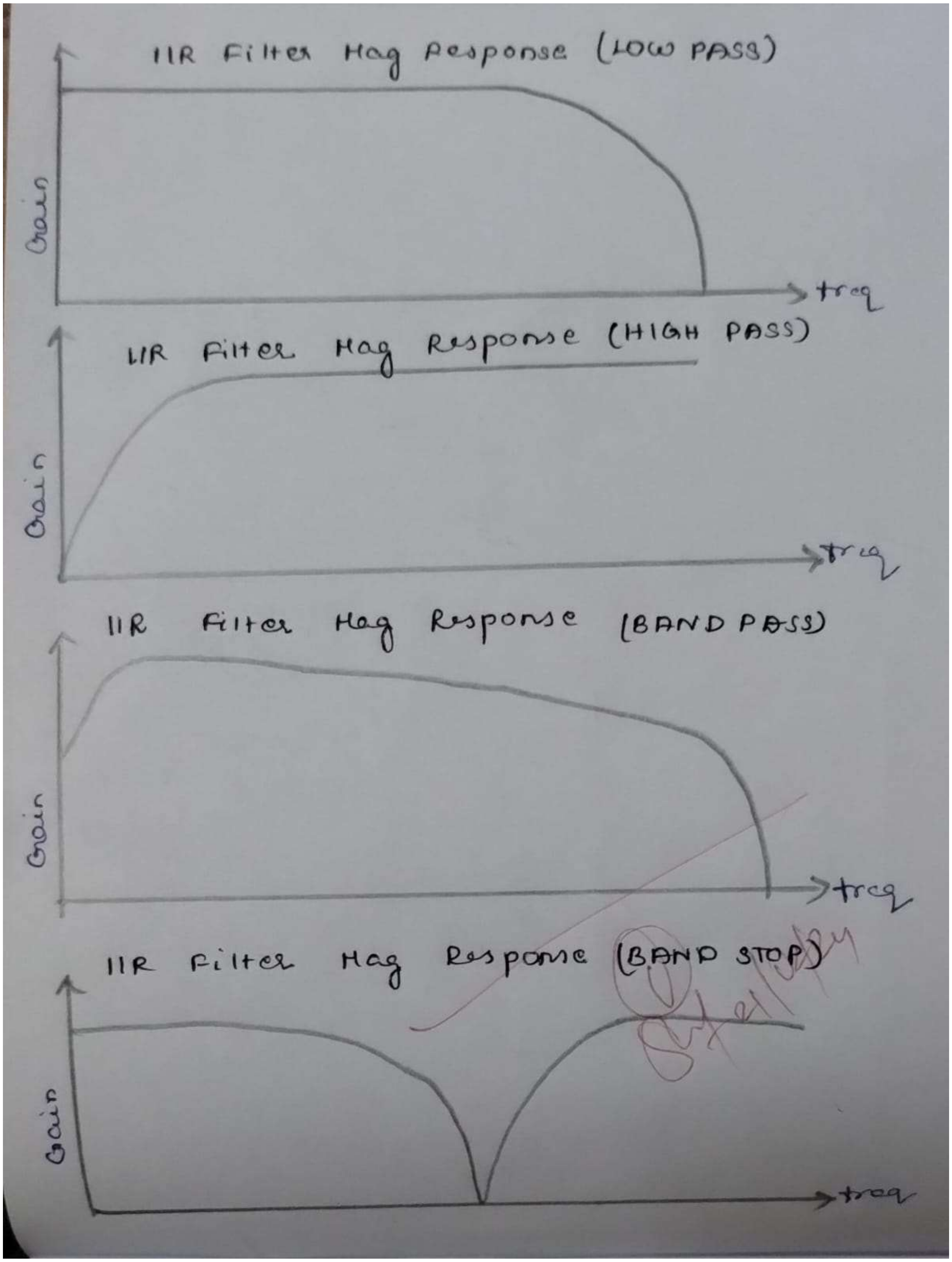
```
subplot(4,2,4);
plot(Om2/pi,A2,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Phase (in Radians)','fontsize',13,'fontweight','bold');
title('IIR Filter Phase Response (HIGH PASS)','fontsize',13);
grid on;

FS3 = 3000;
wp3 = [200 900]/FS3;
ws3 = [100 1100]/FS3;
[n3,wn3] = buttord(wp3,ws3,0.1,1);
[b3,a3] = butter(n3,wn3);
w3= 0:0.01:pi;
[h3,Om3] = freqz(b3,a3,w3);
M3 = 20*log(abs(h3));
A3 = angle(h3);

subplot(4,2,5);
plot(Om3/pi,M3,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Gain (in dB)','fontsize',13,'fontweight','bold');
title('IIR Filter Magnitude Response (BAND PASS)','fontsize',13);
grid on;

subplot(4,2,6);
plot(Om3/pi,A3,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Phase (in Radians)','fontsize',13,'fontweight','bold');
title('IIR Filter Phase Response (BAND PASS)','fontsize',13);
grid on;

[n4,wn4] = buttord([0.3,0.7],[0.4,0.6],0.4,50);
[b4,a4] = butter(n4,wn4,'stop');
w4= 0:0.01:pi;
[h4,Om4] = freqz(b4,a4,w4);
M4 = 20*log(abs(h4));
A4 = angle(h4);


subplot(4,2,7);
```

```
plot(Om4/pi,M4,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Gain (in dB)','fontsize',13,'fontweight','bold');
title('IIR Filter Magnitude Response (BAND STOP)','fontsize',13);
grid on;

subplot(4,2,8);
plot(Om4/pi,A4,'LineWidth',2);
set(gca,'fontsize',13,'fontweight','bold');
xlabel('Normalized Frequency','fontsize',13,'fontweight','bold');
ylabel('Phase (in Radians)','fontsize',13,'fontweight','bold');
title('IIR Filter Phase Response (BAND STOP)','fontsize',13);
grid on;
```

Output:



Output Verification:


IIR Filter Mag Response (LOW PASS)
Gain
freq
IIR Filter Mag Response (HIGH PASS)
Gain
freq
IIR Filter Mag Response (BAND PASS)
Gain
freq
IIR Filter Mag Response (BAND STOP)
Gain
freq

Software Experiment - 7: Performing DFT & IDFT in Code Composer Studio

Aim:

To perform 8-point DFT and IDFT in code composer studio and learn it's working.

Software required:

Code Composer Studio

Programs:

1. **Discrete Fourier Transform (DFT):**

   Code:

```
#include<stdio.h>
#include<math.h>
int main(){
float y[]={1,2,3,4,1,0,8,7};
float yr[8];
float yi[8];
int n,k;
for(k = 0; k < 8; k++) {
    yr[k] = 0;
    yi[k] = 0;
    for (n = 0; n < 8; n++) {
      yr[k] = (yr[k] + y[n] * cos(2*3.1415*k*n / 8));
      yi[k] = (yi[k] - y[n] * sin(2*3.1415*k*n / 8));
    }
    printf("(%f) + j(%f)\n", yr[k], yi[k]);
  }
}
```

Output:

```
(26.000000) + j(0.000000)
(3.533988) + j(5.707653)
(-9.002732) + j(8.997684)
(-3.534335) + j(-4.295088)
(0.000001) + j(-0.000463)
(-3.537530) + j(4.289236)
(-8.991797) + j(-9.006946)
(3.546354) + j(-5.703277)
```

2. **Inverse Discrete Fourier Transform (DFT):**

Code:

```
#include <stdio.h>
#include <math.h>
#include <complex.h>
double complex wkn (int k,int n,int N)
{
   double pi = 3.14159265359;
   double x = 2*pi*k*n/N;
     return cos(x) - sin(x)*I;
}
int main(void)
{
     int x[] = {1,2,3,4,1,0,8,7};
     double complex X[8];
     int i,j;
     for (i=0 ; i<8 ; i++)
     {for(j=0 ; j<8 ; j++)
               {X[i] += x[j]*wkn(j,i,8);
               }
     }
     i=0;
     printf("DFT for (1,2,3,4,1,0,8,7) :\n");

for (i=0 ; i< 8 ; i++)
     {
        printf("%.4f + j(%.4f)\n",creal(X[i]),cimag(X[i]));
     }
     double complex H[8];
     i=0,j=0;
     for (i=0 ; i<8 ; i++)
```

```
{for(j=0 ; j<8 ; j++)
        {H[i] += X[j]*wkn(-j,i,8);
        }
        H[i]= H[i]/8;
}
i=0;
printf("IDFT :\n");
for (i=0 ; i< 8 ; i++)
{
  printf("%.0f ",creal(H[i]));
}}
```

Output:

```
DFT for (1,2,3,4,1,0,8,7) :
26.0000 + j(0.0000)
3.5355 + j(5.7071)
-9.0000 + j(9.0000)
-3.5355 + j(-4.2929)
0.0000 + j(0.0000)
-3.5355 + j(4.2929)
-9.0000 + j(-9.0000)
3.5355 + j(-5.7071)
IDFT :
1 2 3 4 1 0 8 7
```

Output Verification:

Experiment - 7 DFT + IDFT in CCS

$x(n) = \{1, 2, 3, 4, 1, 0, 8, 7\}$

$X(0) = 26$

$X(1) = 3.54 + 5.71j$

$X(2) = -9 + 9j$

$X(3) = -3.54 - 4.29j$

$X(4) = 0$

$X(5) = -3.54 + 4.29j$

$X(6) = -9 - 9j$

$X(7) = 3.54 - 5.71j$

28/02/24

Software Experiment - 8: Study Of Various Iir Filter Characteristics (Butterworth, Chebyshev Type 1 & 2)

Aim:

To Design and Perform the Characteristic Analysis of Low Pass, High Pass, Band Pass & Band Stop Filters with Butterworth, Chebyshev – I and Chebyshev – II type of Frequency Responses / Analysis with Varying Orders.

Software required:

MATLAB

Programs:

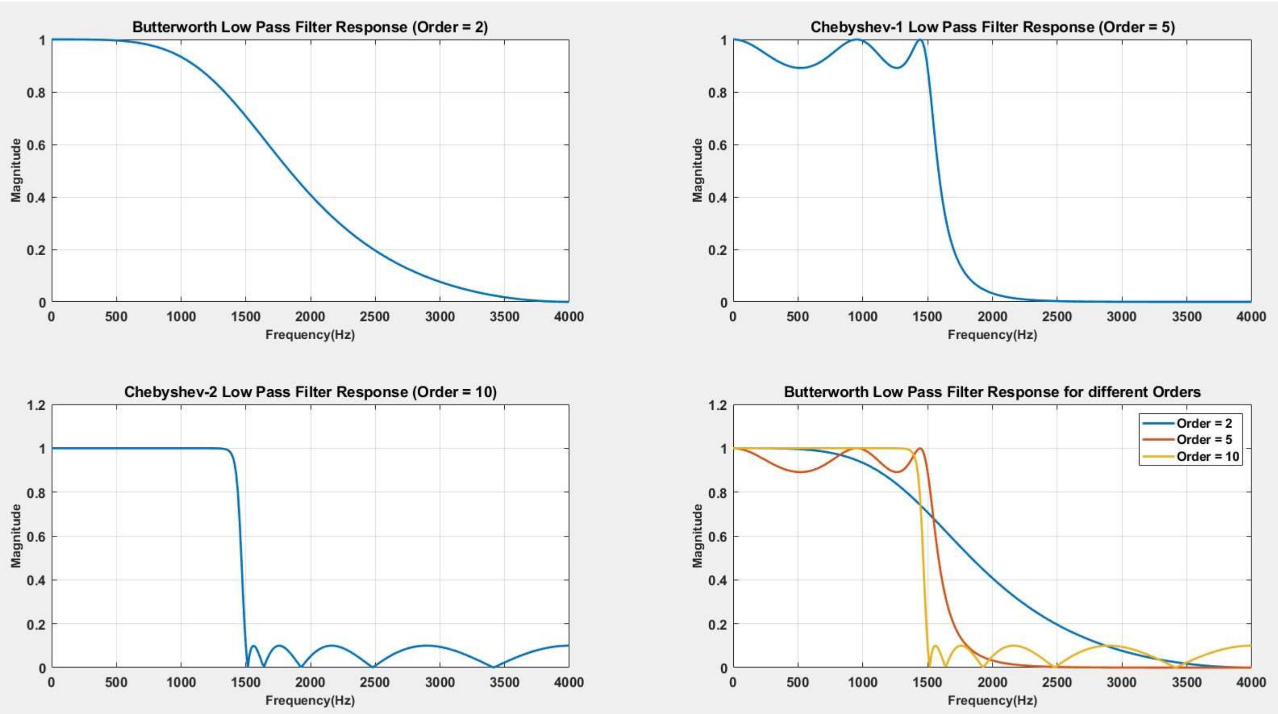
1. **Low-pass filter:**

Code:

```
FC1 = 1500;
FC2 = 1500;
FS = 8000;
N1 = 2;
N2 = 5;
N3 = 10;
Rp = 1;
Rs = 20;

[b1,a1] = butter(N1,2*FC1/FS,'low');
[b2,a2] = cheby1(N2,Rp,2*FC1/FS,'low');
[b3,a3] = cheby2(N3,Rs,2*FC1/FS,'low');
w = 0:0.01:pi;

[h1,om1] = freqz(b1,a1,w);
m1 = abs(h1);
[h2,om2] = freqz(b2,a2,w);
m2 = abs(h2);
[h3,om3] = freqz(b3,a3,w);
m3 = abs(h3);

subplot(2,2,1);
```

```
plot(om1/pi*FS/2, m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Butterworth Low Pass Filter Response (Order =
',num2str(N1),')'],'Fontsize',14);
ylabel('Magnitude','fontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,2);
plot(om2/pi*FS/2,m2,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-1 Low Pass Filter Response (Order =
',num2str(N2),')'],'Fontsize',14);
ylabel('Magnitude','FontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,3);
plot(om3/pi*FS/2,m3,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-2 Low Pass Filter Response (Order =
',num2str(N3),')'],'Fontsize',14);
ylabel('Magnitude','FontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,4);
plot(om3/pi*FS/2,m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title('Butterworth Low Pass Filter Response for different Orders','Fontsize',14);
ylabel('Magnitude','FontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;
hold on;
plot(om3/pi*FS/2,m2,'linewidth',2);
hold on;
plot(om3/pi*FS/2,m3,'linewidth',2);
legend(['Order = ',num2str(N1)]',['Order = ',num2str(N2)]',['Order =
',num2str(N3)]');
grid on;
```

Output:



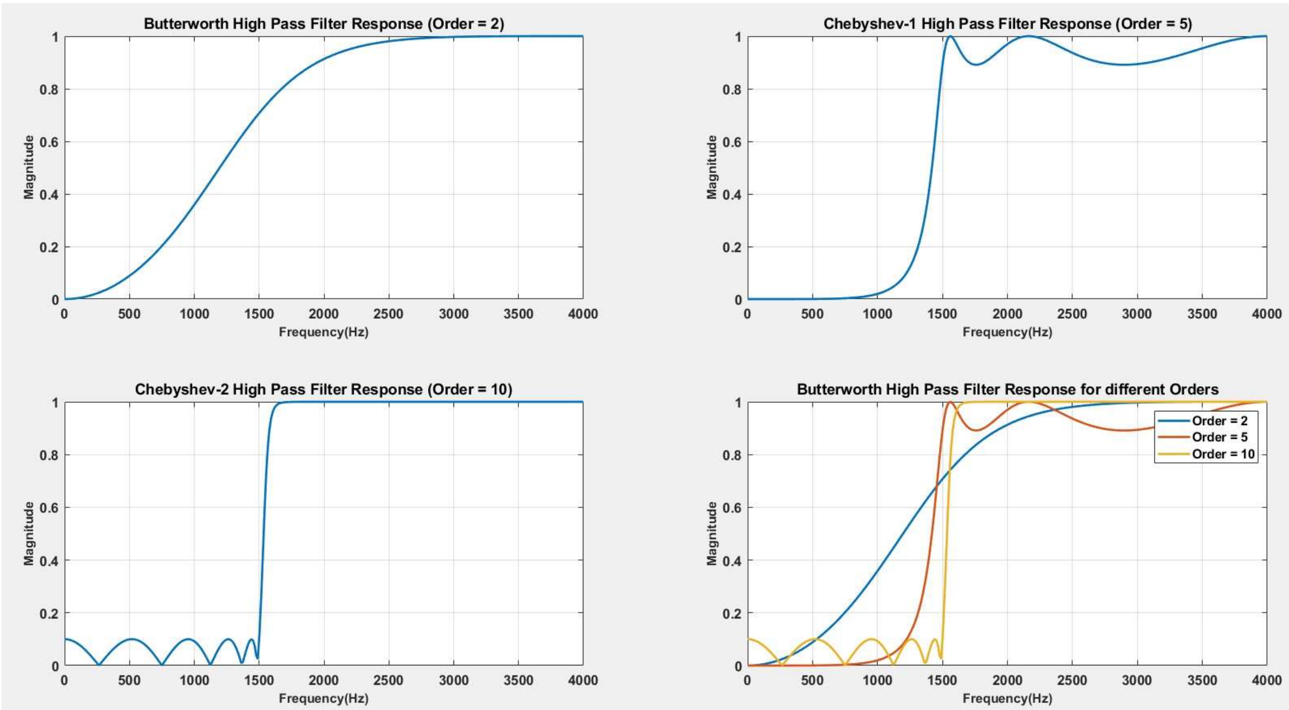
2. High-pass filter:

Code:

```
FC1 = 1500;
FC2 = 1500;
FS = 8000;
N1 = 2;
N2 = 5;
N3 = 10;
Rp = 1;
Rs = 20;

[b1,a1] = butter(N1,2*FC1/FS,'high');
[b2,a2] = cheby1(N2,Rp,2*FC1/FS,'high');
[b3,a3] = cheby2(N3,Rs,2*FC1/FS,'high');
w = 0:0.01:pi;

[h1,om1] = freqz(b1,a1,w);
m1 = abs(h1);
[h2,om2] = freqz(b2,a2,w);
m2 = abs(h2);
[h3,om3] = freqz(b3,a3,w);
```

```
m3 = abs(h3);

subplot(2,2,1);
plot(om1/pi*FS/2, m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Butterworth High Pass Filter Response (Order =
',num2str(N1),')'],'Fontsize',14);
ylabel('Magnitude','fontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,2);
plot(om2/pi*FS/2,m2,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-1 High Pass Filter Response (Order =
',num2str(N2),')'],'Fontsize',14);
ylabel('Magnitude','FontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,3);
plot(om3/pi*FS/2,m3,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-2 High Pass Filter Response (Order =
',num2str(N3),')'],'Fontsize',14);
ylabel('Magnitude','FontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,4);
plot(om3/pi*FS/2,m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title('Butterworth High Pass Filter Response for different Orders','Fontsize',14);
ylabel('Magnitude','FontSize',12,'fontweight','bold');
xlabel('Frequency(Hz)','fontsize',12,'fontweight','bold');
grid on;
hold on;
plot(om3/pi*FS/2,m2,'linewidth',2);
hold on;
plot(om3/pi*FS/2,m3,'linewidth',2);
legend(['Order = ',num2str(N1)'],['Order = ',num2str(N2)]',['Order =
',num2str(N3)]');
grid on;
```

Output:



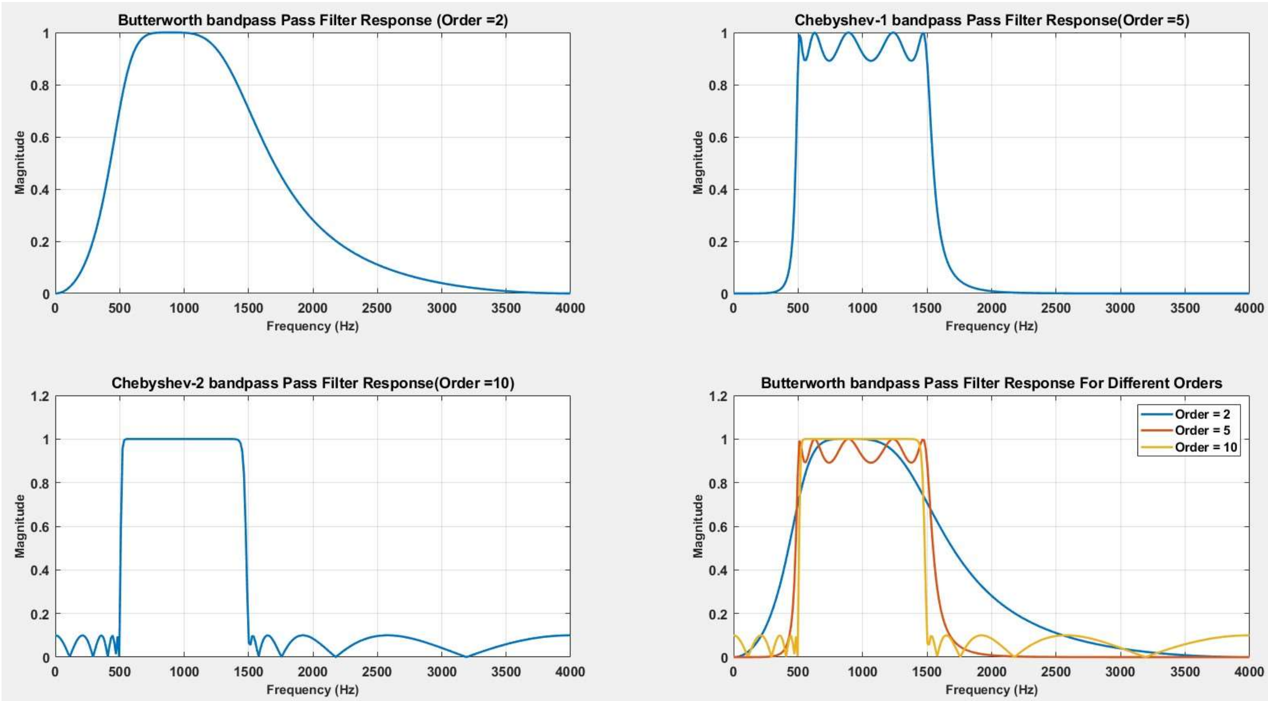
3. Band-pass filter:

Code:

```
FC1=1500;
FC2=500;
FS=8000;
N1=2;
N2=5;
N3=10;
Rp=1;
Rs=20;
wn=[2*FC1/FS 2*FC2/FS];
[b1,a1]=butter(N1,wn,'bandpass');
[b2,a2]=cheby1(N2,Rp,wn,'bandpass');
[b3,a3]=cheby2(N3,Rs,wn,'bandpass');
w=0:0.01:pi;

[h1,om1]=freqz(b1,a1,w);
m1=abs(h1);
[h2,om2]=freqz(b2,a2,w);
m2=abs(h2);
[h3,om3]=freqz(b3,a3,w);
```

```
m3=abs(h3);

subplot(2,2,1);
plot(om1/pi*FS/2,m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Butterworth bandpass Pass Filter Response (Order
=',num2str(N1),')'],'FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,2);
plot(om2/pi*FS/2,m2,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-1 bandpass Pass Filter Response(Order
=',num2str(N2),')'],'FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,3);
plot(om3/pi*FS/2,m3,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-2 bandpass Pass Filter Response(Order
=',num2str(N3),')'],'FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,4);
plot(om1/pi*FS/2,m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title('Butterworth bandpass Pass Filter Response For Different Orders','FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;
hold on;
plot(om3/pi*FS/2,m2,'linewidth',2);
hold on;
plot(om3/pi*FS/2,m3,'linewidth',2);
hold on;
legend(['Order = ',num2str(N1)]',['Order = ',num2str(N2)]',['Order = ',num2str(N3)]');
grid on;
```

Output:



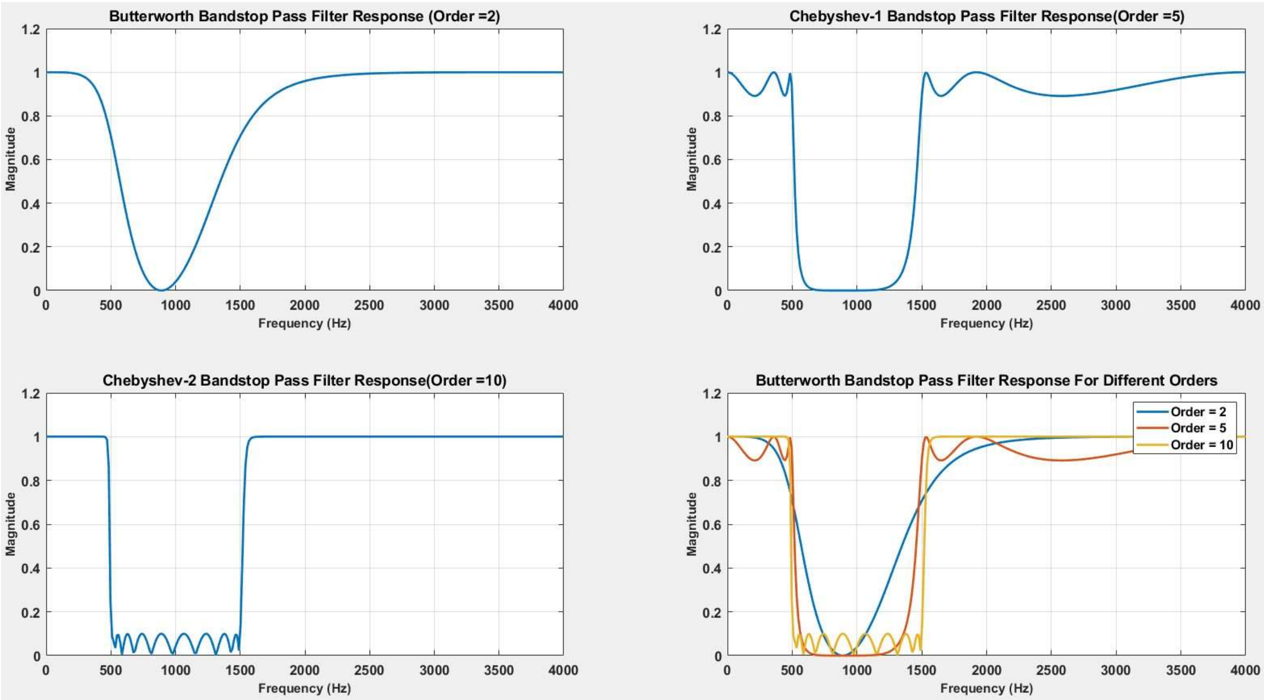
4. Band-stop filter:

Code:

```
FC1=1500;
FC2=500;
FS=8000;
N1=2;
N2=5;
N3=10;
Rp=1;
Rs=20;
wn=[2*FC1/FS 2*FC2/FS];
[b1,a1]=butter(N1,wn,'stop');
[b2,a2]=cheby1(N2,Rp,wn,'stop');
[b3,a3]=cheby2(N3,Rs,wn,'stop');
w=0:0.01:pi;

[h1,om1]=freqz(b1,a1,w);
m1=abs(h1);
[h2,om2]=freqz(b2,a2,w);
m2=abs(h2);
[h3,om3]=freqz(b3,a3,w);
```

```
m3=abs(h3);

subplot(2,2,1);
plot(om1/pi*FS/2,m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Butterworth Bandstop Pass Filter Response (Order
=',num2str(N1),')'],'FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,2);
plot(om2/pi*FS/2,m2,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-1 Bandstop Pass Filter Response(Order
=',num2str(N2),')'],'FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,3);
plot(om3/pi*FS/2,m3,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title(['Chebyshev-2 Bandstop Pass Filter Response(Order
=',num2str(N3),')'],'FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;

subplot(2,2,4);
plot(om1/pi*FS/2,m1,'linewidth',2);
set(gca,'fontsize',13,'fontweight','bold');
title('Butterworth Bandstop Pass Filter Response For Different Orders','FontSize',14);
ylabel('Magnitude','fontsize',12,'fontweight','bold');
xlabel('Frequency (Hz)','fontsize',12,'fontweight','bold');
grid on;
hold on;
plot(om3/pi*FS/2,m2,'linewidth',2);
hold on;
plot(om3/pi*FS/2,m3,'linewidth',2);
hold on;
legend(['Order = ',num2str(N1)]',['Order = ',num2str(N2)]',['Order = ',num2str(N3)]');
grid on;
```

Output:


Butterworth Bandstop Pass Filter Response (Order =2)
Chebyshev-1 Bandstop Pass Filter Response(Order =5)
Chebyshev-2 Bandstop Pass Filter Response(Order =10)
Butterworth Bandstop Pass Filter Response For Different Orders
Order = 2
Order = 5
Order = 10
Magnitude
Frequency (Hz)

Output Verification:

Exp No: 08 Date: 13-3-24

Experiment - 8 : Study of various IIR filter characteristics

I Low-pass:

```
[b1, a1] = butter(N1, 2*FC1/FS, 'low');
[b2, a2] = cheby1(N2, Rp, 2*FC1/FS, 'low');
[b3, a3] = cheby2(N3, Rs, 2*FC1/FS, 'low');
w = 0:0.01:pi;
```

II High-pass:

```
[b1, a1] = butter(N1, 2*FC1/FS, 'high');
[b2, a2] = cheby1(N2, Rp, 2*FC1/FS, 'high');
[b3, a3] = cheby2(N3, RS, 2*FC1/FS, 'high');
w = 0:0.01:pi;
```

8 13/3/24

III Band-pass:

```
wn = [2* FC1/FS   2* FC2/FS]
[b1, a1] = butter(N1, wn, 'bandpass');
[b2, a2] = cheby1(N2, Rp, wn, 'bandpass');
[b3, a3] = cheby2(N3, RS, wn, 'bandpass');
w = 0:0.01:pi;
```

IV Band-Stop:

```
wn = [2* FC1/FS   2* FC2/FS]
[b1, a1] = butter(N1, wn, 'stop');
[b2, a2] = cheby1(N2, Rp, wn, 'stop');
[b3, a3] = cheby2(N3, Rs, wn, 'stop');
w = 0:0.01:pi;
```

Software Experiment – 11: Design of various FIR filter using windowing method

Aim:

➢ To design various FIR filters using different windowing techniques-Rectangular, Hamming and Hanning.

Software required:

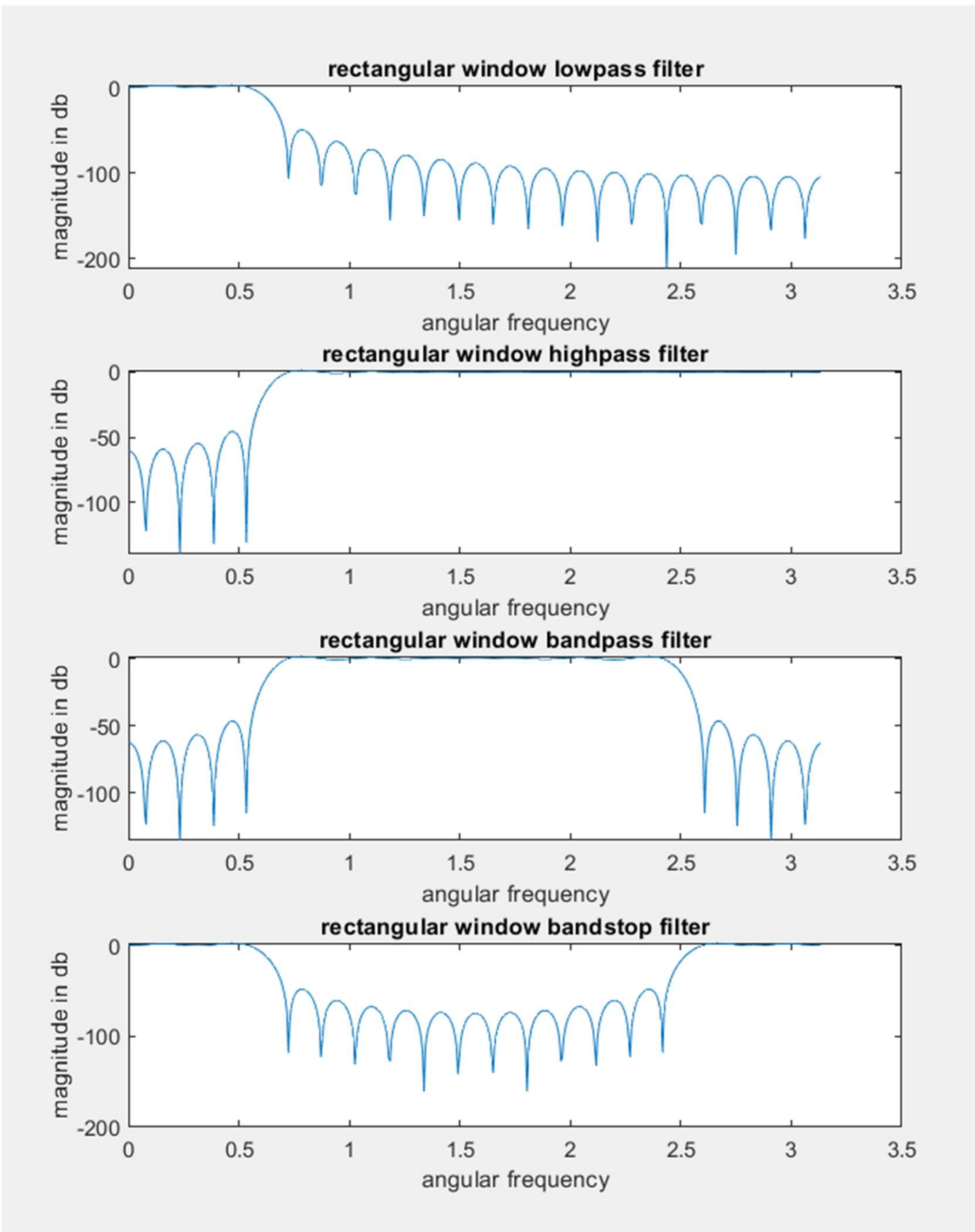
MATLAB

Code & output:

1. **Rectangular window:**

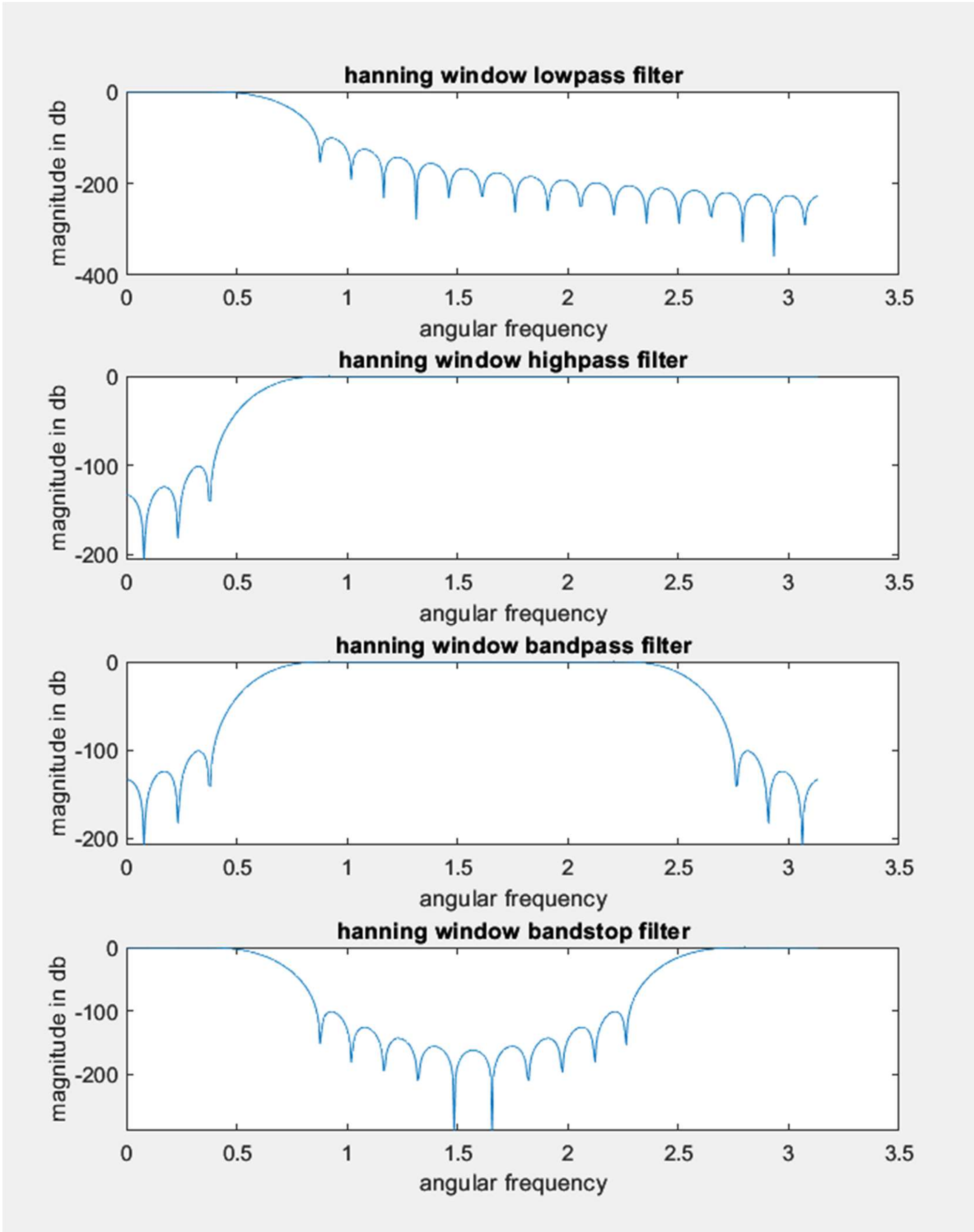
```
w1=0.2*pi;w2=0.8*pi;n=40
f1=w1/pi;
f2=w2/pi;

subplot(4,1,1)
b=fir1(n,f1,'low',rectwin(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('rectangular window lowpass filter')

subplot(4,1,2)
b=fir1(n,f1,'high',rectwin(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('rectangular window highpass filter')

subplot(4,1,3)
b=fir1(n,[f1,f2],'bandpass',rectwin(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
```

```
ylabel('magnitude in db')
title('rectangular window bandpass filter')

subplot(4,1,4)
b=fir1(n,[f1,f2],'stop',rectwin(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('rectangular window bandstop filter')
```



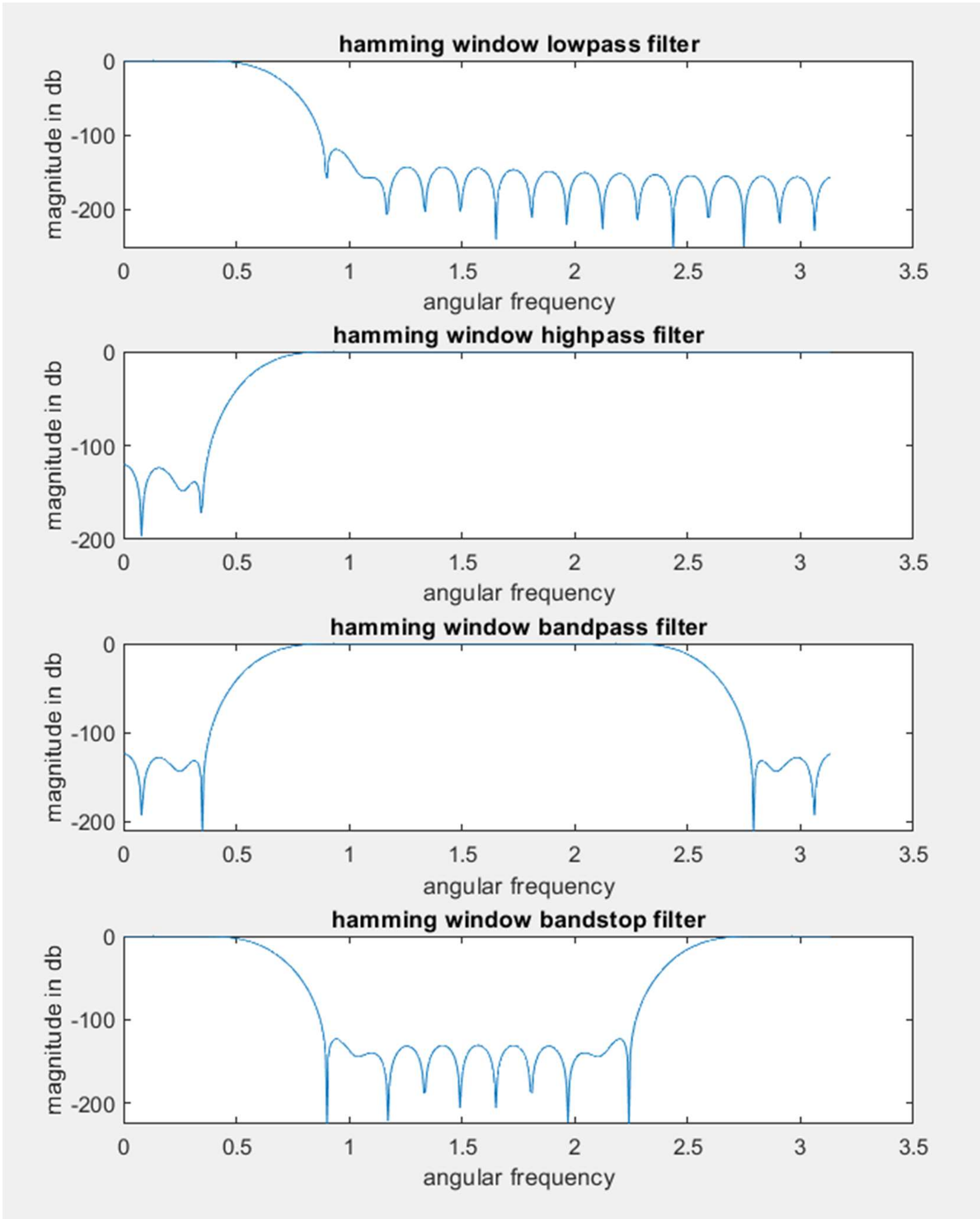
2. Hanning window:

```
w1=0.2*pi;w2=0.8*pi;n=40
f1=w1/pi;
f2=w2/pi;

subplot(4,1,1)
b=fir1(n,f1,'low',hanning(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hanning window lowpass filter')

subplot(4,1,2)
b=fir1(n,f1,'high',hanning(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hanning window highpass filter')

subplot(4,1,3)
b=fir1(n,[f1,f2],'bandpass',hanning(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hanning window bandpass filter')

subplot(4,1,4)
b=fir1(n,[f1,f2],'stop',hanning(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hanning window bandstop filter')
```


hanning window lowpass filter
magnitude in db
0
-200
-400
0
0.5
1
1.5
2
2.5
3
3.5
angular frequency
hanning window highpass filter
magnitude in db
0
-100
-200
angular frequency
hanning window bandpass filter
magnitude in db
0
-100
-200
angular frequency
hanning window bandstop filter
magnitude in db
0
-100
-200
angular frequency

1. **Hamming window:**

```
w1=0.2*pi;w2=0.8*pi;n=40
f1=w1/pi;
f2=w2/pi;

subplot(4,1,1)
b=fir1(n,f1,'low',hamming(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hamming window lowpass filter')

subplot(4,1,2)
b=fir1(n,f1,'high',hamming(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hamming window highpass filter')

subplot(4,1,3)
b=fir1(n,[f1,f2],'bandpass',hamming(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hamming window bandpass filter')

subplot(4,1,4)
b=fir1(n,[f1,f2],'stop',hamming(n+1));
[h,w]=freqz(b,1,512);
mag=20*log(abs(h));
plot(w,mag);
xlabel('angular frequency')
ylabel('magnitude in db')
title('hamming window bandstop filter')
```


hamming window lowpass filter
magnitude in db
0
-100
-200
0
0.5
1
1.5
2
2.5
3
3.5
angular frequency
hamming window highpass filter
magnitude in db
0
-100
-200
0
0.5
1
1.5
2
2.5
3
3.5
angular frequency
hamming window bandpass filter
magnitude in db
0
-100
-200
0
0.5
1
1.5
2
2.5
3
3.5
angular frequency
hamming window bandstop filter
magnitude in db
0
-100
-200
0
0.5
1
1.5
2
2.5
3
3.5
angular frequency

Output Verification:

Experiment 11 - FIR filters

firfm

filter

boxcar - rectangular window < Hamming / Hanning { BPF, BSF, LPF, HPF

fs = 100

f1 = 10

f2 = 20 f3 = 30

N = 30 fc1 = 15 fc2 = 25

t = 0 : 1/fs : 10

x = 2 * sin (2 * pi * f1 * t) + 3 * cos (2 * pi * f2 * t)

b = fir1 (N, [fc1 / (fs/2)], 'low', boxcar (N+1))

freqz (b, 1, N)

xf = filter (b, 0.1, x);

figure;

plot (t, xf)

17/04/24