P.E.S. UNIVERSITY
Department of Electronics and Communication Engineering
Session: Jan - May 2025

PRINCIPLES OF DIGITAL SIGNAL PROCESSING

EXPERIMENT No. 1: Verification of Sampling theorem

Aim: To verify Sampling theorem for a signal of given frequency.

Theory: Sampling is a process of converting a continuous time signal (analog signal) x(t) into a discrete time signal x[n], which is represented as a sequence of numbers. (A/D converter)Converting back x[n] into analog (resulting in the process of reconstruction.) (D/A converter)
For the reconstructed signal to be exactly the same as x(t), sampling theorem is used to get x(n) from x(t). The sampling frequency fs determines the spacing between samples.
Aliasing - A high frequency signal is converted to a lower frequency, results due to under sampling. Though it is undesirable in ADCs, it finds practical applications in stroboscope and sampling oscilloscopes.

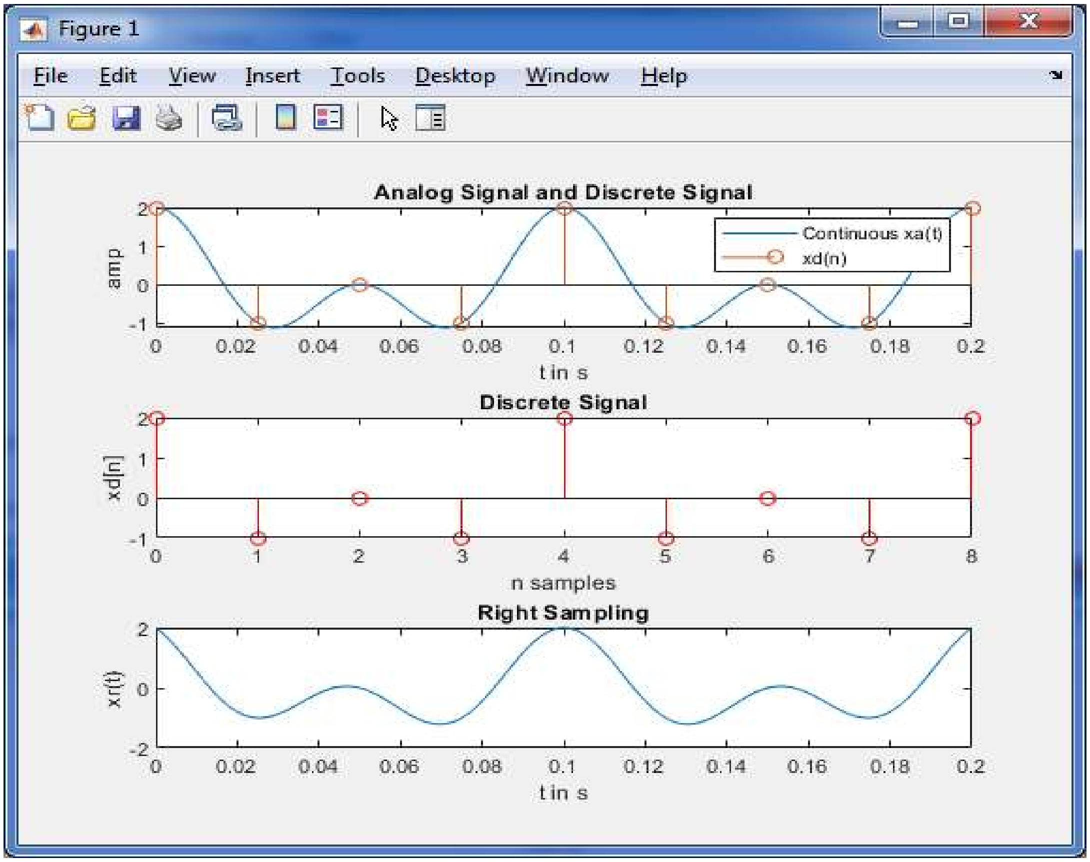
Algorithm:
1. Input the desired frequency f1,f2 and generate y (for which sampling theorem is to be verified).
2. Input the Sampling Frequency fs.
3. Generate oversampled, Nyquist & under sampled discrete time signals.
4. Plot the waveforms and hence prove sampling theorem.

```
%% Program
clc;  % clears the command window
clear all; %clears variables in workspace
close all;  %close all fig windows
t=0:0.001:0.2; %analog time axis

f1=input ('Enter the input frequency1  = ');
f2=input ('Enter the input frequency2  = ');

xa=cos(2*pi*f1*t)+cos(2*pi*f2*t);

fm=max(f1,f2);
% For Right sampling fs=2*fm
% For Over sampling fs>2*fm so let fs=4*fm
% For Under sampling fs<2*fm so let fs=fm
fs=2*fm;                          %fs = sampling freequency
ts=1/fs;

n=0:1:(0.2*fs);
xd=cos(2*pi*f1*(n*ts))+cos(2*pi*f2*(n*ts));
figure;
subplot(3,1,1);
plot(t,xa);
xlabel('t in s');
```

```
ylabel('amp');
title('Analog Signal and Discrete Signal');
hold on;
stem(n*ts,xd);
hold off;
legend('Continuous xa(t)','xd(n)');
%% Plot only Discrete signal
subplot(3,1,2);
stem(n,xd,'r');
xlabel('n samples');
ylabel('xd[n]');
title('Discrete Signal');
%% Reconstruction by using the formula:
% xr(t) = sum over n=0,...,N-1: x(nT)*sin(pi*(t-nT)/T)/(pi*(t-
nT)/T)
% Note that sin(pi*(t-nT)/T)/(pi*(t-nT)/T) = sinc((t-nT)/T)
% sinc(x) = sin(pi*x)/(pi*x) according to MATLAB
xr=xd*sinc((t-(n'*ts))/ts);
subplot(3,1,3);
plot(t,xr);
xlabel('t in s');
ylabel('xr(t)');
title('Right Sampling'); %or /Over Sampling/ Under Sampling
%%%%%%%%%%%%%%%%%%%%
```

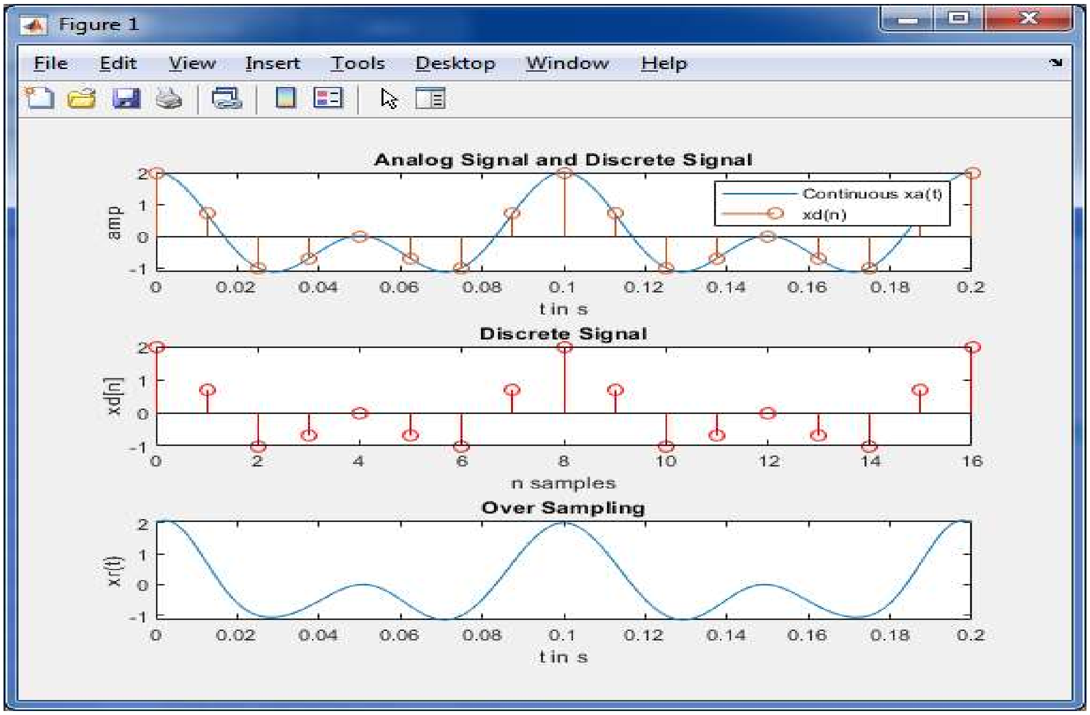
OUTPUT:

Right Sampling:
Enter the input frequency1 = 10
Enter the input frequency2 = 20



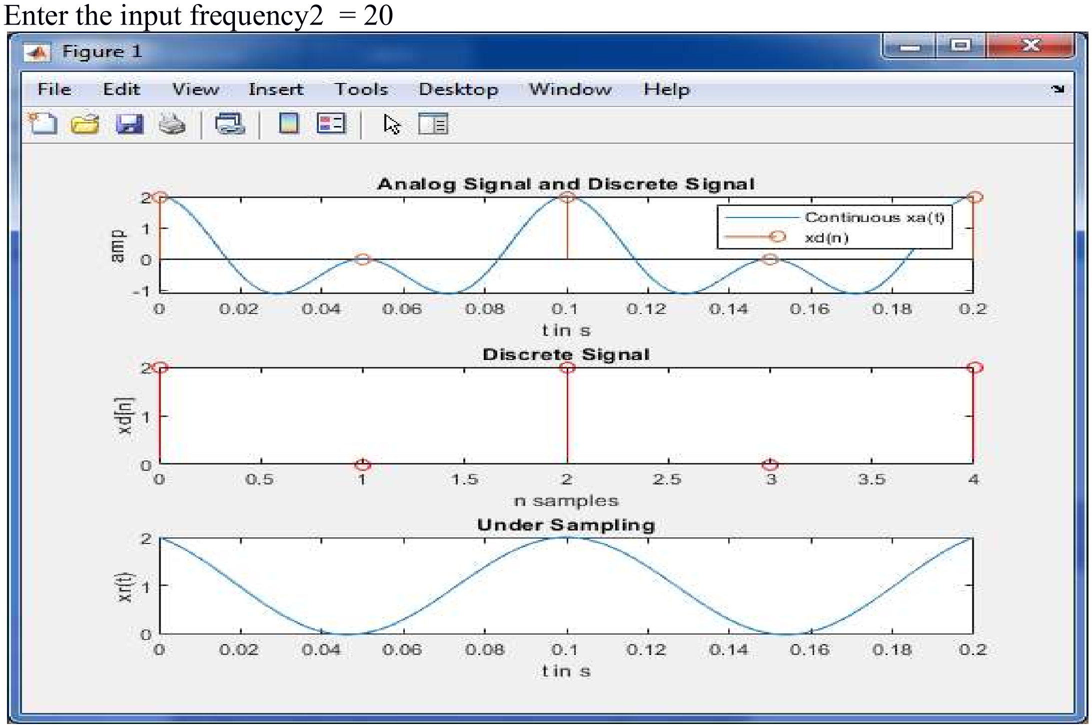
Over Sampling:

Enter the input frequency1 = 10
Enter the input frequency2 = 20



Under Sampling:

Enter the input frequency1 = 10
Enter the input frequency2 = 20



EXPERIMENT No.2 : Program to solve difference equation and program to find the Impulse response of the system

WITHOUT INBUILT FUNCTION

```
%y(n)-0.25y(n-1)-0.125y(n-2)=x(n)+0.5x(n-1)
%x(n)=u(n)-u(n-2)
%y(-1)=1 and y(-2)=2  (enter the values as [2 1])

clc; %clears the console window
clear all;%deletes the user defined variable in variable browser
close all;%close the figure window

%a=[1 -0.25 -0.125]
%b=[1 0.5]
b=input('Enter the coefficients of x: ');
a=input('Enter the coefficients of y: ');

M=length(b)-1;
N=length(a)-1;
IC=input('Enter the initial conditions for y: ');

n=[-N:20];%number of terms

%x[n]=u[n]-u[n-2]
x=[(n>=0)]-[(n>=2)];
subplot(211);
stem(n,x);

title('input sequence x[n]');
xlabel('n');
ylabel('x[n]');

y=[IC zeros(1,length(n)-N)];


for n=N+1:20   %loop runs length(n) times to find y(n)
      sumx=0;sumy=0;

      for k=0:M
            sumx=sumx+(b(k+1)*x(n-k));
      end
      for k=1:N
            sumy=sumy+(a(k+1)*y(n-k));
      end
      y(n)=sumx-sumy;
end

n=[-N:20];%number of terms

subplot(212);
stem(n,y);

title('output sequence y[n]');
```

```
xlabel('n');
ylabel('y[n]');
disp('y[n]=');
disp(y)
```

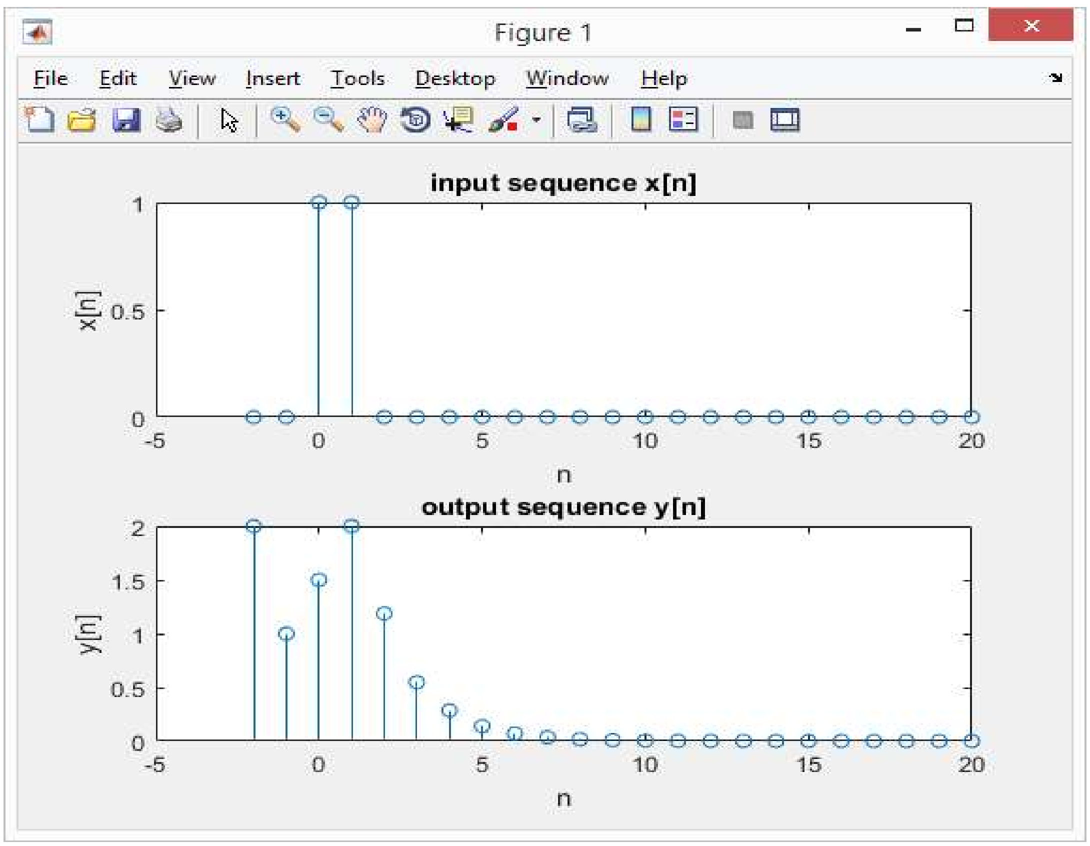
OUTPUT:

Enter the coefficients of x: [1 0.5]
Enter the coefficients of y: [1 -0.25 -0.125]
Enter the initial conditions for y: [2 1]
y[n]=
Columns 1 through 13

2.0000 1.0000 1.5000 2.0000 1.1875 0.5469 0.2852 0.1396 0.0706 0.0351 0.0176 0.0088 0.0044

Columns 14 through 23

0.0022 0.0011 0.0005 0.0003 0.0001 0.0001 0.0000 0 0 0



WITH INBUILT FUNCTION

```
%y(n)-0.25y(n-1)-0.125y(n-2)=x(n)+0.5x(n-1)
%x(n)=u(n)-u(n-2)
```

```
%y(-1)=1 and y(-2)=2 , (enter the values as [2 1])

clc; %clears the console window
clear all;%deletes the user defined variable in variable browser
close all;%close the figure window

%a=[1 -0.25 -0.125]
%b=[1 0.5]
b=input('Enter the coefficients of x: ');
a=input('Enter the coefficients of y: ');

M=length(b)-1;
N=length(a)-1;
IC=input('Enter the initial conditions for y: ');

%Initial conditions for transposed direct-form II filter
ic=filtic(b,a,flip(IC));
n=[0:20];%number of terms

%x[n]=u[n]-u[n-2]
x=[(n>=0)]-[(n>=2)];
y=filter(b, a, x, ic);

subplot(211);
stem(n,x);

title('input sequence x[n]');
xlabel('n');
ylabel('x[n]');

subplot(212);
stem(n,y);

title('output sequence y[n]');
xlabel('n');
ylabel('y[n]');
disp('y[n]=');
disp(y)
```

OUTPUT:

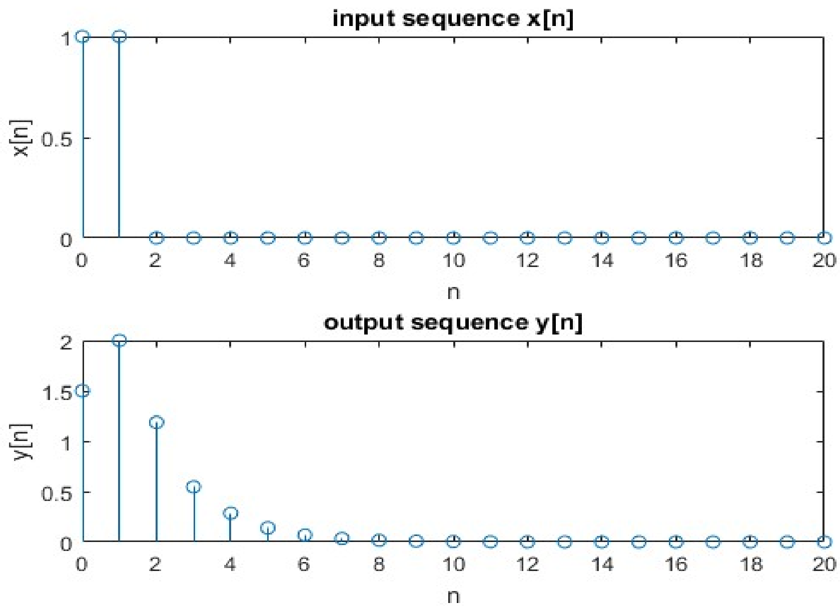
Enter the coefficients of x: [1 0.5]
Enter the coefficients of y: [1 -0.25 -0.125]
Enter the initial conditions for y: [2 1]

y[n]=
Columns 1 through 13

1.5000 2.0000 1.1875 0.5469 0.2852 0.1396 0.0706 0.0351 0.0176 0.0088 0.0044 0.0022 0.0011

Columns 14 through 21

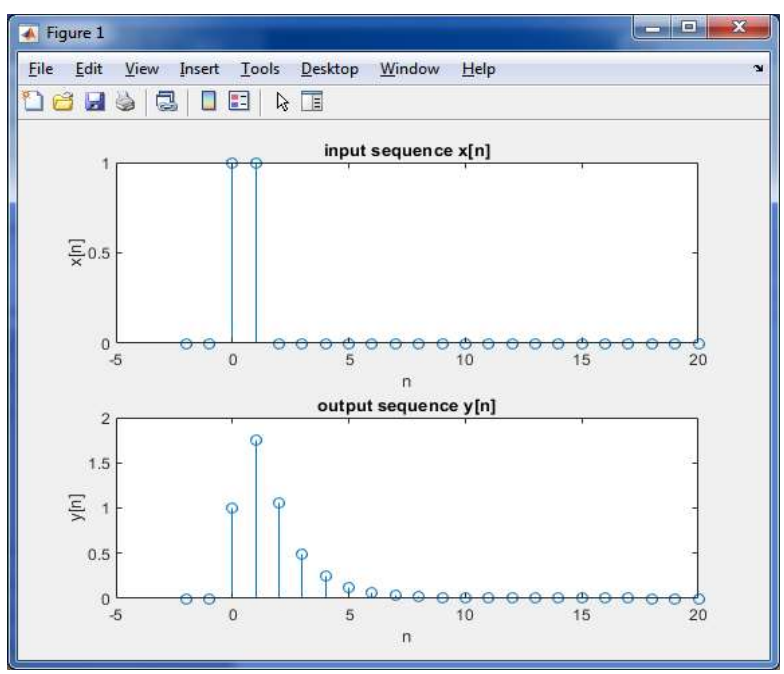
0.0005   0.0003   0.0001   0.0001   0.0000   0.0000   0.0000   0.0000



With zero initial conditions

Enter the coefficients of x: [1 0.5]
Enter the coefficients of y: [1 -0.25 -0.125]
Enter the initial conditions for y: [0 0]

y[n]	yinbuilt[n]
0	0
0	0
1.0000	1.0000
1.7500	1.7500
1.0625	1.0625
0.4844	0.4844
0.2539	0.2539
0.1240	0.1240
0.0627	0.0627
0.0312	0.0312
0.0156	0.0156
0.0078	0.0078
0.0039	0.0039
0.0020	0.0020
0.0010	0.0010
0.0005	0.0005
0.0002	0.0002
0.0001	0.0001
0.0001	0.0001



To find impulse response

```
x=double([n==0]);
```

IC =[0 0];

Impulse Response Output:

Enter the coefficients of x: [1 0.5]
Enter the coefficients of y: [1 -0.25 -0.125]
Enter the initial conditions for y: [0 0]

y[n]	yinbuilt[n]
0	0
0	0
1.0000	1.0000
0.7500	0.7500
0.3125	0.3125
0.1719	0.1719
0.0820	0.0820
0.0420	0.0420
0.0208	0.0208
0.0104	0.0104
0.0052	0.0052
0.0026	0.0026
0.0013	0.0013
0.0007	0.0007
0.0003	0.0003
0.0002	0.0002
0.0001	0.0001

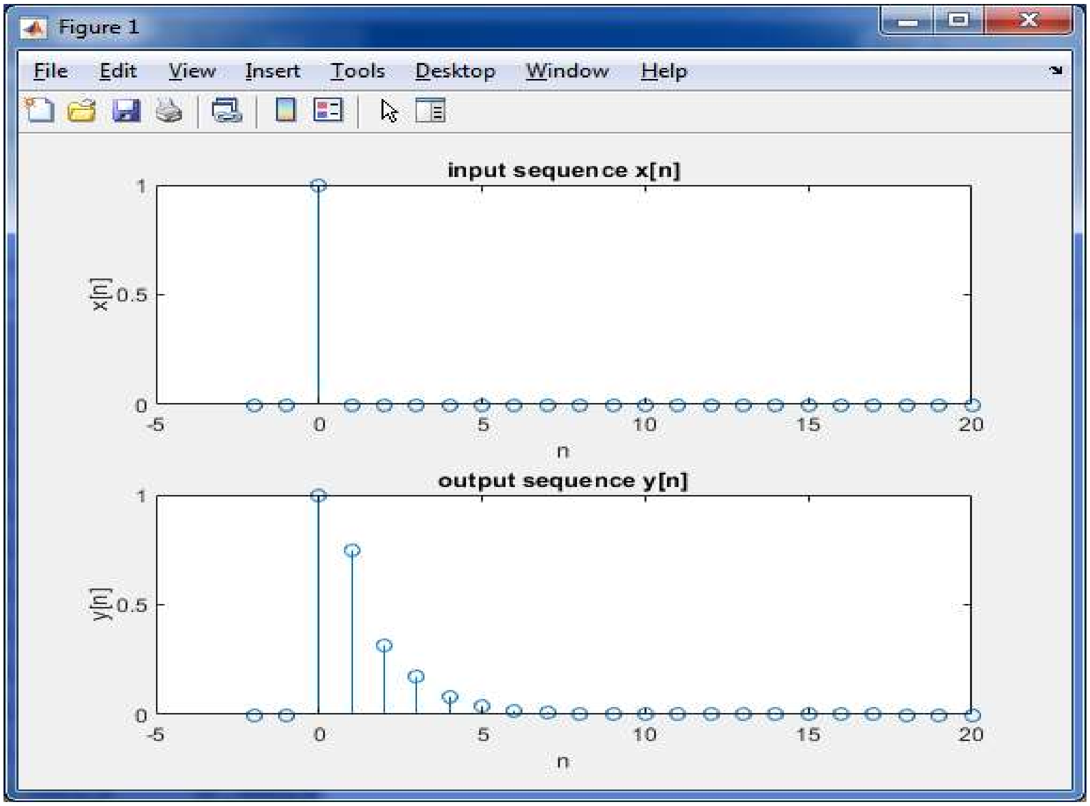


Figure 1
File Edit View Insert Tools Desktop Window Help
input sequence x[n]
x[n]
n
output sequence y[n]
y[n]
n

EXPERIMENT No. 3: (a) Computation of N point DFT of a given sequence x(n) and to plot magnitude and phase spectrum (b) Computation N-pt IDFT of the given X(k).

(a) Computation of N point DFT of a given sequence x(n) and to plot magnitude and phase spectrum

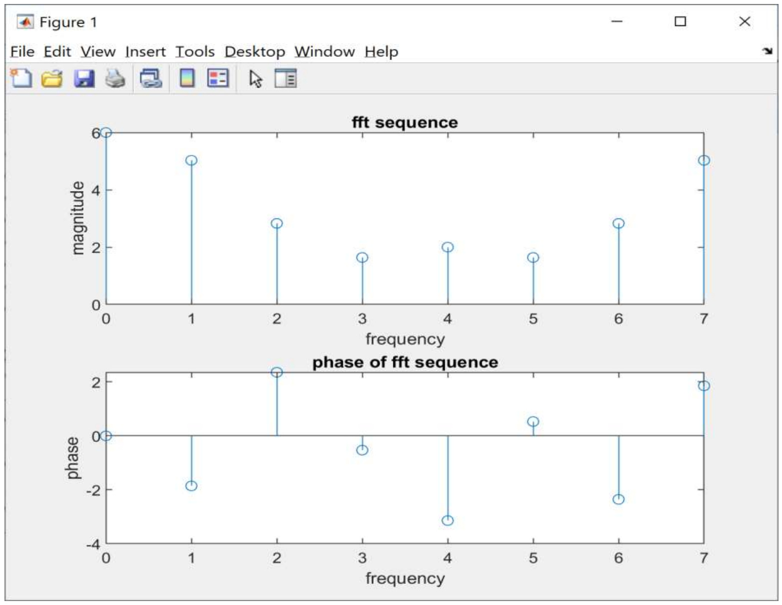
```
%%Program to find dft
clc;
clear all;
close all;
%%input sequence
xn=input('enter the input sequence: ');
N=input('enter the number of points: ');
Xk=calcdft(xn,N);

disp('DFT X(K)= ');
disp(Xk);
%%magnitude of dft
magXk=abs(Xk);
%%phase of dft
phaseXk=angle(Xk);
k=0:N-1;
subplot(2,1,1);
stem(k,magXk);
title('fft sequence');
xlabel('frequency');
ylabel('magnitude');
subplot(2,1,2);
stem(k,phaseXk);
title('phase of fft sequence');
xlabel('frequency');
ylabel('phase');

function [Xk]=calcdft(xn, N)
 L=length(xn);
if(N<L)
error('N must be >=L');
end
x1=[xn zeros(1,N-L)];
for k=0:1:N-1;
for n=0:1:N-1;
p=exp(-i*2*pi*n*k/N);
T(k+1,n+1)=p;
end
end
disp('Transformation Matrix for dft')
disp(T);
```

```
Xk=T*x1.';
end
```

OUTPUT:

```
enter the input sequence: [0 1 2 3]
enter the number of points: 8

x1 =

     0     1     2     3     0     0     0     0

Transformation Matrix for dft
  Columns 1 through 4

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 + 1.0000i   0.7071 - 0.7071i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 - 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i  -0.0000 + 1.0000i  -0.7071 + 0.7071i

  Columns 5 through 8

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
  -1.0000 - 0.0000i  -0.7071 + 0.7071i  -0.0000 + 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
  -1.0000 - 0.0000i   0.7071 + 0.7071i   0.0000 - 1.0000i  -0.7071 + 0.7071i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
  -1.0000 - 0.0000i   0.7071 - 0.7071i  -0.0000 + 1.0000i  -0.7071 - 0.7071i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i  -0.0000 - 1.0000i
  -1.0000 - 0.0000i  -0.7071 - 0.7071i  -0.0000 - 1.0000i   0.7071 - 0.7071i
```

DFT X(K)=
6.0000 + 0.0000i
-1.4142 - 4.8284i
-2.0000 + 2.0000i
1.4142 - 0.8284i
-2.0000 - 0.0000i
1.4142 + 0.8284i
-2.0000 - 2.0000i
-1.4142 + 4.8284i



(b) Computation N-pt IDFT of the given X(k).

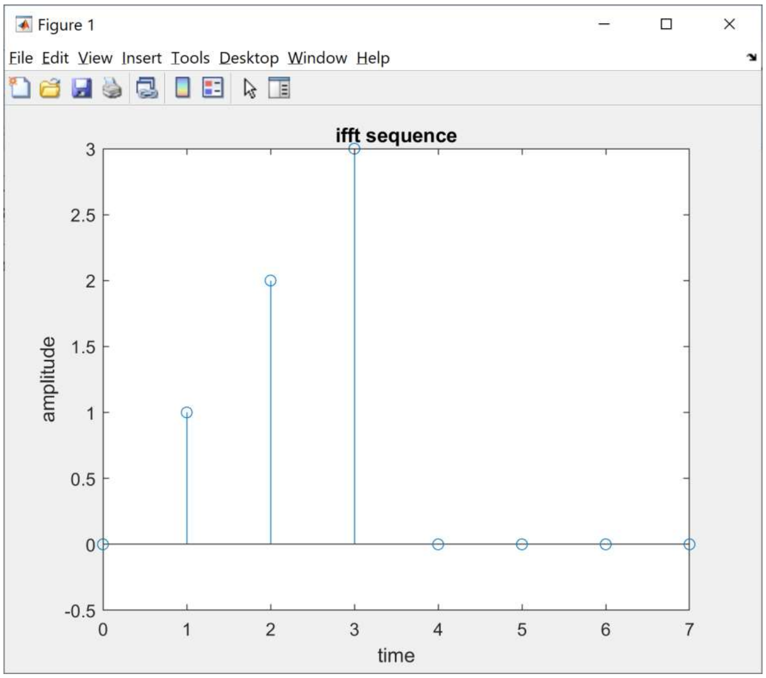
```
%%Program to find idft
clc;
clear all;
close all;

%%input sequence
Xk=input('enter the input sequence');
xn=calcidft(Xk);

N=length(xn);

disp('x(n)=');
disp(xn);
n=0:N-1;
stem(n,xn);
title('ifft sequence');
xlabel('time');
ylabel('amplitude');



%%Xk should be row vector
function [xn]=calcidft(Xk)
N=length(Xk);
for k=0:1:N-1;
for n=0:1:N-1;
p=exp(i*2*pi*n*k/N);
IT(k+1,n+1)=p;
```

```
end
end
disp('Transformation Matrix for idft')
disp(IT);
xn=(IT*(Xk.'))./N;
end
```

OUTPUT:

```
enter the input sequence[6.0000 + 0.0000i  -1.4142 - 4.8284i  -2.0000 + 2.0000i   1.4142 - 0.8284i -
2.0000 - 0.0000i   1.4142 + 0.8284i  -2.0000 - 2.0000i  -1.4142 + 4.8284i]
Transformation Matrix for idft
  Columns 1 through 4

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i   0.0000 + 1.0000i  -0.7071 + 0.7071i
   1.0000 + 0.0000i   0.0000 + 1.0000i  -1.0000 + 0.0000i  -0.0000 - 1.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i  -0.0000 - 1.0000i   0.7071 + 0.7071i
   1.0000 + 0.0000i  -1.0000 + 0.0000i   1.0000 - 0.0000i  -1.0000 + 0.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i   0.0000 + 1.0000i   0.7071 - 0.7071i
   1.0000 + 0.0000i  -0.0000 - 1.0000i  -1.0000 + 0.0000i   0.0000 + 1.0000i
   1.0000 + 0.0000i   0.7071 - 0.7071i  -0.0000 - 1.0000i  -0.7071 - 0.7071i

  Columns 5 through 8

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
  -1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 - 1.0000i   0.7071 - 0.7071i
   1.0000 - 0.0000i   0.0000 + 1.0000i  -1.0000 + 0.0000i  -0.0000 - 1.0000i
  -1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 + 1.0000i  -0.7071 - 0.7071i
   1.0000 - 0.0000i  -1.0000 + 0.0000i   1.0000 - 0.0000i  -1.0000 + 0.0000i
  -1.0000 + 0.0000i   0.7071 + 0.7071i  -0.0000 - 1.0000i  -0.7071 + 0.7071i
   1.0000 - 0.0000i  -0.0000 - 1.0000i  -1.0000 + 0.0000i  -0.0000 + 1.0000i
  -1.0000 + 0.0000i  -0.7071 + 0.7071i  -0.0000 + 1.0000i   0.7071 + 0.7071i

x(n)=
   0.0000 + 0.0000i
   1.0000 - 0.0000i
   2.0000 - 0.0000i
   3.0000 - 0.0000i
  -0.0000 + 0.0000i
   0.0000 + 0.0000i
  -0.0000 - 0.0000i
   0.0000 + 0.0000i

Warning: Using only the real component of complex data.
```

```
> In getRealData (line 52)
  In stem (line 40)
  In prog5b (line 15)
>>
```



EXPERIMENT No. 4: **Convolution of two sequences in time and frequency domain.**

a) **Linear convolution**

```
%%This program gives the Linear convolution of two sequences.

%% Main Program
clc; %clears the console window
clear all;%deletes the user defined variable in variable browser
close all;%close the figure window

x=input('Enter the 1st sequence:');
nx=input('Enter the time index sequence:');
h=input('Enter the 2nd sequence:');
nh=input('Enter the time index sequence:');
[y,ny]=findconv(x,nx,h,nh);
figure;
subplot(3,1,1);
stem(nx,x);
xlabel('Time');
ylabel('Amplitude');
title('1st sequence');
subplot(3,1,2);
stem(nh,h);
xlabel('Time');
ylabel('Amplitude');
title('2nd sequence');
subplot(3,1,3);
stem(ny,y);
xlabel('Time');
ylabel('Amplitude');
title('Linear convolution');
disp(y);
disp(ny);
%%function
function [y,ny]=findconv(x,nx,h,nh)
nybegin=nx(1)+nh(1);
nyend=nx(length(nx))+nh(length(nh));
ny=nybegin:nyend;
%y=conv(x,h); %calling inbuilt function
 y=calcconv(x,h)
end

function [y]=calcconv(x,h)
l1=length(x);
l2=length(h);
N = l1+l2-1; %length of linear convolution
%-----------------------------------------------------------
%y=linear convolution of x[n] and h[n]
%note: in matlab index starts with 1 and not 0
for n=1:1:N
y(n)=0;
for k=1:1:l1
if(n-k+1>=1 & n-k+1<=l2) % to avoid negative index
y(n)=y(n)+x(k)*h(n-k+1);
end
end
end
end
```

OUTPUT :

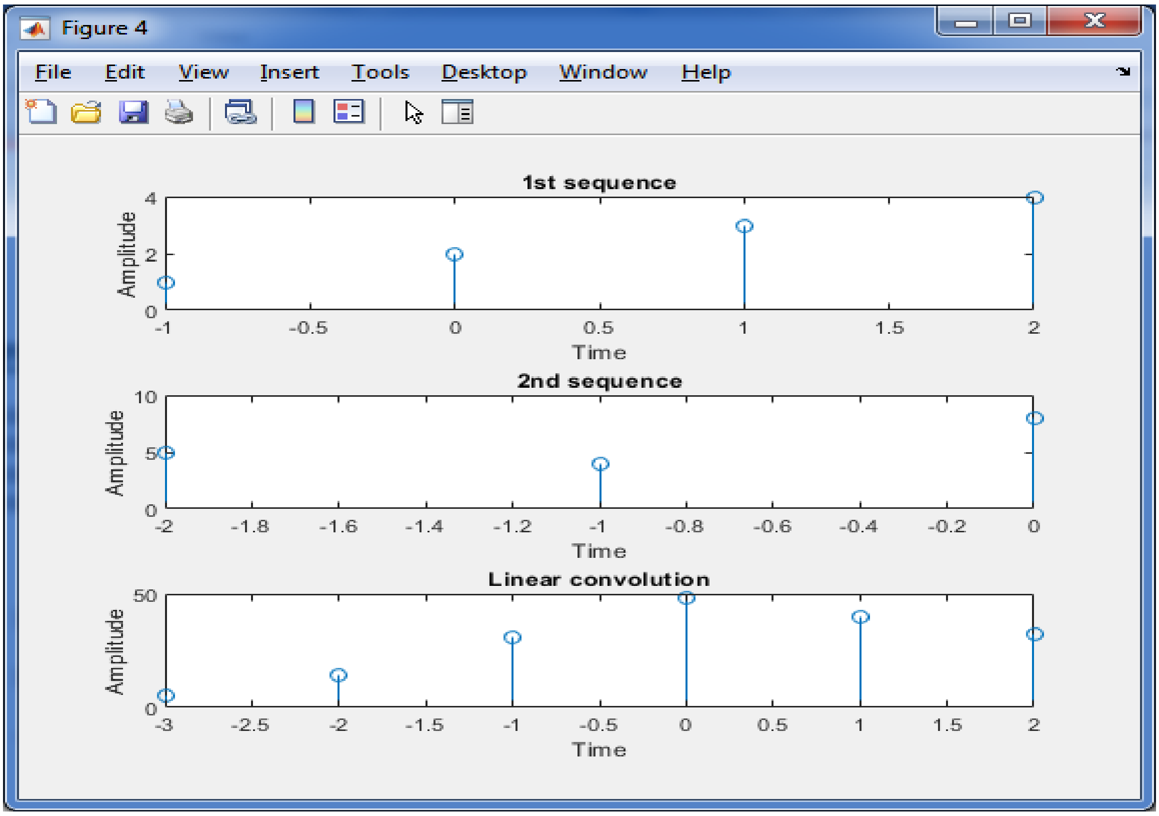
Enter the 1st sequence:[1 2 3 4]
Enter the time index sequence:[-1 0 1 2]
Enter the 2nd sequence:[5 4 8]
Enter the time index sequence:[-2 -1 0]

y =

    5   14   31   48   40   32

    5   14   31   48   40   32

   -3   -2   -1   0   1   2



b) Circular convolution

```
%%This program gives the Circular convolution of two sequences.

%Circular convolution without using inbuilt function
clc; %clears the console window
clear all;%deletes the user defined variable in variable browser
close all;%close the figure window
x= input('Enter x[n]: '); %read input x[n]
h= input('Enter h[n]: '); % read impulse response h[n]
N= input('Enter N: ');
l1=length(x);
l2=length(h);

x=[x zeros(1,(N-l1))]
h=[h zeros(1,(N-l2))]

%inbuilt function
%y=cconv(x,h,N);

%-----------------------------------------------------------
%y=Circular convolution of x[n] and h[n]
%note: in matlab index starts with 1 and not 0
for n=1:1:N
y(n)=0;
for k=1:1:N

y(n)=y(n)+x(k)*h(mod((n-k),N)+1);
end
end
%end
%-----------------------------------------------------------
disp('The circular convolution of two given sequence');

n=0:N-1;
figure;
subplot(3,1,1);
stem(n,x);
xlabel('Time');
ylabel('Amplitude');
title('1st sequence');
subplot(3,1,2);
stem(n,h);
xlabel('Time');
ylabel('Amplitude');
title('2nd sequence');
subplot(3,1,3);
stem(n,y);
xlabel('Time');
ylabel('Amplitude');
title('circular convolution');
disp(y);
disp(n);
```

OUTPUT:

Enter x[n]: [1 2 3 4]
Enter h[n]: [1 2 2 1]

Enter N: 4

x =

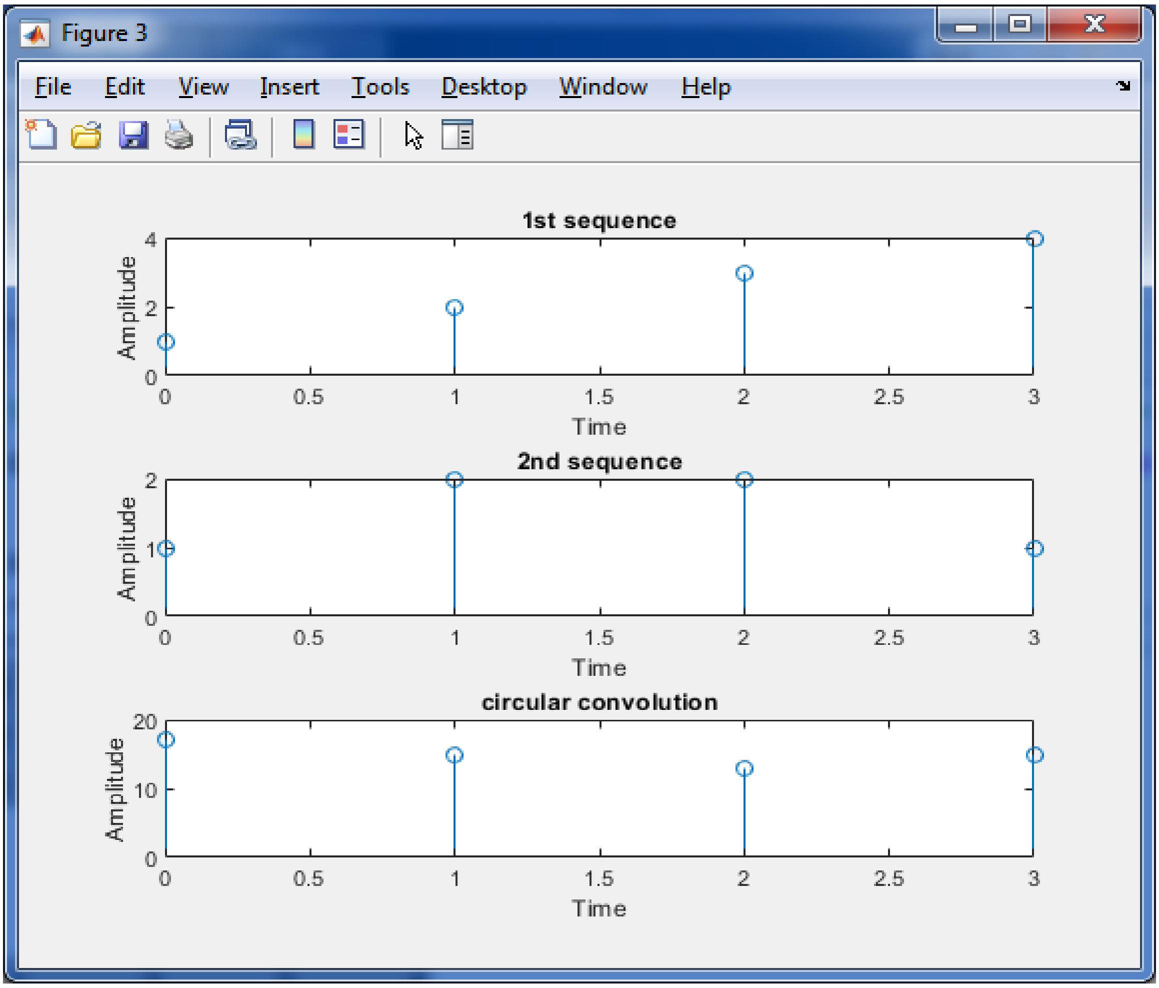
1 2 3 4

h =

1 2 2 1

The circular convolution of two given sequence
17 15 13 15

0 1 2 3



c) Linear convolution of two sequences using DFT and IDFT.

```
%% Program to find linear convolution using dft and idft
clear all;
close all;
clc;
x=input('Enter the 1st Sequence: ');
h=input('Enter the 2nd Sequence: ');
N=length(x)+length(h)-1;
X=fft(x,N);
H=fft(h,N);
Y=X.*H;
y=calcidft(Y);
figure;
subplot(3,1,1);
stem(0:length(x)-1,x);
xlabel('Time');
ylabel('Amplitude');
title('1st seq');
subplot(3,1,2);
stem(0:length(h)-1, h);
xlabel('Time');
ylabel('Amplitude');
title('2nd seq');
subplot(3,1,3);
stem(0:N-1,real(y));
xlabel('Time');
ylabel('Amplitude');
title('Linear convolution');
disp('Linear convolution');
disp(real(y));
```

OUTPUT:

Enter the 1st Sequence: [1 2 3 4]

Enter the 2nd Sequence: [3 2 4]

Linear convolution

3.0000

8.0000

17.0000

26.0000

20.0000

16.0000

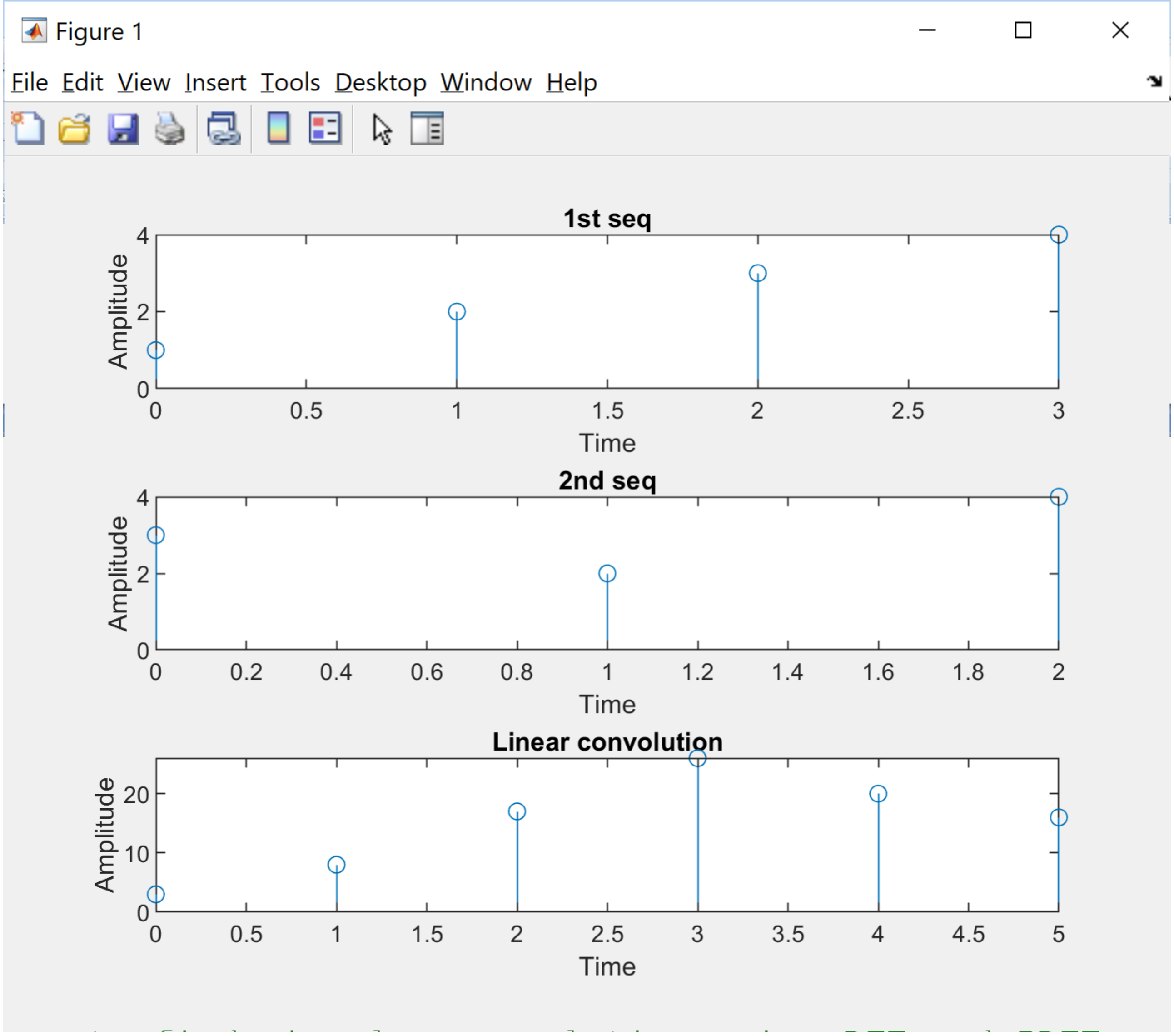


Figure 1
File Edit View Insert Tools Desktop Window Help
1st seq
Amplitude
Time
2nd seq
Amplitude
Time
Linear convolution
Amplitude
Time

d) Circular convolution of two given sequences using DFT and IDFT.

```
%% Program to find circular convolution using DFT and IDFT
clear all;
close all;
clc;
x=input('Enter the 1st Sequence: ');
h=input('Enter the 2nd Sequence: ');
N=max(length(x), length(h));
X=calcdft(x,N);
H=calcdft(h,N);
Y=X.*H;
y=ifft(Y,N);
figure;
subplot(3,1,1);
stem(0:length(x)-1,x);
xlabel('Time');
ylabel('Amplitude');
title('1st seq');
subplot(3,1,2);
stem(0:length(h)-1, h);
xlabel('Time');
ylabel('Amplitude');
title('2nd seq');
subplot(3,1,3);
stem(0:N-1,real(y));
xlabel('Time');
ylabel('Amplitude');
title('circular convolution');
disp('circular convolution');
disp(real(y));
```

OUTPUT:

Enter the 1st Sequence: [1 2 3 4]

Enter the 2nd Sequence: [1 2 2 1]

Circular convolution

17

15

13

15

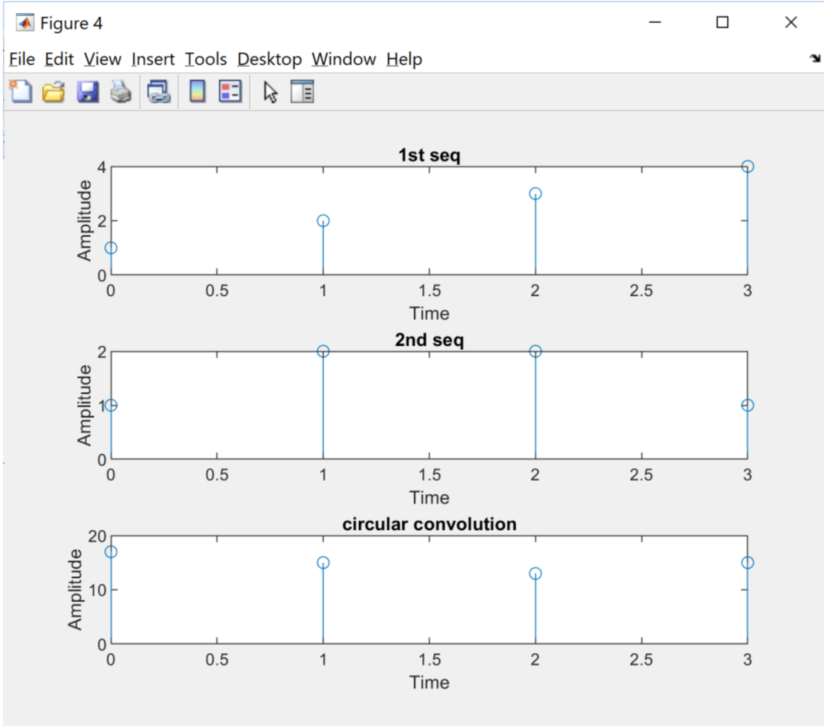


Figure 4
File Edit View Insert Tools Desktop Window Help
1st seq
Amplitude
Time
2nd seq
Amplitude
Time
circular convolution
Amplitude
Time

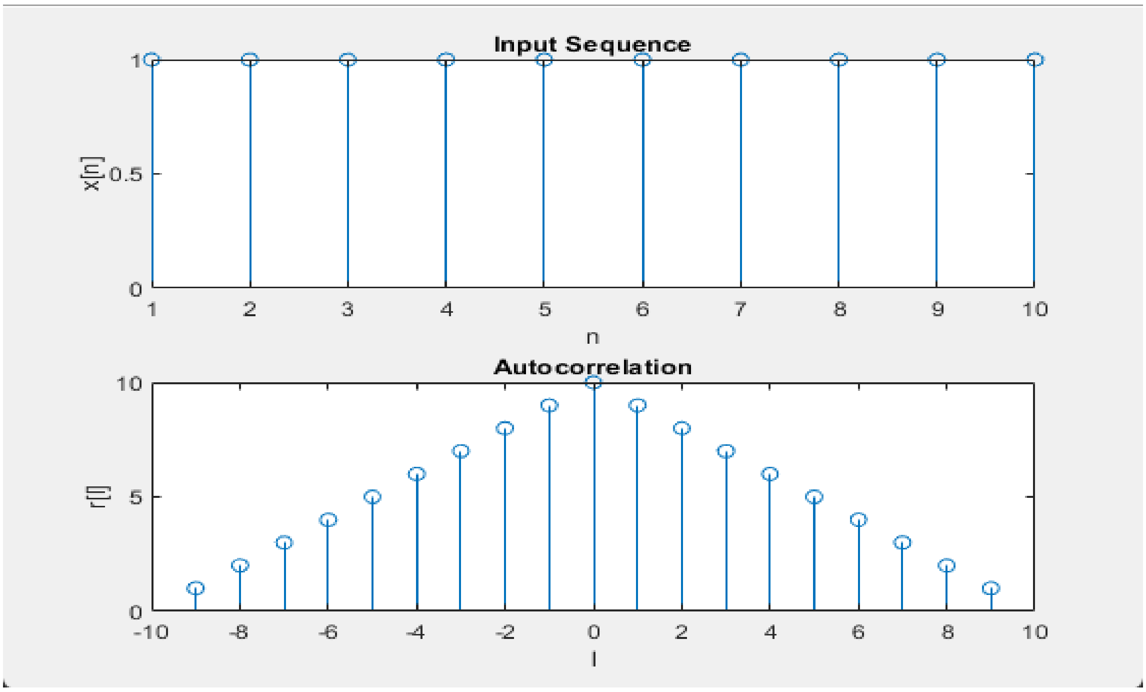
EXPERIMENT No.5.(a) Autocorrelation of a given sequence.
(b)Cross correlation of given sequences

(a) Autocorrelation of a given sequence

```
clc; % Clear the console window
clear all; % Delete user-defined variables
close all; % Close figure window

% Input sequence and index
x = input('Enter the sequence x= ');
nx = input('Enter the index for sequence nx= ');

% Compute autocorrelation
[rxx, nrxx] = findconv(x, nx, flip(x), -flip(nx));

% Display results
disp('Autocorrelation sequence:');
disp(rxx);
disp('Corresponding indices:');
disp(nrxx);

% Plot the input sequence and its autocorrelation
subplot(2,1,1);
stem(nx, x);
xlabel('n');
ylabel('x[n]');
title('Input Sequence');

subplot(2,1,2);
stem(nrxx, rxx);
xlabel('l');
ylabel('r[l]');
title('Autocorrelation');

% Function for convolution-based correlation
function [y, ny] = findconv(x, nx, h, nh)
    nyb = nx(1) + nh(1);
    nye = nx(length(nx)) + nh(length(nh));
    ny = nyb:nye;
    y = conv(x, h);
end
```

OUTPUT:

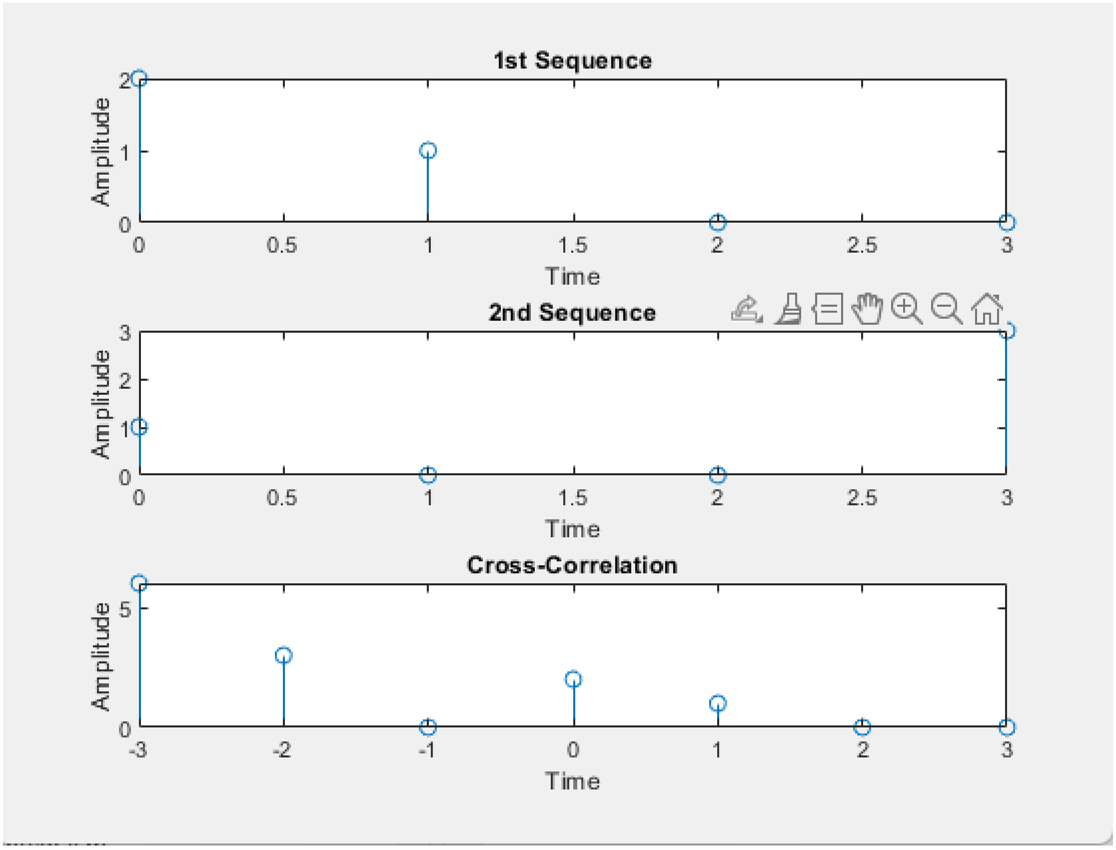
```
Enter the sequence x= [1 1 1 1 1 1 1 1 1 1]
Enter the index for sequence nx= [1:10]
Autocorrelation sequence:
  Columns 1 through 7

     1     2     3     4     5     6     7

  Columns 8 through 14

     8     9    10     9     8     7     6

  Columns 15 through 19

     5     4     3     2     1

Corresponding indices:
  Columns 1 through 7

    -9    -8    -7    -6    -5    -4    -3

  Columns 8 through 14

    -2    -1     0     1     2     3     4

  Columns 15 through 19

     5     6     7     8     9
```



```
clc; % Clear the console window
clear all; % Delete user-defined variables
close all; % Close figure window

% Input sequences and indices
x = input('Enter the 1st sequence x= ');
nx = input('Enter the index for sequence nx= ');
y = input('Enter the 2nd sequence y= ');
ny = input('Enter the index for sequence ny= ');

% Compute cross-correlation
[rxy, nrxy] = findconv(x, nx, flip(y), -flip(ny));

% Display results
disp('Cross-correlation sequence:');
disp(rxy);
disp('Corresponding indices:');
disp(nrxy);

% Plot the input sequences and their cross-correlation
subplot(3,1,1);
stem(nx, x);
xlabel('Time');
ylabel('Amplitude');
title('1st Sequence');

subplot(3,1,2);
stem(ny, y);
xlabel('Time');
ylabel('Amplitude');
title('2nd Sequence');

subplot(3,1,3);
stem(nrxy, rxy);
xlabel('Time');
ylabel('Amplitude');
title('Cross-Correlation');

% Function for convolution-based correlation
function [y, ny] = findconv(x, nx, h, nh)
    nyb = nx(1) + nh(1);
    nye = nx(length(nx)) + nh(length(nh));
    ny = nyb:nye;
    y = conv(x, h);
end
```

OUTPUT:

```
Enter the 1st sequence x= [2 1 0 0]
Enter the index for sequence nx= [0:3]
Enter the 2nd sequence y= [1 0 0 3]
Enter the index for sequence ny= [0:3]
Cross-correlation sequence:
     6     3     0     2     1     0     0

Corresponding indices:
    -3    -2    -1     0     1     2     3
```



P.E.S. UNIVERSITY

Department of Electronics and Communication Engineering

Session: Jan - May 2025

PRINCIPLE OF DIGITAL SIGNAL PROCESSING

Procedure for Non-Real Time experiments

- Double click on **Code Composer Studio 6.2.0**, then click on **OK**
- Go to **View** > **Target Configuration** , click on it
- Then come to **CCS Edit** mode > right click on **User Defined >** then select **New Target Configuration**.
- Then give name as My_Target.ccxml
- Then a window by the name **Basic** appears.
- Then in the **Connection : Texas Instruments XDS100v3 USB Debug Probe** select it.
- Then in the **Board or Device > LCDKC6748**
- Then under Save Configuration **Save**
- Now under Test Configuration **Test Connection**
  At this moment, your DSP board should be turned on.
  Then a window appears, in that window at the end:
  "The JTAG DR Intergrity scan-test has succeeded" appears.
- Now make sure you are in **CCS Edit** mode. Under User Defined tap select **My_Target.ccxml** and right click and select **Set as Default**. Then, again right click on **My_Target.ccxml** and click on
  Launch Selected Configuration .
- Then come to **CCS Edit** mode.
- Then click on **Project** > **New CCS Project**. A window appears. In that window:

Target: **C674x Floating-point DSP** **LCDKC6748**

Connection **Texas Instruments XDS100V3 USB Debug Probe** > **Verify**

Project name: **India**

Then click on **Verify**.

Then a message "The JTAG DR Integrity scan-test has succeeded" appears
Then click on **Finish**

- Now do the following settings:
  New ccs project > CCS Project
  Create a new CCS Project
  Target: c674x Floating-point DSP LCDKC6748
  Connection: Texas Instruments XDS100V3 USBDebug Probe
  Project name : india
  Compiler version : TI v8.2.2

  Advanced settings {expand the dropdown arrow}
  Output type : Executable
  Output format: eabi (ELF)
  Device endianness: little
  Linker command file: c6748.cmd
  Runtime support library: rts64plus_elf.lib

  Project templates and examples
  Empty project (with main.c)

  Then click on **Finish**

- Then in project browser > in main.c type

  ```
  #include<stdio.h>
  int main(void){
          printf("\n HELLO INDIA");
          return 0
  ```

- Go to Project > Build All
- Go to CCS Debug Mode at the right corner
- Go to RUN > Debug
- Go to Run > Resume. Then a message, HELLO INDIA appears on the console
- To stop got to Run > Terminate.
- To come to console there is an  icon on the right, click on it

- To display the Graph go to

Tools > Graph > Single Time

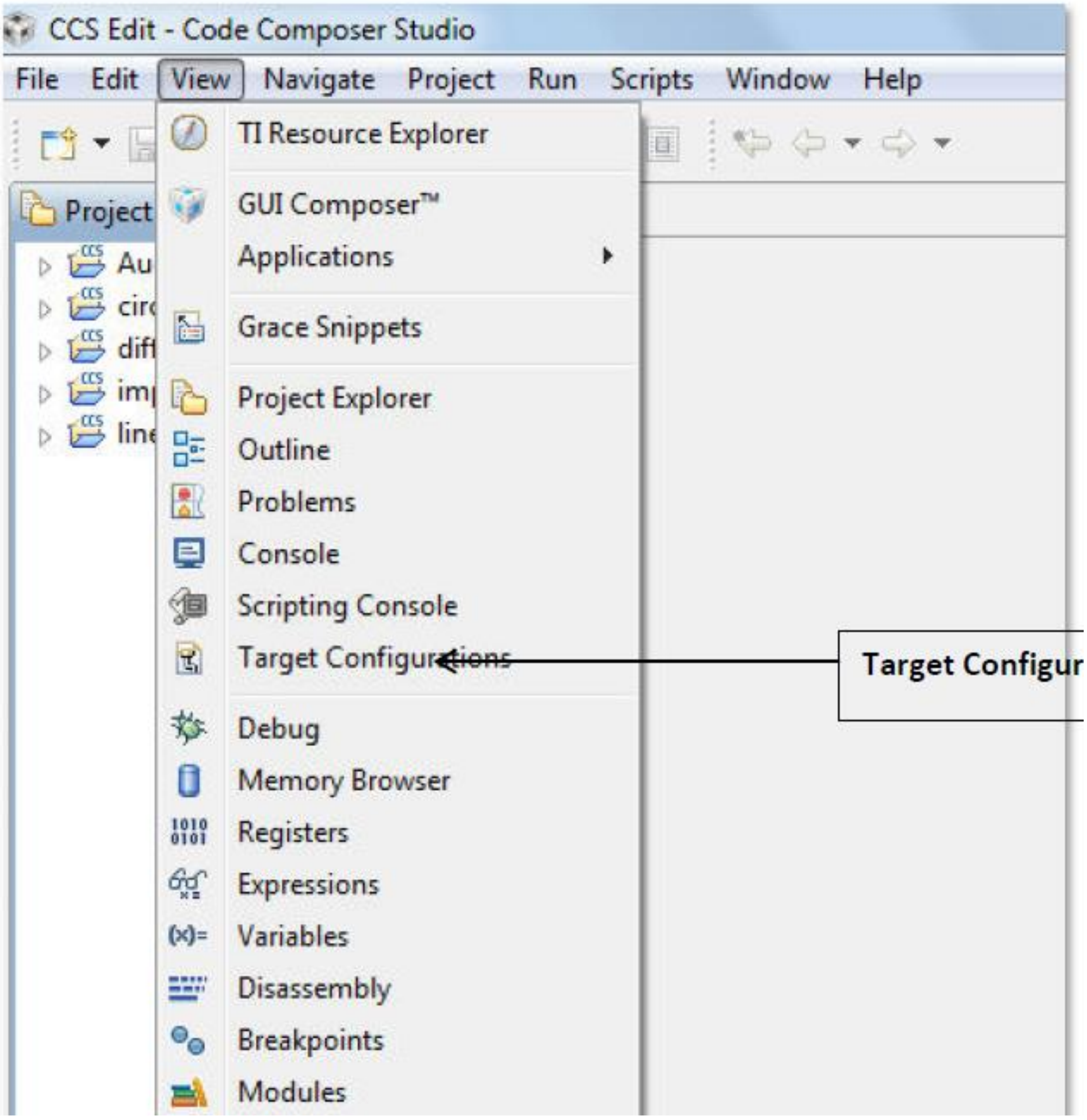
Data Properties

Acquisition Buffer Size	10
DSP Data type	32-bit floating point
Index increment	1
Sampling rate(in Hz)	1
Start Address	y { As in impulse response program O/P is stored in variable "y"
Axius display	☑ True
Data plot size	Bar
Display Data size	10
Grid Style	Major Grid
Magnitude Display Scale	Linear
Use Dc value for Graph	☐ False

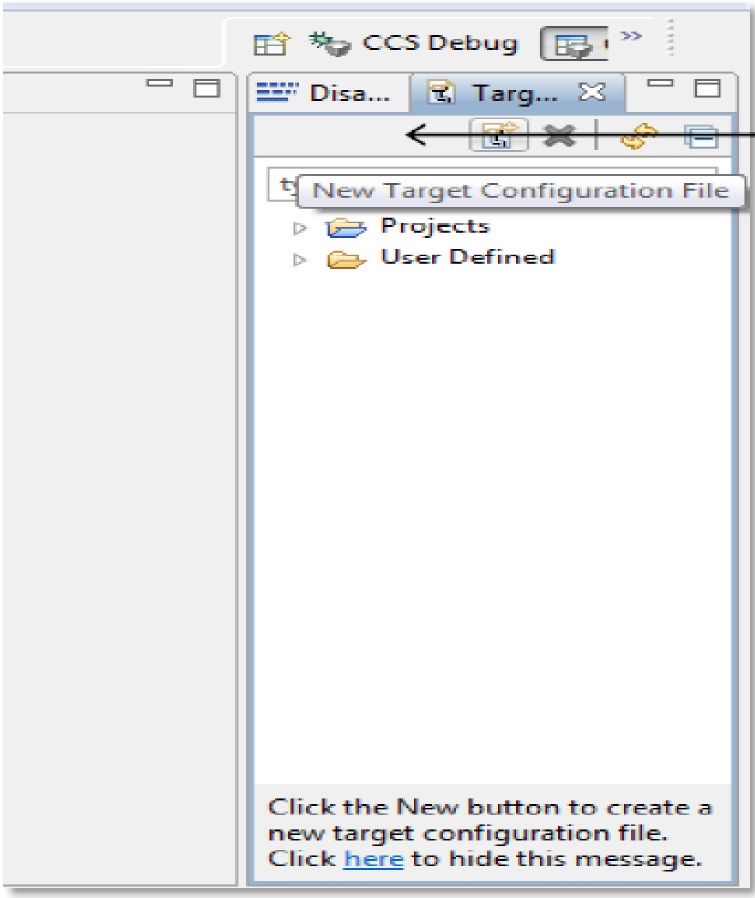
PRINCIPLES OF DIGITAL SIGNAL PROCESSING LABORATORY
Subject Code: UE22EC252B

PROCEDURE TO USE Code Composer Studio
It is assumed that the user has LAUNCHED THE CODE COMPOSER STUDIO according to the procedure mentioned above. Once the project explorer window appears, the user has to set the configurations for the target device (LCDK C6748).
Setting Target Configuration:

Go to View->Target Configurations



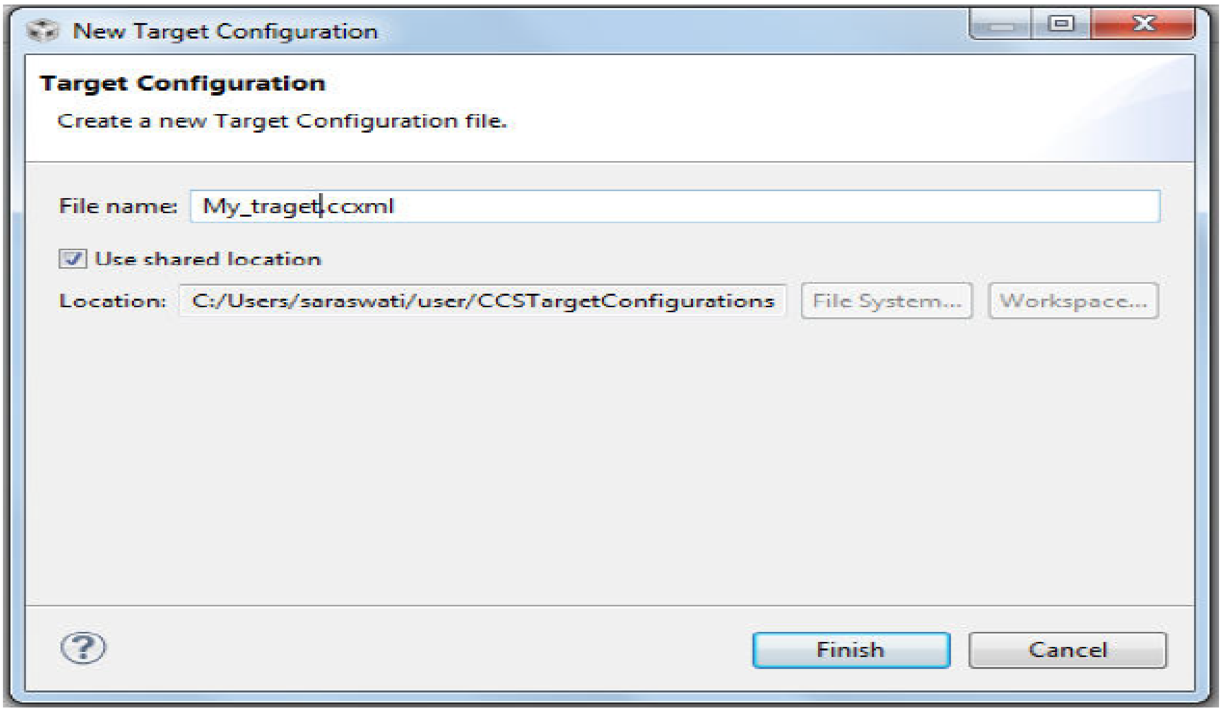
On the Right Hand Side of your project window, you can notice a target configuration space. Click on the new target configuration file as shown in the screen shot.



New Target Configuration File

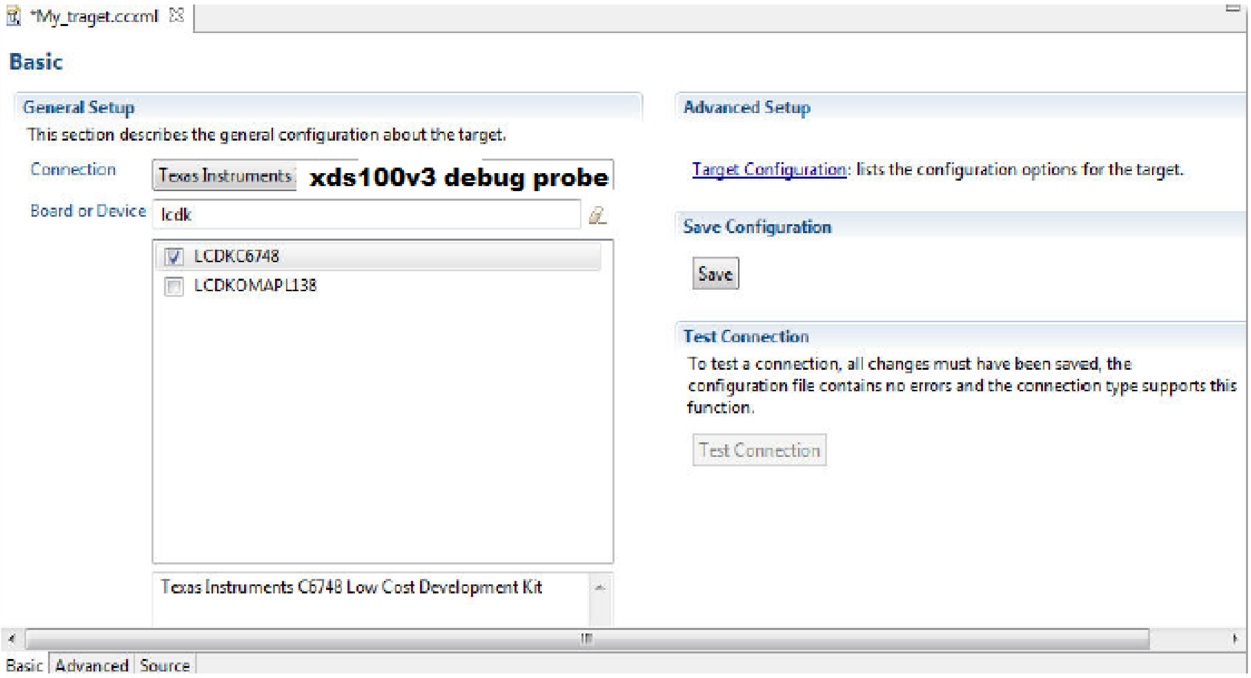
Fig: Target Configuration Space

- Give a suitable name to your target device(Here its named as My_target.ccxml)



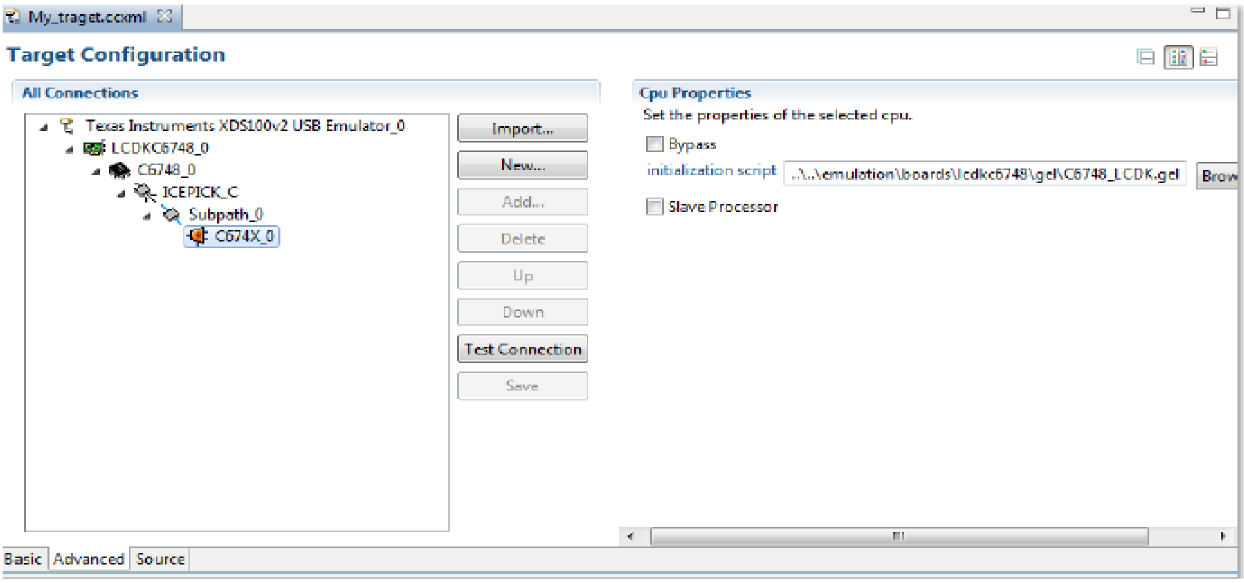
Click on Finish.

The Connection should be “Texas Instruments XDS100V3 Debug Probe”. The Board or Device should be LCDKC6748.



・ Click on Save and then Test Connection. If your connection is OK, then a message as “JTAGDR
integrity test succeeded” will appear.

- Click on the "Advanced" tab and then Click on C674x_0, verify the GEL file path.



Without the proper GEL file, hardware will not be connected to CCSv5.
Close "Target Configuration" window and Right Click on My_target and select SET AS DEFAULT and then LAUNCH SELECTED CONFIGURATION.

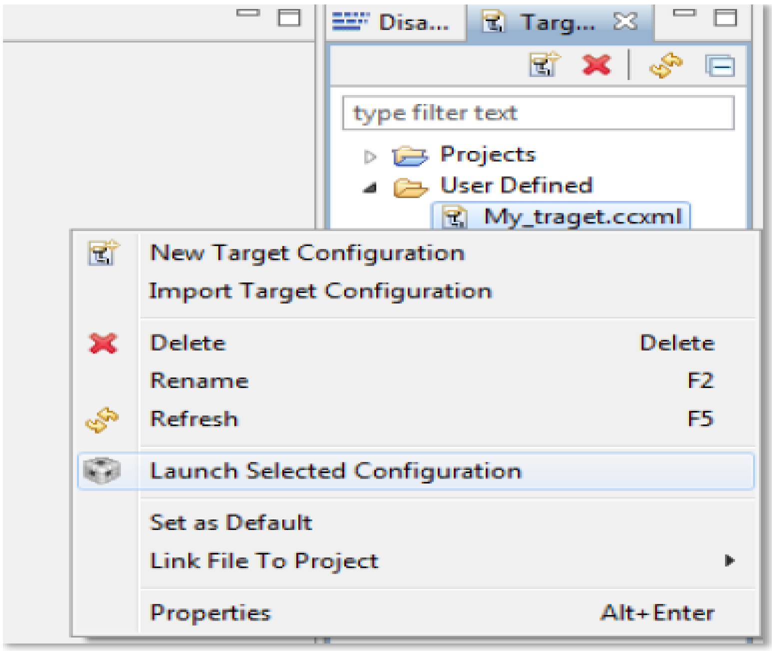


Fig: Launching Target Configuration

- Debug window will be launched.

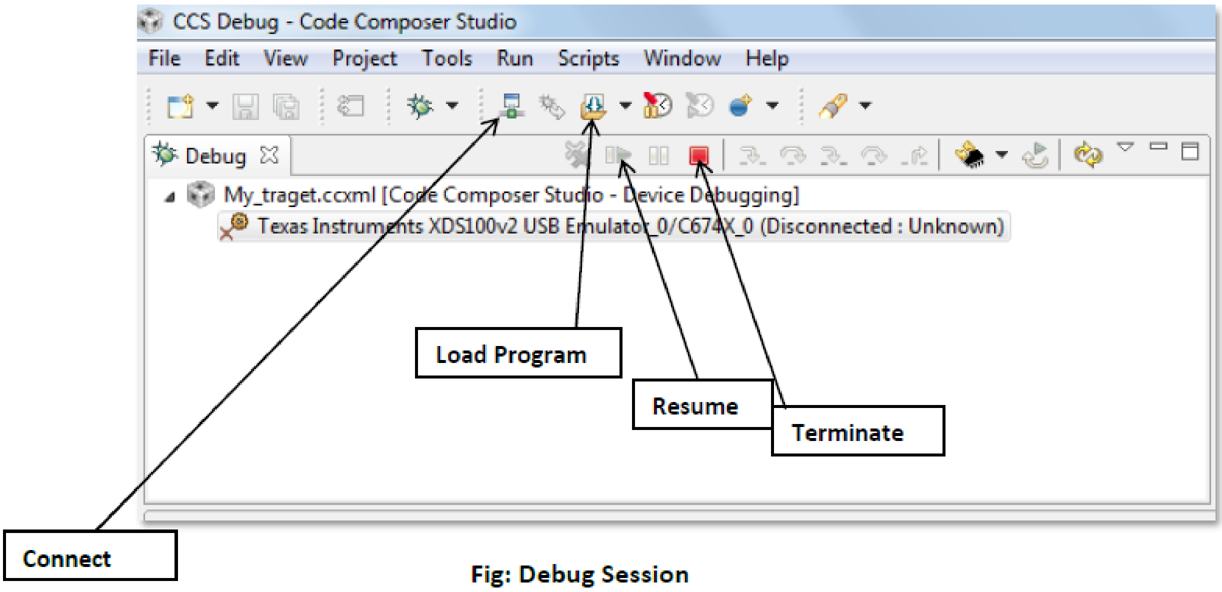


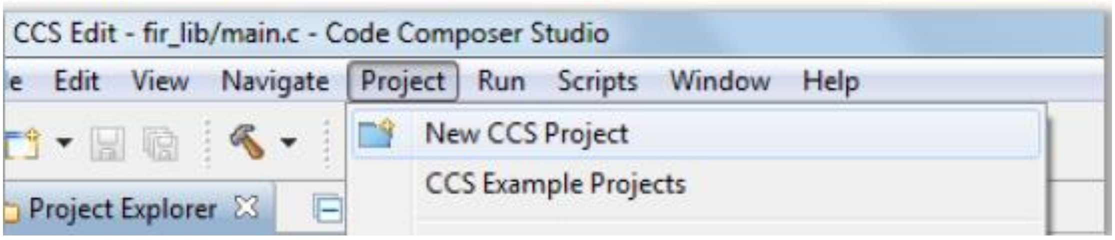
Fig: Debug Session

CREATE A NEW CCS PROJECT

Let's see how we can link these libraries to be used for our examples.

We are going to link the dsplib and implement an FIR filter, this is a non real time example.

- **Create a new CCS project**. Name it as **fir_lib**



Make the changes as follows. Please note that the connection field can be "Texas Instruments XDS100v3 Debug Probe" if you are using one.

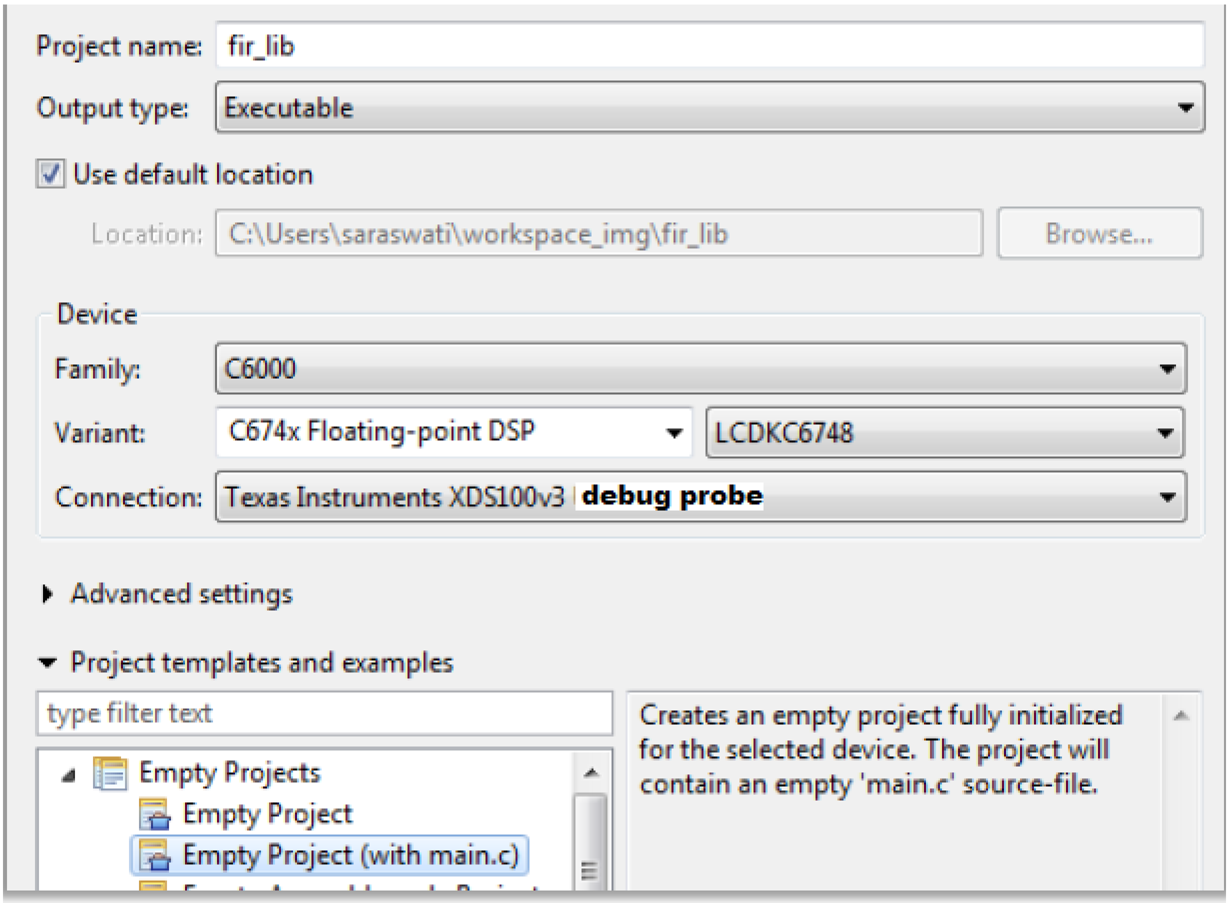


Fig: Project Creation

- Click on FINISH.

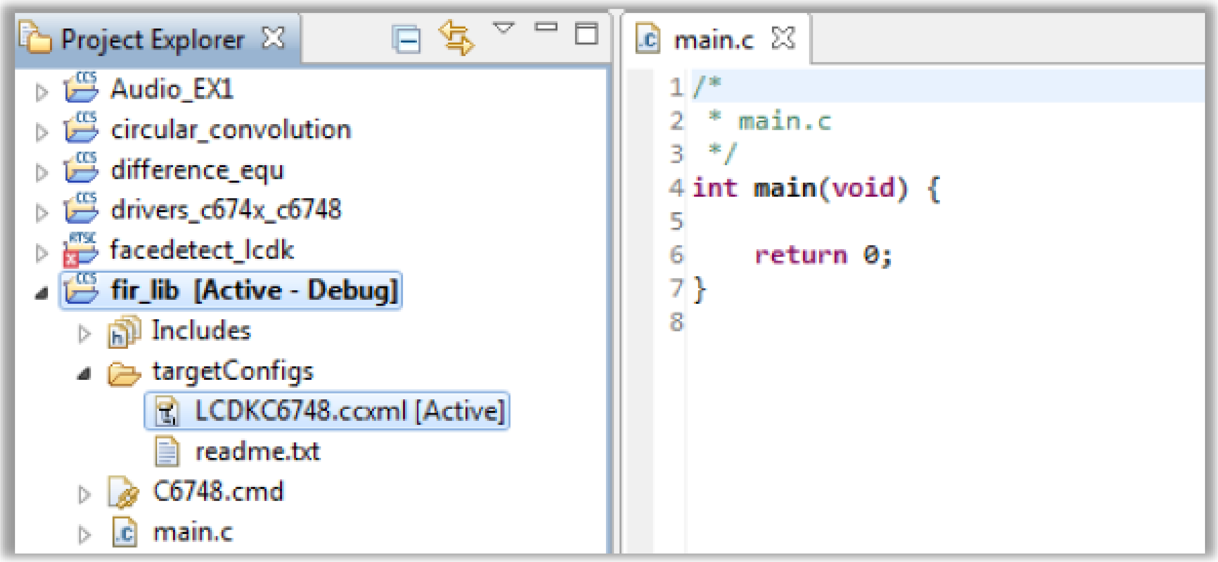


Fig: Project Explorer

1. Go to view-> target configurations
2. New target configuration: File name- test1.ccxml -> finish

3. Connection type- TIXDS100V3 Debug Probe
4. Board/Device – LCDK 6748
5. Save
6. Test connection-> wait for message "JTAG INTEGRITY SCAN TEST SUCCEEDED"
7. Right click on configuration file and set the configuration file as "default" and "launch selected configuration".
8. To open new project: (In Edit mode) Go to Project-> New CCS project
    a. Target-LCDK 6748
    b. Connection type- TIXDS100V3
    c. Name the project- eg: test123
    d. Output type: EXCE
    e. Output format- Legacycoff
    f. Linker command- c6748.cmd
    g. Runtime- rts6740.lib
    h. Finish
9. Type code in main.c
10. Build, Debug (Debug mode) and Run the code

Experiment No. 1:-Impulse response

```
#include<stdio.h>
float x[5]={1,0,0,0,0};
float y[5]={0,0,0,0,0};
void main()
{
int i,j;
printf("\nTo calculate impulse response\n");
for(i=0;i<5;i++)
{
if(i==0)
y[i]=x[i]*0.1311;
if(i==1)
y[i]=x[i]*0.1311+0.2622*x[i-1]+0.7488*y[i-1];
if(i>=2)
y[i]=x[i]*0.1311+0.2622*x[i-1]+0.7488*y[i-1]-0.2722*y[i-
2];
}
for(j=0;j<5;j++)
printf("\n %f",y[j]);
}
```

Experiment No. 2:- N point DFT and IDFT

N point DFT USING DSP KIT

```
#include <stdio.h>
#include <math.h>
float x[4]={0,1,2,3}; //input only real sequence
float Yreal[4],Yimag[4];
//for 4 point DFT to store real & imaginary
float theta;
int n,k,N=4,xlen=4;
void main()
{
for(k=0;k<N;k++)
{
Yreal[k]=0;
Yimag[k]=0; //initialize real & imag parts
for(n=0;n<xlen;n++)
{
theta = -2*3.141592*k*n/N;
//careful about minus sign
Yreal[k]=Yreal[k]+x[n]*cos(theta);
Yimag[k]=Yimag[k]+x[n]*sin(theta);
}
printf("%0.2f+j%0.2f \n", Yreal[k], Yimag[k]);
}
}//end of main
```

IDFT USING DSP KIT

```
#include<stdio.h>
#include<math.h>
float x[4],XReal[4],XImag[4],theta;
int k,n,N;
void main()
{
printf("\t\tInverse Discrete Fourier Transform(IDFT)\n");
printf("\nEnter the length of the DFT N = ");
scanf("%d",&N);
// length of the DFT
printf("\nEnter the real amd imaginary parts of X(k) as
follows\n"
"X(k) = Re{X(k)} Im{X(k)}\n");
for(k = 0; k < N; k++)
{
printf("X(%1.0f) = ",k);
// enter values of DFT
```

```
scanf("%f %f",&XReal[k],&XImag[k]);
}
// next part of the program computes inverse DFT
for(n = 0; n < N; n++)
{
x[n] = 0;
for(k = 0; k < N; k++)
// this loop computes one value of x(n)
{
theta=(2*3.141592*k*n)/N;
x[n] = x[n] + XReal[k]*cos(theta)
- XImag[k]*sin(theta);
}
x[n] = x[n]/N;
}
// Next part of program displays x(n) on the screen
printf("\nThe sequence x(n) is as follows...");
for(n = 0; n < N; n++)
// displaying x(n) on the screen
printf("\nx(%1.0f) = %0.2f",n,x[n]);
}
```

Experiment No. 3:- Linear convolution and Circular convolution

Linear convolution

```
#include<stdio.h>
int x[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int h[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int y[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int l1,l2,N;
int i,n,k;
void main()
{
printf("Enter length of x: ");
scanf("%d",&l1);
printf("Enter x[20]:\n ");
for(i=0;i<l1;i++)
scanf("%d",&x[i]);
printf("Enter length of h: ");
scanf("%d",&l2);
printf("Enter h[20]:\n ");
for(i=0;i<l2;i++)
scanf("%d",&h[i]);
N=l1+l2-1;
for(n=0;n<N;n++)
{
y[n]=0;
for(k=0;k<l1;k++)
{
if(n-k>=0 && n-k<l2)
y[n]=y[n]+x[k]*h[n-k];
}
}
printf("Convolution result y[n]= \n");
for(n=0;n<N;n++)
printf("%d\n",y[n]);

}
```

Circular convolution

```
#include<stdio.h>
int x[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int h[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int y[20]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
int m,N,sum;
int i,n,k;


void main()
{
printf("Enter N: ");
scanf("%d",&N);
printf("Enter x[20]:\n ");
for(i=0;i<N;i++)
scanf("%d",&x[i]);
printf("Enter h[20]:\n ");
for(i=0;i<N;i++)
scanf("%d",&h[i]);
// Next nested for loop calculates circular convolution
printf("\nThe circular of convolution is...");
for(m = 0; m < N; m++)
{
sum = 0.0;
for(k = 0; k < N; k++) //computation of one value in circular
{//convolution
if((m-k)>= 0)
n = m-k;
//modulo index
else
n = m-k+N;
sum += x[k] * h[n];
}
y[m] = sum;
printf("\ny[%d] = %d",m,y[m]);
}
// displaying the value on screen
}
```

EXPERIMENT No. 7:

a. Design and implementation of Analog Butterworth and Chebyshev Filters to meet the given specifications.

b. Design and implementation of IIR, Butterworth and Chebyshev Filters to meet the given specifications.

---

a. Design and implementation of Analog Butterworth and Chebyshev Filters to meet the given specifications.

i. Design an analog low pass Butterworth filter for the following specifications.
   a. Pass band ripple=-3dB
   b. Stop band ripple=-40dB
   c. Pass band frequency=1000Hz
   d. Stop band frequency=2000Hz

Show all the design steps plot and verify the same using MATLAB.

```
clc;
clear all;
close all;

ap=-3;%%db
as=-40;%%db
fp=1000;%%Hz
fs=2000;%%Hz


% ap=input('Enter the pass band ripple (in dB)');
% as=input('Enter the stop band ripple (in dB)');
% fp=input('Enter the pass band frequency (in Hz)');
% fs=input('Enter the stop band frequency (in Hz)');

op=2*pi*fp; %%angular freq in rad/sec
os=2*pi*fs; %%angular freq in rad/sec
A=log10((10^(-ap/10)-1)/(10^(-as/10)-1 ));
B=2*log10(op/os);
N= A/B;
N=ceil(N);
disp('Orderofthefilter,N=');
disp(N);

%oc=op/((10^(-ap/10)-1)^(1/(2*N))); %or
oc=os/((10^(-as/10)-1)^(1/(2*N)));

disp('Cutoff freq,oc=');
disp(oc);
% %using inbuilt function
% [N,oc]=buttord(op,os,-ap,-as,'s');
```

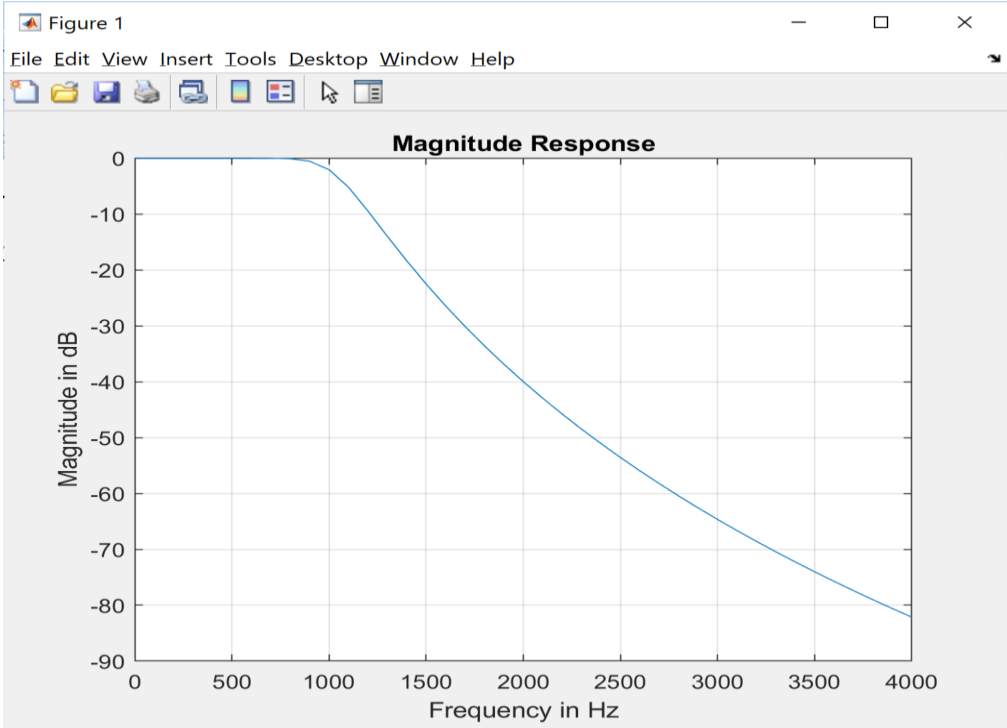
```
[b,a]=butter(N,oc,'s');

fr=0:100:4000;
wr=2*pi*fr;

[H,w]=freqs(b,a,wr);
f=w/(2*pi);
H_mag=20*log10(abs(H));
figure;
plot(f,H_mag);
grid on;
xlabel('Frequency in Hz');
ylabel('Magnitude in dB');
title('Magnitude Response');

Transferfunc=tf(b,a)
```

OUTPUT:

Orderofthefilter,N= 7
Cutoff freq,oc= 6.5088e+03

Transferfunc =

```
                            4.949e26
  -----------------------------------------------------------------------------
  s^7 + 2.925e04 s^6 + 4.278e08 s^5 + 4.024e12 s^4 + 2.619e16 s^3 + 1.18e20 s^2
                                                 + 3.417e23 s + 4.949e26
```



ii. Design an analog low pass Chebyshev type 1 filter for the following specifications.

a. Pass band ripple=-3dB
b. Stop band ripple=-40dB
c. Pass band frequency=1000Hz
d. Stop band frequency=2000Hz

Show all the design steps plot and verify the same using MATLAB.

```
clc;
clear all;
close all;

ap=-3;%%db
as=-40;%%db
fp=1000;%%Hz
fs=2000;%%Hz


% ap=input('Enter the pass band ripple (in dB)');
% as=input('Enter the stop band ripple (in dB)');
% fp=input('Enter the pass band frequency (in Hz)');
% fs=input('Enter the stop band frequency (in Hz)');

op=2*pi*fp; %%angular freq in rad/sec
os=2*pi*fs; %%angular freq in rad/sec

omegar=os/op; %%omegar in rad/sec
```

```
epsilon=sqrt(10^(-ap/10)-1);

A=sqrt(10^(-as/10));

g=sqrt((A^2-1)/(epsilon^2));

N=log10(g+sqrt(g^2-1))/log10(omegar+sqrt(omegar^2-1))%%or
%%N=acosh(g)/acosh(omegar);

N=ceil(N);
disp('Orderofthefilter,N=');
disp(N);


%using inbuilt function
% [N,oc]=cheb1ord(op,os,-ap,-as,'s');
[b,a]=cheby1(N,-ap,op,'s');

fr=0:100:4000;
wr=2*pi*fr;

[H,w]=freqs(b,a,wr);
f=w/(2*pi);
H_mag=20*log10(abs(H));
figure;
plot(f,H_mag);
grid on;
xlabel('Frequency in Hz');
ylabel('Magnitude in dB');
title('Magnitude Response');

Transferfunc=tf(b,a)
```

OUTPUT:

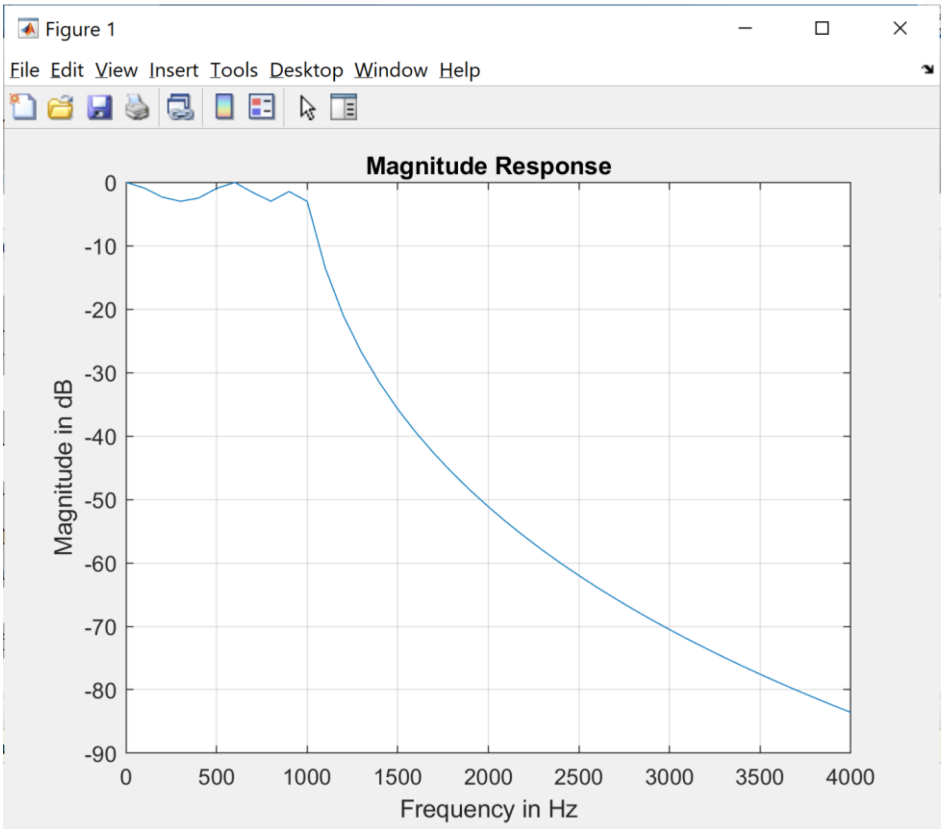
N =

4.0249

Orderofthefilter,N=
5

Transferfunc =

```
                 6.135e17
  ---------------------------------------------------------------------
  s^5 + 3610 s^4 + 5.586e07 s^3 + 1.362e11 s^2 + 6.358e14 s + 6.135e17
```



b. Design and implementation of IIR, Butterworth and Chebyshev Filters to meet the given specifications.

iii. Design a low pass Butterworth filter (using bilinear transformation) for the following specifications.

a. Pass band ripple=3.01dB
b. Stop band ripple=15dB
c. Pass band frequency=500Hz
d. Stop band frequency=750Hz
e. Sampling frequency=2kHz

Show all the design steps plot and verify the same using MATLAB.

```
clc;
clear all;
close all;
rp=input('Enter the pass band ripple (in dB)');
rs=input('Enter the stop band ripple (in dB)');
fs=input('Enter the sampling frequency (in Hz)');
fp=input('Enter the pass band frequency (in Hz)');
fst=input('Enter the stop band frequency (in Hz)');
fn=fs/2;%normalized sampling frequency
```

```
fpn=fp/fn;%normalized pass band frequency
fstn=fst/fn;%normalized stop band frequency

[N,wc]=buttord(fpn,fstn,rp,rs);
[b,a]=butter(N,wc);
[H,f]=freqz(b,a,256,fs);

Transferfunc=tf(b,a,(1/fs))
H_mag=20*log10(abs(H));
figure;
plot(f,H_mag);
grid on;
xlabel('Frequency in Hz');
ylabel('Magnitude in dB');
title('Magnitude Response');
```

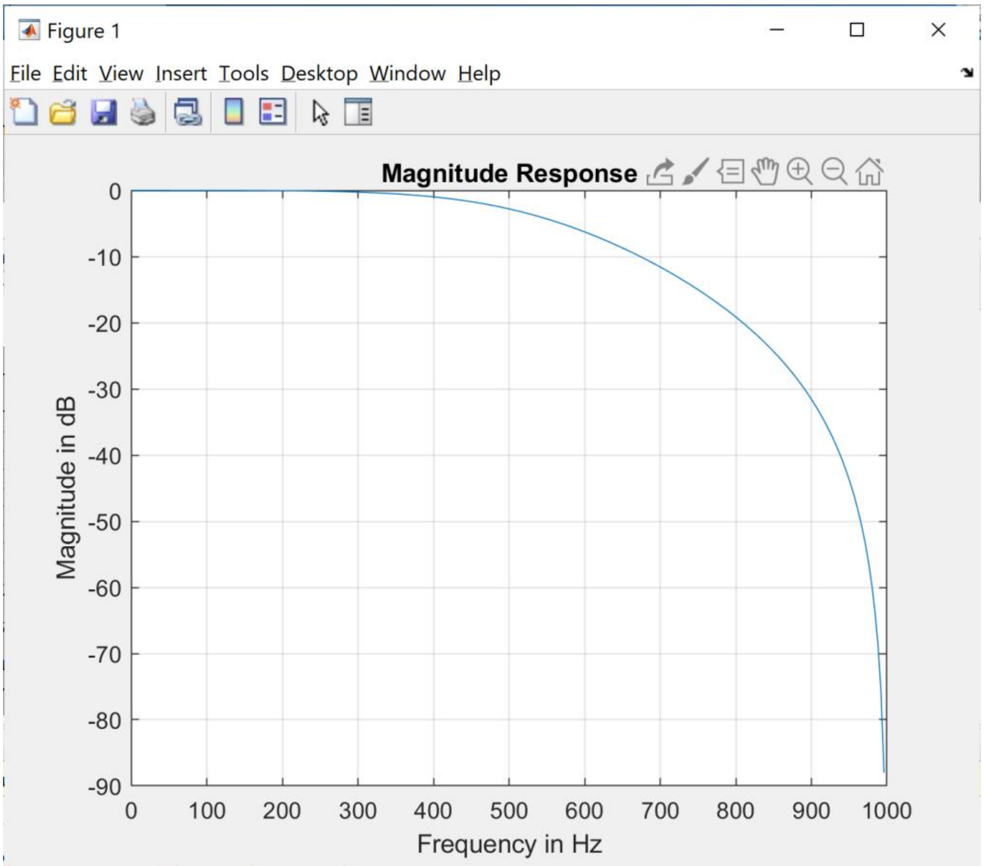
OUTPUT:

Enter the pass band ripple (in dB)3
Enter the stop band ripple (in dB)15
Enter the sampling frequency (in Hz)2000
Enter the pass band frequency (in Hz)500
Enter the stop band frequency (in Hz)750

Transferfunc =

```
  0.3005 z^2 + 0.6011 z + 0.3005
  ------------------------------
    z^2 + 0.03039 z + 0.1717
```

Sample time: 0.0005 seconds
Discrete-time transfer function.



iv. Design a low pass Chebyshev filter type-1 (using bilinear transformation) for the following specifications.

a. Pass band ripple=3dB
b. Stop band ripple=20dB
c. Pass band frequency=1200Hz
d. Stop band frequency=1800Hz
e. Sampling frequency=8kHz

Show all the design steps. Plot and verify the same using MATLAB.

```
clc;
clear all;
close all;
rp=input('Enter the pass band ripple (in dB)');
rs=input('Enter the stop band ripple (in dB)');
fs=input('Enter the sampling frequency (in Hz)');
fp=input('Enter the pass band frequency (in Hz)');
fst=input('Enter the stop band frequency (in Hz)');
fn=fs/2;%normalized sampling frequency
fpn=fp/fn;%normalized pass band frequency
fstn=fst/fn;%normalized stop band frequency
```

```
[N,wc]=cheb1ord(fpn,fstn,rp,rs);
[b,a]=cheby1(N,rp,wc);

Transferfunc=tf(b,a,(1/fs))

[H,f]=freqz(b,a,256,fs);
hn=ifft(H,256);
disp(hn);
H_mag=20*log10(abs(H));
figure;
plot(f,H_mag);
grid on;
xlabel('Frequency in Hz');
ylabel('Magnitude in dB');
title('Magnitude Response');
```

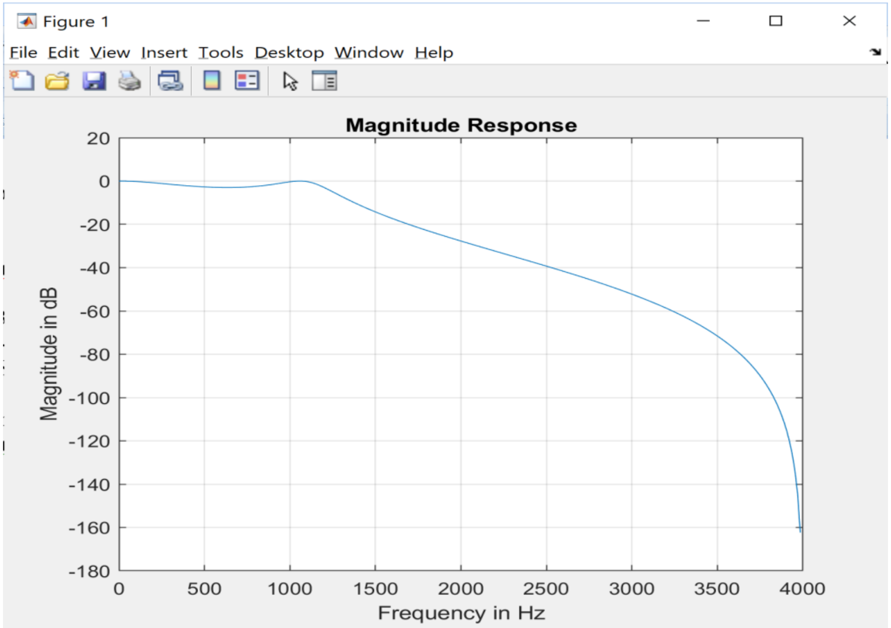
OUTPUT:

Enter the pass band ripple (in dB)3
Enter the stop band ripple (in dB)20
Enter the sampling frequency (in Hz)8000
Enter the pass band frequency (in Hz)1200
Enter the stop band frequency (in Hz)1800

Transferfunc =

```
 0.021 z^3 + 0.063 z^2 + 0.063 z + 0.021
 ---------------------------------------
   z^3 - 1.878 z^2 + 1.618 z - 0.5724
```

Sample time: 0.000125 seconds, Discrete-time transfer function.



P.E.S. UNIVERSITY
Department of Electronics and Communication Engineering
Session: Jan - May 2025

EXPERIMENT No. 8 **.Design and implementation of FIR filter to meet given specifications.**

A: FIR Filter design using windows

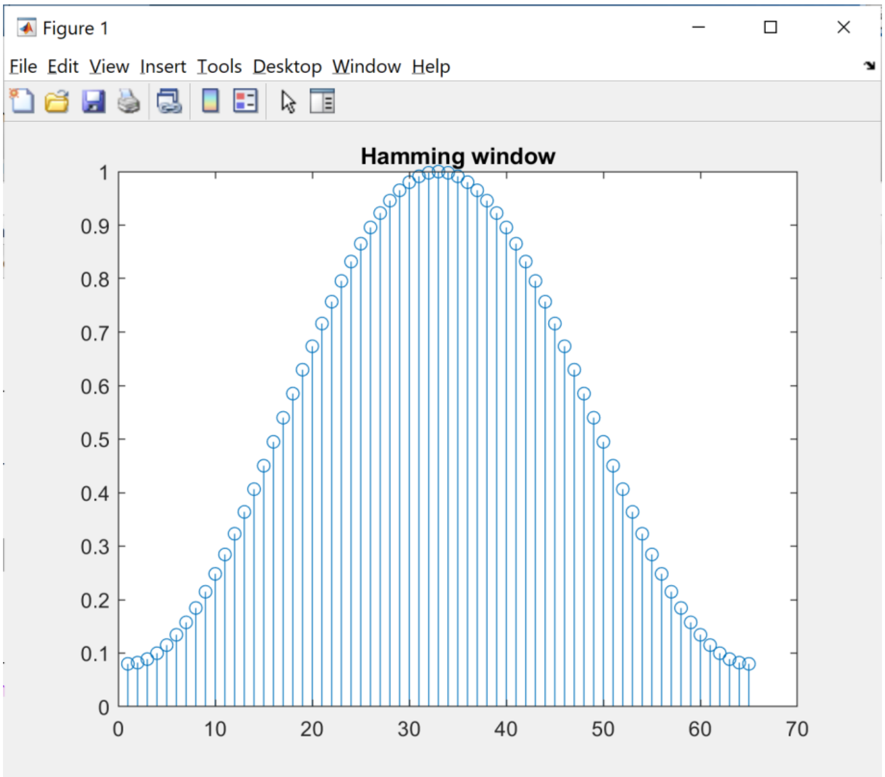
```
%%Designing Low Pass filter with pass band edge frequency=1500hz
minimum stop band attenuation is 50dB Transition bandwidth is
0.5khz sampling frequency is 8khz
clear all;
close all;

clc;
fs=8000; %%sampling frequency
fp=1500; %%pass band edge frequency
fst=2000;%%stop band edge frequency
wp=(2*pi*fp)/fs;
ws=(2*pi*fst)/fs;
twn=(ws-wp)/pi;%%normalize transition width choose hamming window
To find
wc=wp+twn*pi/2;
%%LENGTH(TAPS) of filter (i.e M).The filter order is always equal
to the number of taps minus 1
N=ceil(8/twn);
%%To design type I FIR filter , taps=odd or order=even
if(mod(N,2)==0)
N=N+1;
end
alpha=(N-1)/2;
er=0.001;
for n=0:1:N-1
hd(:,n+1)=(sin(wc*(n-alpha+er)))./(pi*(n-alpha+er));
end
%disp(hd)

figure;
whm=hamming(N);%%hamming window
disp('Hammind window coeff');
disp(whm);
stem(whm);
title('Hamming window');

hn=hd.*whm';

[H,f]=freqz(hn,1,1000,fs);
H_mag=20*log10(abs(H));

figure;
plot(f,H_mag);
xlabel('Normalized frequency');
```

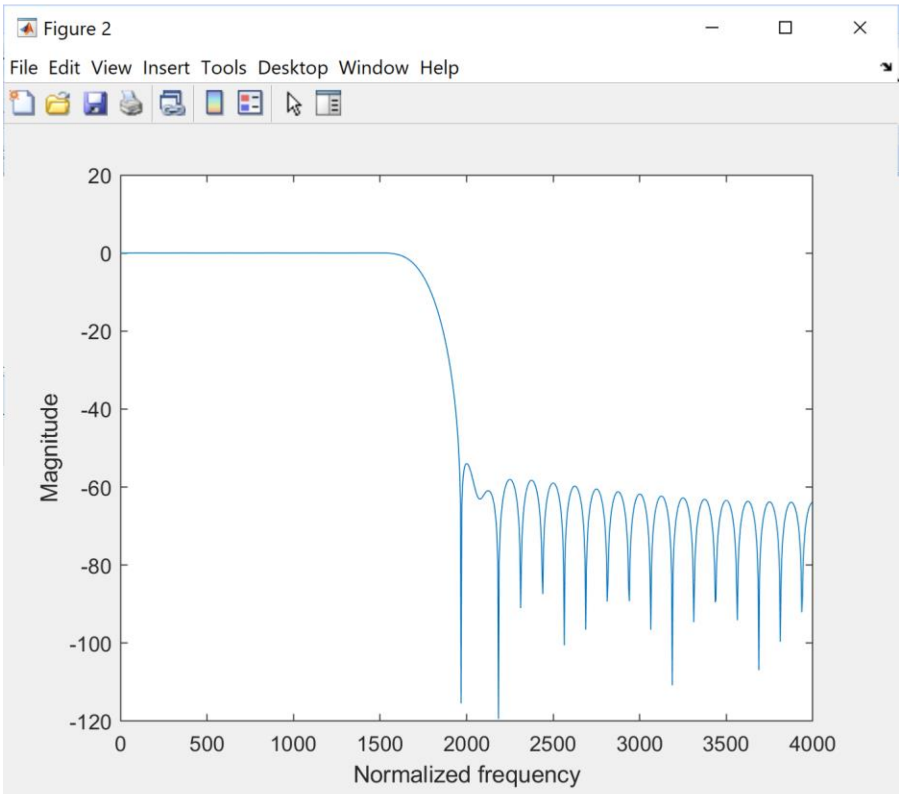
```
ylabel('Magnitude');

disp('FIR Filter coeff hn');
disp(hn);
```

OUTPUT:

```
FIR Filter coeff hn
  Columns 1 through 8

   -0.0000   -0.0008   -0.0004    0.0009    0.0009   -0.0009   -
0.0018    0.0005

  Columns 9 through 16

    0.0028    0.0007   -0.0038   -0.0027    0.0041    0.0057   -
0.0031   -0.0091

  Columns 17 through 24

    0.0000    0.0122    0.0055   -0.0137   -0.0134    0.0122
0.0234   -0.0058

  Columns 25 through 32

   -0.0344   -0.0079    0.0452    0.0334   -0.0544   -0.0864
0.0606    0.3117

  Columns 33 through 40

    0.4375    0.3113    0.0601   -0.0865   -0.0542    0.0335
0.0452   -0.0080

  Columns 41 through 48

   -0.0344   -0.0057    0.0234    0.0121   -0.0134   -0.0137
0.0055    0.0122

  Columns 49 through 56

   -0.0000   -0.0091   -0.0030    0.0057    0.0041   -0.0027   -
0.0038    0.0007

  Columns 57 through 64

    0.0028    0.0005   -0.0018   -0.0009    0.0009    0.0009   -
0.0004   -0.0008
```

```
Column 65

  0.0000
```





B. Frequency sampling technique.

```
%%Design a linear phase FIR low pass filter using
%%frequency sampling method. Let M=17(order)
%%and cutoff frequency is wc=pi/2 rad/sample
%%Desired frequency response
%%Hd(w)= exp(-%j*w(M-1)/2) for 0<=w<=pi/2
%%     = 0                 for pi/2<=w<=pi

%%from Hd(w) we get H(k) by sampling w=(2*pi*k)/M
%% H(k)= Hd(w)| at w=(2*pi*k)/M
%% h(n)=1/M <summ k=0 to M-1>[H(k)e^j2pikn/M]  0<=n<=M-1
%%h(n) must be real
%% h(n)=1/M(H(0)+ 2 <summ n=1 to p> Re[H(k)e^j2pikn/M])
%%p=(M-1)/2 if N is odd
%% =M/2 -1 if N is even

clc;
clear all;
close all;

wc=pi/2; %%cutt of freq in rad/sample

M=17; %%take odd order
```

```
%%Desired frequency response
w=0:(pi/256):pi;
Hd_w =([w<=pi/2]).* exp(-i*w*(M-1)/2);
figure;
subplot(3,1,1);
plot(w,abs(Hd_w));
title('Magnitude response of desired filter');
xlabel('w');
ylabel('|Hd(w)|');


%%(2*pi*kc)/M <= pi/2
%% kc <= (pi/2)*(M/(2*pi))
%% kc<= M/4 = 17/4, but k can take only integer=4

kc = floor(wc*(M/(2*pi)));

k=0:(M-1)/2;

%%frequency sampling Hd_w to get Hk
Hk = ([k<=kc]).* exp(-i*(2*pi*k/M)*((M-1)/2))

subplot(3,1,2);
stem(k,abs(Hk));
title('Sampled Magnitude response Hd_w');
xlabel('k');
ylabel('|H(k)|');

%% Taking idft of H(k) to find h(n)

for n=0:(M-1)
       sumHk=0;
    for k=1:(M-1)/2
        sumHk = sumHk+real(Hk(k+1)*exp(i*2*pi*k*n/M));
    end

    hn(n+1)=(1/M)*(Hk(1)+2*sumHk);
end



n=0:(M-1);
subplot(3,1,3);
stem(n,hn);
title('Impulse response h(n) of filter')
xlabel('n');
ylabel('hn');
```

```
[H,f]=freqz(hn,1,1000);
H_mag=20*log10(abs(H));

figure;
plot(f,H_mag);
xlabel('Normalized frequency');
ylabel('Magnitude');

disp('FIR Filter coeff hn');
disp(hn);
```

OUTPUT:

```
Hk =

  Columns 1 through 4

   1.0000 + 0.0000i  -0.9830 - 0.1837i   0.9325 + 0.3612i  -0.8502
- 0.5264i

  Columns 5 through 8

   0.7390 + 0.6737i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000
+ 0.0000i

  Column 9

   0.0000 + 0.0000i

FIR Filter coeff hn
  Columns 1 through 8

    0.0398   -0.0488   -0.0346    0.0660    0.0315   -0.1075   -
0.0299    0.3188

  Columns 9 through 16

    0.5294    0.3188   -0.0299   -0.1075    0.0315    0.0660   -
0.0346   -0.0488

  Column 17

    0.0398
```

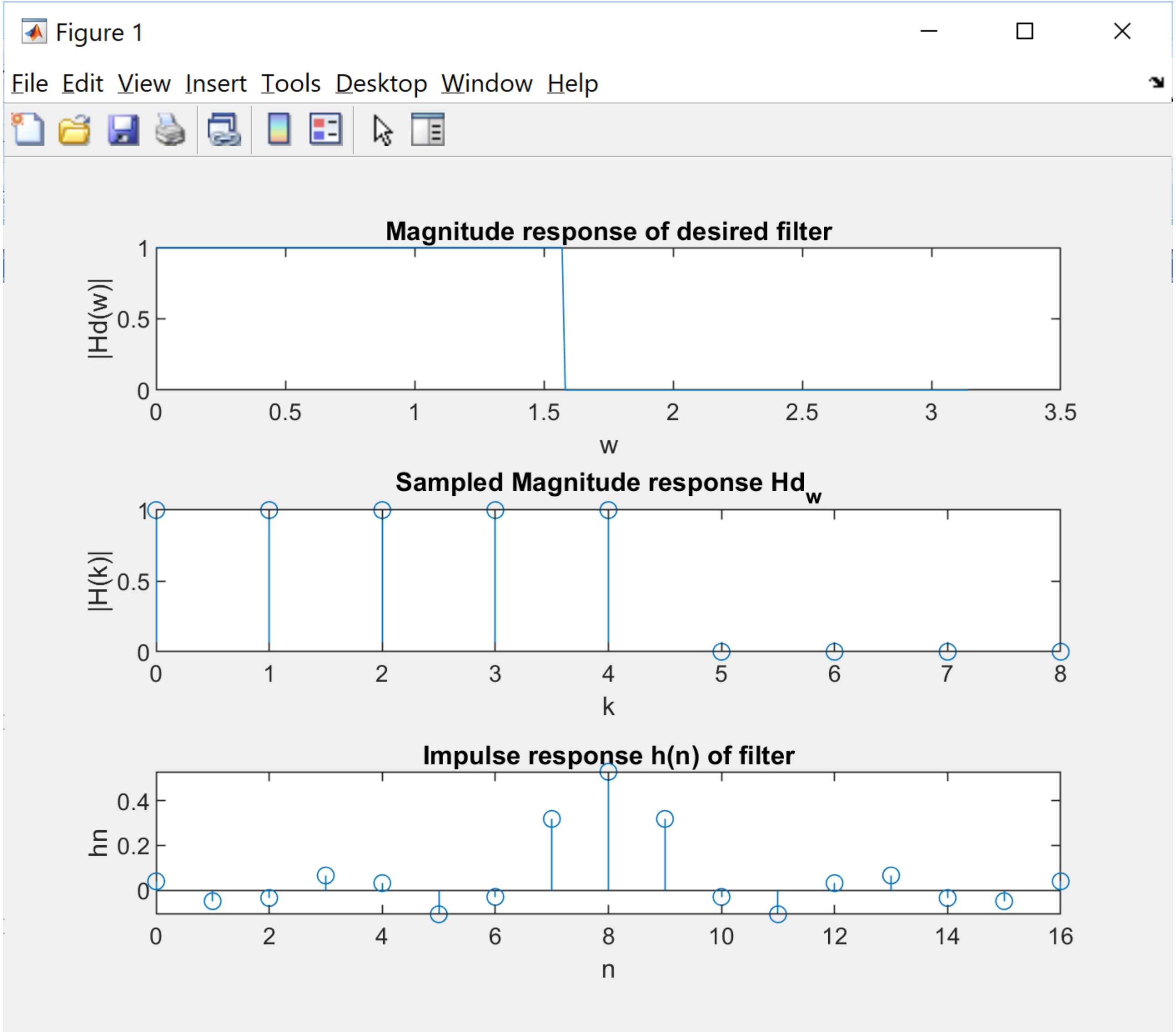


Figure 1
File Edit View Insert Tools Desktop Window Help
Magnitude response of desired filter
|Hd(w)|
w
Sampled Magnitude response Hd
|H(k)|
k
Impulse response h(n) of filter
hn
n

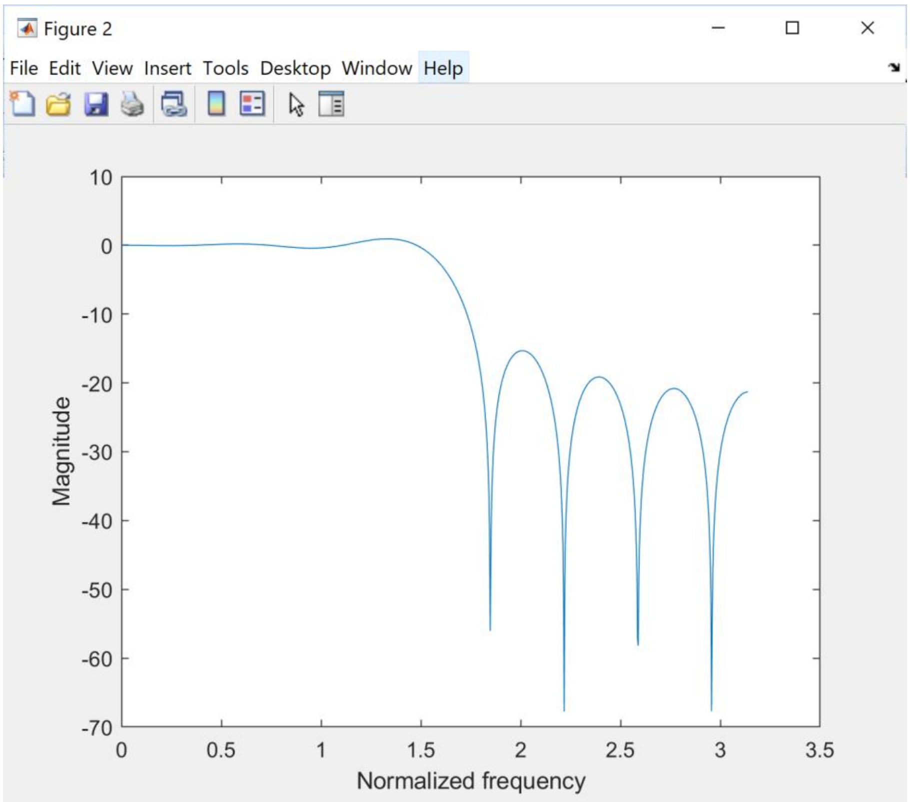


Figure 2
File Edit View Insert Tools Desktop Window Help
10
0
-10
-20
-30
-40
-50
-60
-70
Magnitude
0
0.5
1
1.5
2
2.5
3
3.5
Normalized frequency