CONTENTS

FAMILIARISATION OF MATLAB

AIM

To familiarize MATLAB and its functions.

THEORY

MATLAB is an ideal software tool for studying digital signal processing. The plotting capacity of MATLAB makes it possible to view the result of processing and gain understanding into complicated operations. The tool box support a wide range of signal processing from waveform generation to filter design and implementation, parametric modelling and spectral analysis. There are two categories of tools Signal processing functions, Graphical interaction tools. The first category of tools is made of functions that we can call from the common line or from your own application. Many of the functions are MATLAB statements that implement specialized signal processing algorithm.

(a) DEFINTION OF VARIABLES

Variables are assigned numerical values by typing the expression directly

Eg: a=1+2 yields a=3

The answer will not be displayed when semicolon is put at the end of an expression

Eg: a=1+2

MATLAB utilizes the following arithmetic operations

+ addition

- subtraction

* multiplication

/ division

ˆpower operation ,

‘ transpose

A variable can be assigned using a formula that utilizes these operations and either numbers or previously defined variables.

Eg: since ‘a’ was defined previously the following expression is valid, B=z*a. To determine the value of a previously defined quantity type the quantity by b=6. If your expression does not fill on the line, use an ellipse and continue on the next line.

C=1+2+3+. . . . .+5+6+7.

These are certain predefined variables which can be used at any time, in the same manner as user defined variables.

i=sqrt(-1); j=sqrt(-1)

pi = 3.1416

Eg: y = 2*(1+4*j) yields y = 2.000+8.000y

These are also a number of predefined functions that can be used when defining a variable. Some common functions that are used in this text are Abs- magnitude of a number Angle- angle of a complex number Cos- cosine function, assume arguments in radian.

Exp-exponential functions

For example with y defined as above E= abs(y), yields e=8.2462, c=angle(y) yields c=1.3258 with a=3 as defined previously c=cos(a)yields, c= -0.9900c = exp(a), yields c=20.0855. Note that exp can be used on complex numbers example with y = 2 + 8i as defined above

c = -1.0751 + 7.3104i which can be verified by using Euler’s formula

c = c2cos(8) + jc2sin(8).

(b) DEFINTION OF MATRICES

MATLAB is based on matrix and vector algebra. Even scalars are treated 1x1 matrix. Therefore, vector and matrix operation are simple as common calculator operations. Vectors can be defined in two ways. The first method is used for arbitrary elements, v=[1 3 5 7] creates 1x4 vector elements with elements 1 3 5 &7. Note that commas would have been used in the place of spaces to separate the elements. Additional elements can be added to the vector v(5)=8 yields the vector v = [1357]. Previously defined vectors can be sued to define a new vectors. For example with we defined above a= [910]; b = [v a]; creates the vector b = [1357910]. The second method is used for creating vector with equally spaced elements t=0:0.1:10; creates 1x101 vector with elements

0,0.1,0.2. . . . . 10. Note that the middle number defines the increments is set to a default of 1 k=0,10 creates 1x11 vector with the elements 0,1,2. . . .10. Matrices are defined by entering the element row by row. M = [124; 368] creates the matrix M= 1 2 4 3 6 8 There are number of special matrices that can be defined Null matrix: [ ];

Nxm matrixes of zero: M=zeros (m,m);

Nxm matrix of ones: M= ones (n,m);

Nxn matrix of identity: M=eye (n)

A particular elements of matrix can be assigned M(1,2)=5 place the number 5 in the first row, 2nd column. Operations and functions that were defined for scalars in the previous section can be used on vectors and matrices. For example a=[1 2 3]; b=[4 5 6]; c=a+b yield c=579. Functions are applied element by element. For example t=0:'0; x=cos(2*t) creates a vector 'x' with elements equal to cos(2t) for t=0,1,2. 10

(c) GENERAL INFORMATION

MATLAB is case sensitive. So 'a' and 'A' are two different names. Comment statements are preceded by 'a'.

1) M-files M-files are macros of MATLAB commands that are stored as ordinary text file with the extension 'in' that is 'filename.m'. An m-file can be either a function with input and output variables or a set of commands. MATLAB requires that the m-file must be stored either in the working directory or in a directory that is specified in the MATLAB path list. The following commands typed from within MATLAB demonstrate how this m-file is used. X=2,y=3,z=y plus x(y,x) MATLAB m-files are most efficient when written in a way that utilizes matrix or vector operations, loops and if statements are available, but should be used sparingly since they are computationally inefficient. An examples is For k=1:10 x(k)=cos(k) end; This creates a 1x10 vector 'x' containing the cosine of the positive integers from 1 to 10. This operation is performed more efficiently with the commands k=1:10 x=cos(k)

which utilizes a function of a vector instead of a for loop. An if statement can be used to define combinational statement.

(d) REPRESENTATION OF SIGNALS

A signal can be defined as function that conveys information, generally about or behaviour of physical system; signals are represented mathematically as functions of one or more independent variables. The independent variables is the mathematical expression of a signal may be either continuous is discrete. Continuous time signals are defined along a number of times and thus they are represented by a continuous independent variable. The MATLAB is case sensitive. Discrete time signals are represented mathematically. Continuous time signals are often referred to as analog signals. Discrete time signals are defined at discrete times and thus the independent variable has discrete values. In a sequence of numbers x, the nth number in the sequence is denoted as x[n]. The basic signals used in digital signal processing are the unit impulse signal [U+0260](n), exponential of the form a u[n], sine waves and their generalization to complex exponentials. Since the only numerical data type in MATLAB is the m x n matrix, signals must be represented as vectors either mx1 matrix if column vector 1xn matrices if row vectors. A constant will be treated as 1x1 matrix. The signals are sampling frequency Fs, which is greater than twice the maximum frequency of the signal. In time domain, the signals are represented as time versus amplitude. In MATLAB you can generate time base for given signal as T=t start: 1/Fs:t-stop, tstart is the starting time of the signal t-stop is the stop time of the signal and Fs sampling frequency. 1/Fs is the time period between the two samples. In signals like speech if the data secured length is too long, plotting of the whole signal will not be clean view. In that case we can use step function.

ALGORITHM

1. Start
2. Generate the time axis
3. Define the unit impulse signal
4. Plot the signal
5. Define unit step signal
6. Plot the unit step signal
7. Define sine function
8. Plot sine function
9. Define ramp function
10. Plot ramp function
11. Define exponential function
12. Plot exponential function
13. Stop

EXPERIMENT NO: 1

GENERATION OF WAVEFORMS

AIM

To Generate Continious And Discrete Waveforms

MATLAB FUNCTIONS USED

• CLC: clc clears all input and output from the Command Window display, giving you a “cleanscreen.”After using clc, you cannot use the scroll bar to see the history of functions, but youstill can use the up arrow to recall statements from the command history.

• CLEAR ALL: clear removes all variables from the workspace. This frees up system memory.

• CLOSE ALL: Closes all the open figure windows

• SUBPLOT: subplot(m,n,P). Subplot divides the current figure into rectangular panes that are numbered row wise. Each pane contains an axes object. Subsequent plots are output to the current plane.

• STEM: stem(X,Y). A two-dimensional stem plot displays data as lines extending from a baseline along the x-axis. A circle (the default) or other marker whose y-position represents the data value terminates each stem.

• TITLE: title('string'). Each axes graphics object can have one title. The title is located at the top and in the centre of the axes.

• XLABEL: xlabel('string'). Each axes graphics object can have one label for the x-, y-, and z-axis. The label appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot.

• YLABEL: ylabel('string'). Each axes graphics object can have one label for the x-, y-, and z-axis. The label appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot.

• ZEROS: Create array of all zeros

• ONES: Create array of all ones

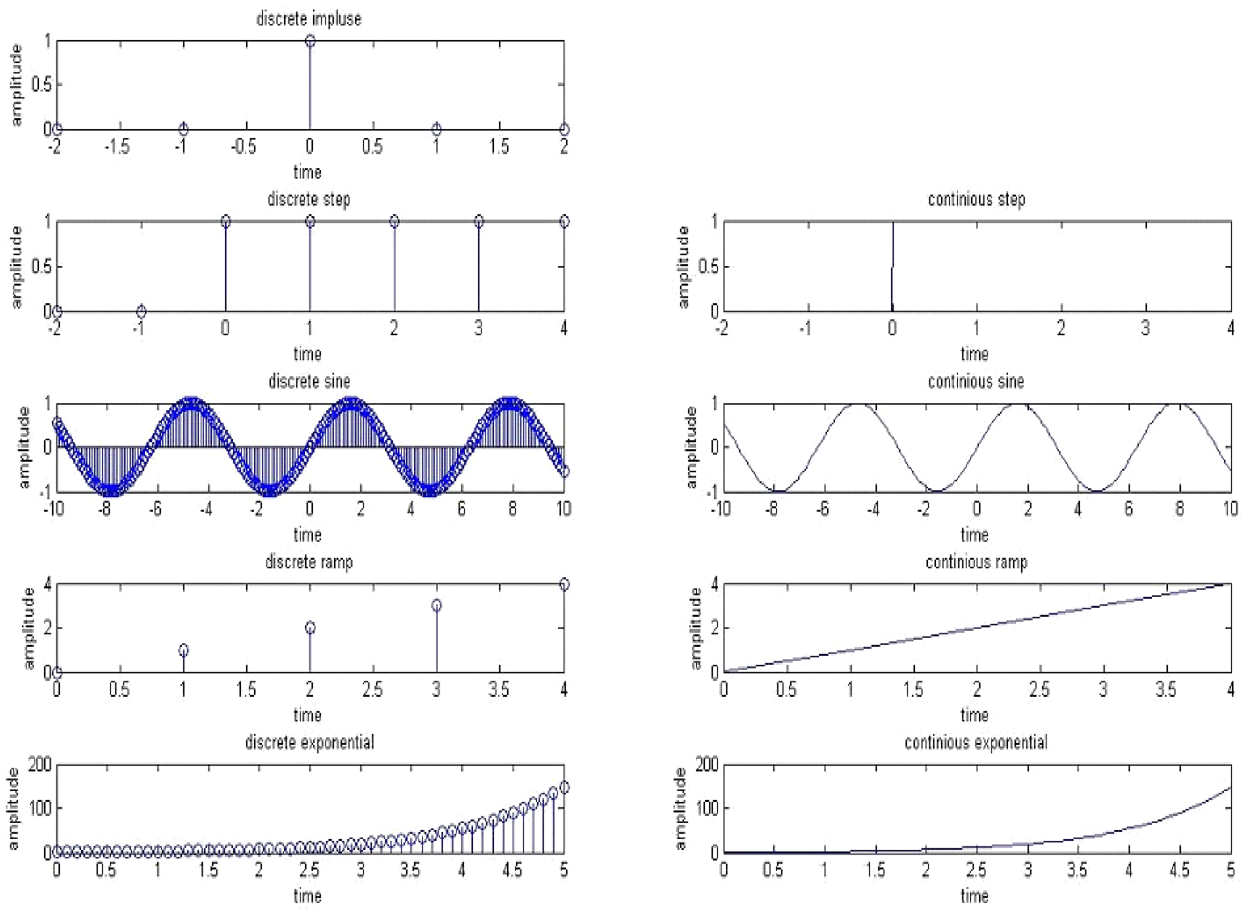
• SIN: Y = sin(X). The sin function operates element-wise on arrays. The function's domains and ranges include complex values. All angles are in radians.

• EXP: Y = exp(X). The exp function is an elementary function that operates element-wise on arrays. Its domain includes complex numbers.

PROGRAM

```
clc;%clears the command window and homes the cursor.
clear all;%removes all variables, globals, functions and MEX links,also removes the Java packages import list.
close all;%closes all the open figure windows.
%impulse function
n=-2:1:2;%TIME INTERVAL(-2,2) WITH DIVITION=1
y=[zeros(1,2),ones(1,1),zeros(1,2)];%zeros(1,2)OR ones(1,2) means matrix of 1 row and 2 coloumns containing zeros OR ones as elements
subplot(5,2,1);%position of next plot in a plot partition : 5 rows 2 coloumns is 1
stem(n,y);%discrete plot
title('discrete impluse');%title of plot
xlabel('time');%variable along x axis
ylabel('amplitude');%variable along y axis
%step function
n=-2:1:4;
y=[zeros(1,2),ones(1,5)];
subplot(5,2,3);
stem(n,y);
```

OUTPUT


discrete impluse
discrete step
continious step
discrete sine
continious sine
discrete ramp
continious ramp
discrete exponential
continious exponential
amplitude
time

```
title('discrete step');
xlabel('time');
ylabel('amplitude');
n=-2:0.01:4;
y=[zeros(1,200),ones(1,401)];
subplot(5,2,4);
plot(n,y);
title('continious step');
xlabel('time');
ylabel('amplitude');
%sine function
n=-10:0.1:10;
y=sin(n);
subplot(5,2,5);
stem(n,y);
title('discrete sine');
xlabel('time');
ylabel('amplitude');
subplot(5,2,6);
plot(n,y);
title('continious sine');
xlabel('time');
ylabel('amplitude');
%ramp function
n=0:1:4;
y=n;
subplot(5,2,7);
stem(n,y);
title('discrete ramp');
```

```
xlabel('time');
ylabel('amplitude');
subplot(5,2,8);
plot(n,y);
title('continious ramp');
xlabel('time');ylabel('amplitude');
%exponential function
n=0:0.1:5;
y=exp(n);
subplot(5,2,9);
stem(n,y);
title('discrete exponential');xlabel('time');
ylabel('amplitude');
subplot(5,2,10);
plot(n,y);title('continious exponential');
xlabel('time');ylabel('amplitude');
```

RESULT

The MATLAB program to generate continuous and discrete waveforms were executed and obtained output waveforms.

ALGORITHM

1. start
2. Read the signal frequency, sampling frequencies
3. Generate Continuous time signal
4. Plot Continuous time signal

5. Plot the sampled signal

6. take inverse Fourier transform of the sampled signal

7. Plot the reconstructed signal

EXPERIMENT NO: 2

SAMPLING THEOREM

AIM

Write a MATLAB program to verify the sampling theorem.

MATLAB FUNCTIONS USED

• CLC: clc clears all input and output from the Command Window display, giving you a “cleanscreen.”After using clc, you cannot use the scroll bar to see the history of functions, but youstill can use the up arrow to recall statements from the command history.

• CLEAR ALL: clear removes all variables from the workspace. This frees up system memory.

• CLOSE ALL: Closes all the open figure windows

• SUBPLOT: subplot(m,n,P). Subplot divides the current figure into rectangular panes that are numbered row wise. Each pane contains an axes object. Subsequent plots are output to the current plane.

• STEM: stem(X,Y). A two-dimensional stem plot displays data as lines extending from a baseline along the x-axis. A circle (the default) or other marker whose y-position represents the data value terminates each stem.

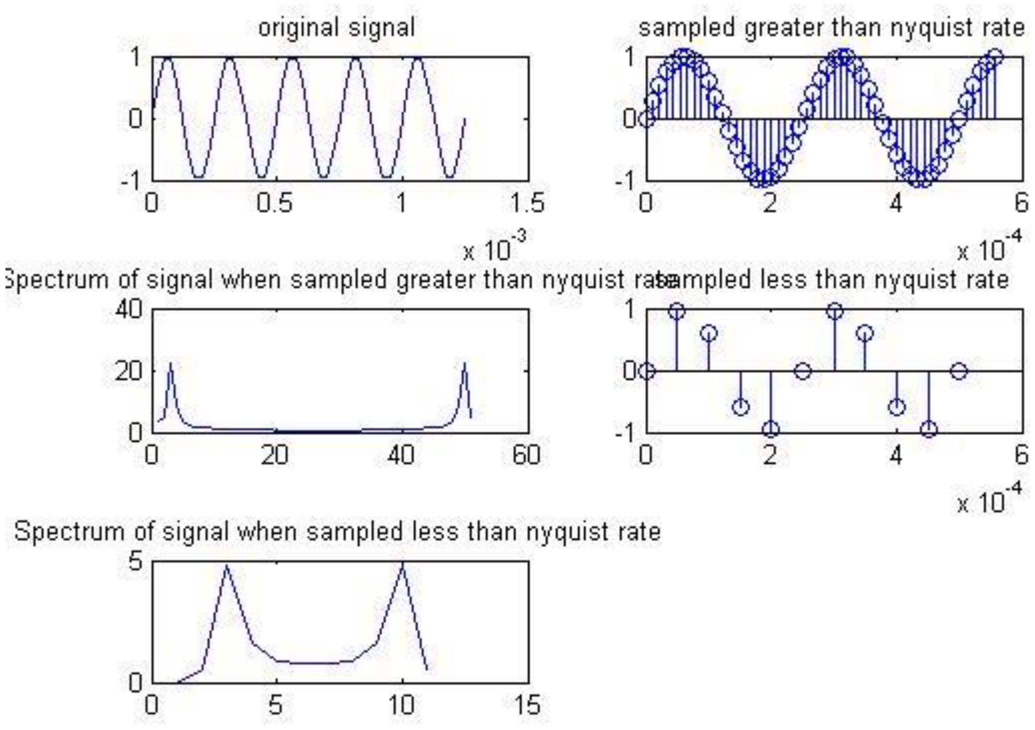
• TITLE: title('string'). Each axes graphics object can have one title. The title is located at the top and in the centre of the axes.

• XLABEL: xlabel('string'). Each axes graphics object can have one label for the x-, y-, and z-axis. The label appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot.

• YLABEL: ylabel('string'). Each axes graphics object can have one label for the x-, y-, and z-axis. The label appears beneath its respective axis in a two-dimensional plot and to the side or beneath the axis in a three-dimensional plot.

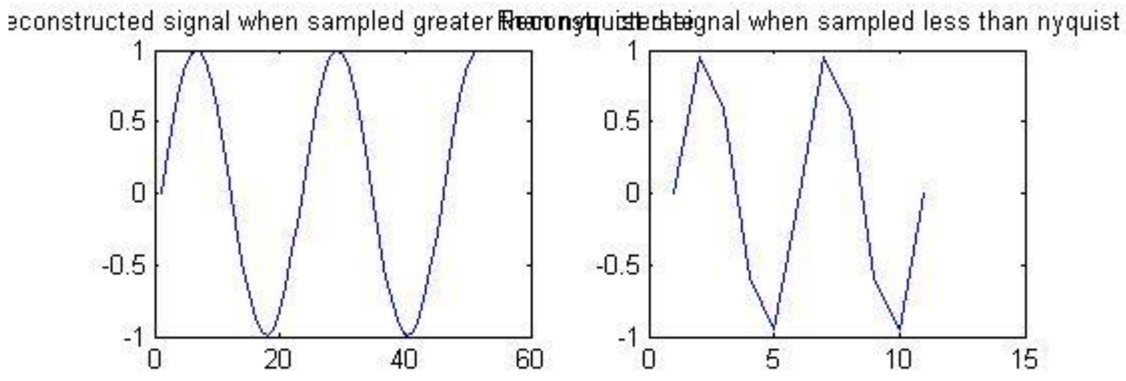
OUTPUT

```
Enter the frequency of the signal =4000
Enter the sampling frequency greater than 2*fm=9000
Enter the sampling frequency less  than fm=2000
```



PROGRAM

```
clear all;
close all;
clc;
fm=input('Enter the frequency of the signal =');
k=1/fm
fs1=input('Enter the sampling frequency greater than 2*fm=');
fs2=input('Enter the sampling frequency less  than fm=');
l=1/fs1;
m=1/fs2;
t=0:k/10:5*k;
t1=0:l/10:5*l;
t2=0:m/10:m;
a=sin(2*pi*fm*t);
b=sin(2*pi*fm*t1);
c=sin(2*pi*fm*t2);
subplot(3,2,1);plot(t,a);
title('original signal')
subplot(3,2,2);stem(t1,b);
title('sampled greater than nyquist rate')
sp1=fft(b);
```



econstructed signal when sampled greater than nyquist rate
Reconstructed signal when sampled less than nyquist
1
0.5
0
-0.5
-1
0
20
40
60
1
0.5
0
-0.5
-1
0
5
10
15

```
subplot(3,2,3);plot(abs(sp1));
title('Spectrum of signal when sampled greater than nyquist rate')
sp2=fft(c);
subplot(3,2,4);stem(t2,c);
title('sampled less than nyquist rate')
subplot(3,2,5);plot(abs(sp2));
title('Spectrum of signal when sampled less than nyquist rate')
figure;
re1=ifft(sp1);
re2=ifft(sp2);
subplot(2,2,1);plot(re1);
title('Reconstructed signal when sampled greater than nyquist rate ')
subplot(2,2,2);plot(re2);
title('Reconstructed signal when sampled less than nyquist rate ')
```

RESULT

The MATLAB program to verify the sampling theory was executed and output waveform for fs>>signal frequency and fs<signal frequency is executed. And theorem is verified.

ALGORITHM

1.Start

2.Enter values of fc,fm and m

3.Generate carrier signal

4. Generate modulating signal

5. Generate AM signal using formula

6. Plot AM signal with xlabel and ylabel

7. Stop

EXPERIMENT NO: 3

AMPLITUDE MODULATION

AIM

Write a MATLAB program to plot the amplitude modulation of the given input signal.

MATLAB FUNCTIONS USED

• SIN : Y = sin(X). The sin function operates element-wise on arrays. The function’s domains and ranges include complex values. All angles are in radians.

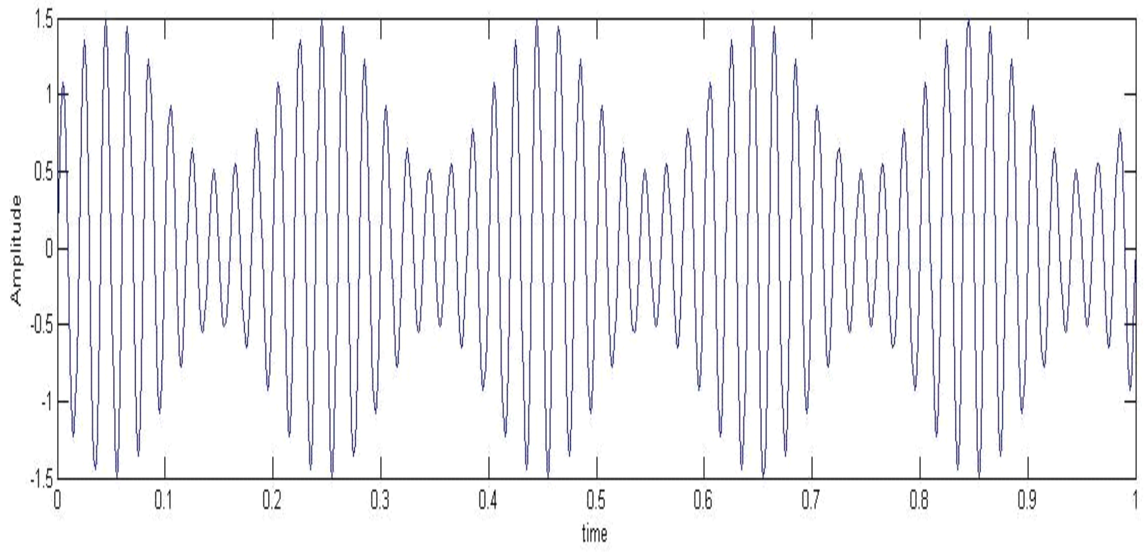
• INPUT( ): Request user input.

PROGRAM

```
% Amplitude modulation of the given input signal.ll;

clc;

clear all;

close all;

fc=input('enter the carrier signal frequency in hz,fc=');

fm=input('enter the modulating signal frequency in hz,fm=');

m=input('modulation index,m=');

n=0:.001:1;

c=sin(2*pi*fc*n); %carrier signal

M=sin(2*pi*fm*n); %modulating signal

y=(1+m*M).*c; %AM signal

subplot(211);

plot(n,y);

ylabel('Amplitude');xlabel('time');
```

OUTPUT

```
enter the carrier signal frequency in hz,fc= 50

enter the modulating signal frequency in hz,fm= 5

modulation index,m=.5
```



RESULT

The MATLAB program to plot the amplitude modulation of the given input signal is executed and output is obtained.

ALGORITHM

1. Start.

2. Input the values of a and b.

3. Obtain magnitude response on decibel scale.

4. Obtain the phase response.

5. Input two sequences x and y.

6. Find the linear convolution.

7. Obtain the time response.

8. Stop

EXPERIMENT NO: 4

TIME AND FREQUENCY RESPONSE OF AN LTI SYSTEM

AIM

Write a MATLAB program to obtain the time and frequency response of an LTI system.

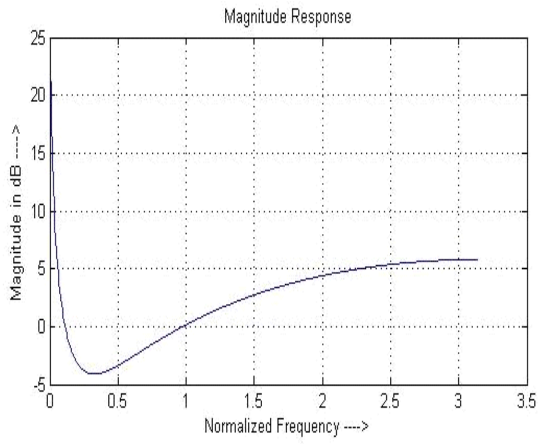
MATLAB FUNCTIONS USED

• FREQZ :Frequency Response Filter. [h,w] = freqz(ha) returns the frequency response vector h and the corresponding frequency vector w for the adaptive filter ha. When ha is a vector of adaptive filters, freqz returns the matrix h. Each column of h corresponds to one filter in the vector ha.
• ABS( ) : abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.
• GRID ON : Grid lines for 2-D and 3-D Plots.

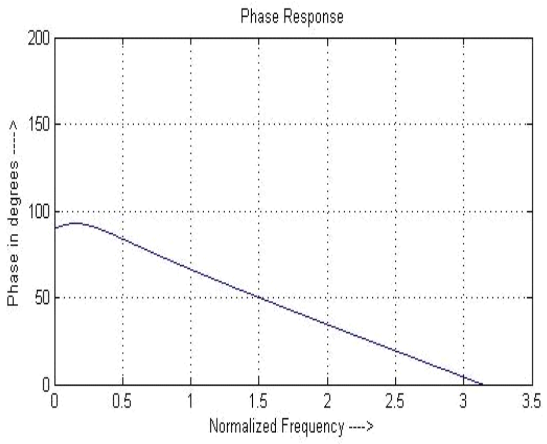
PROGRAM

```
clc;
clear all;
b = [1,-1];
a = [0.9,-2,1];
[h,w] = freqz(a,b);% freqz returns two arguments:
%the vector of frequency response values h at samples of the frequency domain
given by w
subplot(2,2,1);
mag = 20*log10((abs(h)));%magnitude response on a decibel scale
plot(w,mag);%plotting magnitude response
xlabel('Normalized Frequency ---->');
ylabel('Magnitude in dB ---->');
```

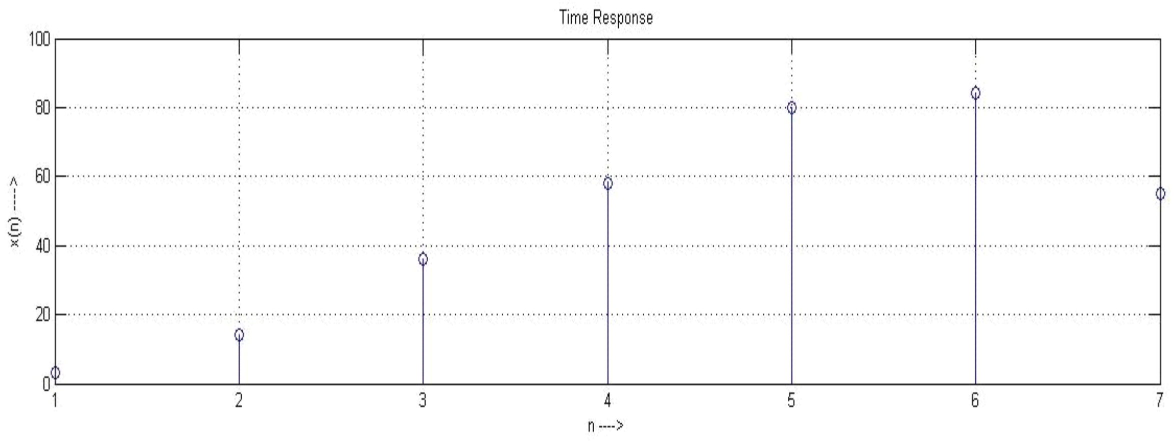
OUTPUT



Magnitude Response
Magnitude in dB ---->
Normalized Frequency ---->



Phase Response
Phase in degrees ---->
Normalized Frequency ---->



Time Response
x(n) ---->
n ---->

```
title('Magnitude Response');
grid on;
subplot(2,2,2);
h2 = (angle(h)*180/pi);%h is in radian.this function converts radian to degree

plot(w,h2);%plotting phase response
xlabel('Normalized Frequency ---->');
ylabel('Phase in degrees ---->');
title('Phase Response');
grid on;
x = [1,2,3,4,5];
y = [3,8,11];
z = conv(x,y);%linear convolution of x and y
subplot(2,1,2);
stem(z);%plotting time response
xlabel('n ---->');
ylabel('x(n) ---->');
title('Time Response');
grid on;
```

RESULT

The MATLAB program to to obtain the time and frequency response of an LTI system is executed and output is obtained.

ALGORITHM

1.Start

2.Enter two sequence

3.Find length of two sequence

4,Find length of output sequences

5.Apply zero padding

6.Calculate convolution using matrix metheod

7.plot output

8.Stop

EXPERIMENT NO: 5

LINEAR CONVOLUTION

AIM

Write a MATLAB program to obtain linear convolution of two sequences.

MATLAB FUNCTIONS USED

• LENGTH: Length of vector. LENGTH(X) returns the length of vector X.

• FLIPLR: Flip matrix left to right. fliplr(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.

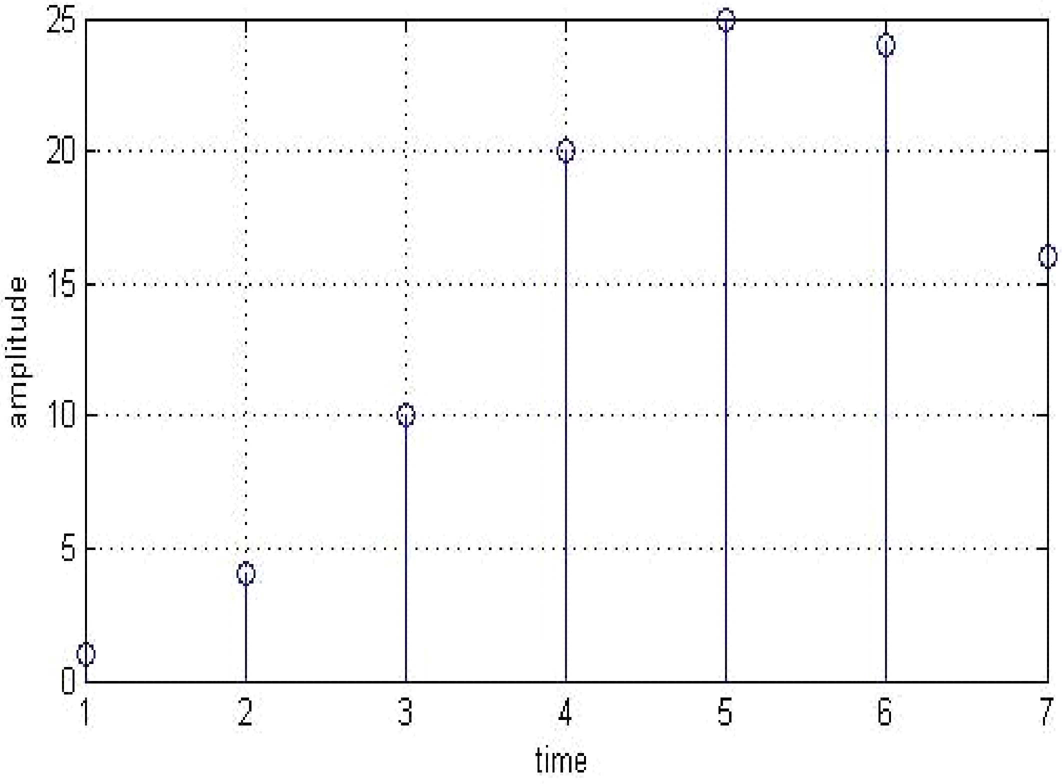
• ZEROS: Create array of all zeros

PROGRAM

```
%linear convolution using matrix metheod
clc;
clear all;
close all;
x=input('enter the first sequence');
h=input('enter the second sequence');
n1=length(x);  %find length of x
n2=length(h);  %find length of h
n=n1+n2-1;%find length of output sequence
x=[x,zeros(1,n-n1)];% zero padding
h=[h,zeros(1,n-n2)];% zero padding
h=fliplr(h) ; %returns h with row preserved and columns flipped in the left/right
direction.
a=h(n);  % convolution using matrix metheod
for j=n:-1:2
   h(j)=h(j-1);
end;
```

OUTPUT

```
enter the first sequence[1 2 3 4]
enter the second sequence[1 2 3 4]
linear convolution of x and y is
    1    4   10   20   25   24   16
```



```
h(1)=a;
for i=1:n
y(i)=0;
for k=1:n
    y(i)=y(i)+(x(k)*h(k));%matrix multiplication
end;
a=h(n);
for j=n:-1:2
    h(j)=h(j-1);
end;
h(1)=a;
end;
subplot(221);
stem(y);% To plot y
disp('linear convolution of x and h is '); disp(y);
xlabel('time'); ylabel('amplitude'); grid on;
```

RESULT

The MATLAB program to find linear convolution of given sequences is executed and output is obtained.

ALGORITHM

1.Start

2.Enter two sequence

3.Find length of two sequence

4,Find length of output sequences

5.Apply zero padding

6.Calculate convolution using matrix metheod

7.plot output

8.Stop

EXPERIMENT NO: 6

CIRCULAR CONVOLUTION

AIM

Write a MATLAB program to obtain circular convolution of two sequences.

MATLAB FUNCTIONS USED

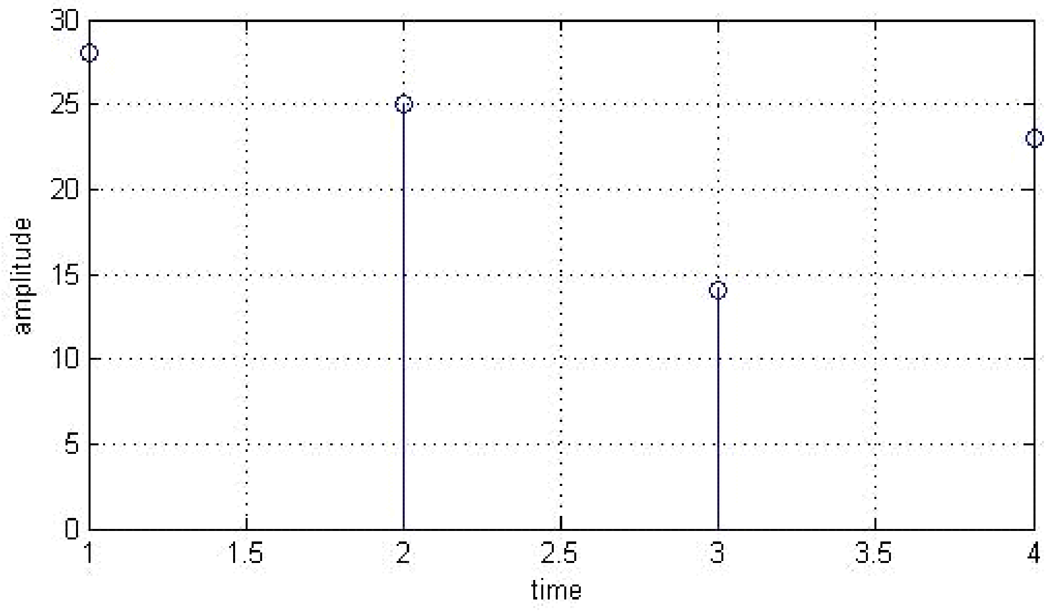
• LENGTH: Length of vector. LENGTH(X) returns the length of vector X.
• FLIPLR: Flip matrix left to right. fliplr(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.
• ZEROS: Create array of all zeros

PROGRAM

```
%circular convolution using matrix metheod

clc; clear all;

close all;

x=input('enter the first sequence');

h=input('enter the second sequence');

n1=length(x);  %find length of x

n2=length(h);  %find length of h

n=max(n1,n2);%find length of output sequence

x=[x,zeros(1,n-n1)];% zero padding

h=[h,zeros(1,n-n2)];% zero padding

h=fliplr(h);%returns h with row preserved and columns flipped in the left/right
direction.

a=h(n);% convolution using matrix metheod
```

OUTPUT

```
enter the first sequence[1 2 3 4]

enter the second sequence[1 3 5]

circular convolution of two sequences:

    28    25    14    23
```



```
for j=n:-1:2
    h(j)=h(j-1);
end;
h(1)=a;
for i=1:n
y(i)=0;
for k=1:n
    y(i)=y(i)+(x(k)*h(k));%matrix multiplication
end;
a=h(n);
for j=n:-1:2
    h(j)=h(j-1);
end;
h(1)=a;
end;
subplot(221);
stem(y);% To plot y
disp('circular convolution of two sequences:');disp(y);
xlabel('time');
ylabel('amplitude');
```

RESULT

The MATLAB program to to find circular convolution of given sequences is executed and output is obtained.

ALGORITHM

- Start
- Enter the input sequence
- Find the length of the sequence
- Find DFT using formula
- Find IDFT using formula
- Stop

EXPERIMENT NO: 7

DFT AND IDFT

AIM

Write a MATLAB program to find DFT and IDFT of input sequence

MATLAB FUNCTIONS USED

• ZEROS: Create array of all zeros

• ABS( ) : abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.

• GRID ON : Grid lines for 2-D and 3-D Plots

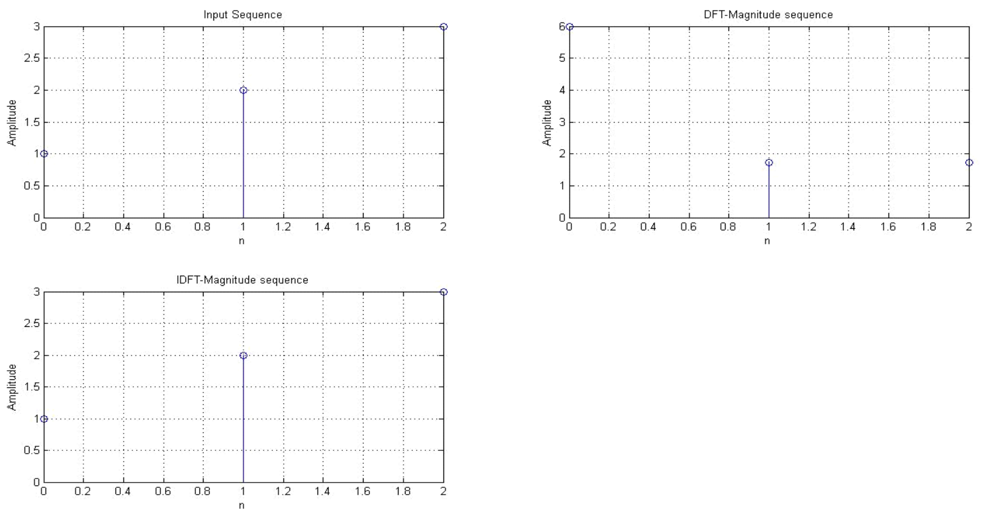
• INPUT( ): Request user input.

PROGRAM

```
% compute the N point DFT of the sequence
clc;
clear all;
N=input( 'Enter the value of N=  ');
x= input ('Enter the sequence...');
j= sqrt(-1);
xk=zeros(1,N);
for k=0:1:N-1 %value of k varies from 0 to N-1
   for n=0:1:N-1 %value of n varies from 0 to N-1
      xk(k+1)=xk(k+1)+x(n+1)*exp(-j*2*pi*k*n/N); %DFT using formula
   end
end
%p=fft(xn,N) %using function
wk=0:1:N-1;
subplot(2,2,1);
```

OUTPUT

```
Enter the value of N=  3

Enter the sequence...[1 2 3]

The DFT of the sequence is ,

xk =

  6.0000             -1.5000 + 0.8660i  -1.5000 - 0.8660i

The magnitude sequence is ,

magxk =

6.0000    1.7321    1.7321

The IDFT of the sequence is ,

xn =

1.0000 - 0.0000i   2.0000 - 0.0000i   3.0000 + 0.0000i

The magnitude sequence is ,

magxn =

1.0000    2.0000    3.0000
```



```
stem(wk,x);%plotting input sequence
xlabel('n');
ylabel('Amplitude');
title('Input Sequence');
grid on;
 disp('The DFT of the sequence is ,');xk
disp('The magnitude sequence is ,'); magxk=abs(xk)
wk=0:1:N-1;
subplot(2,2,2);stem(wk,magxk);% plotting magnitude sequence
xlabel('n');
ylabel('Amplitude');
title('DFT-Magnitude sequence');
grid on;
 %IDFT sequence generation
j= sqrt(-1);
xn=zeros(1,N);
for n=0:1:N-1 %n varies from 0 to N-1
  for k=0:1:N-1 %k varies from 1 to N-1
          xn(n+1)=xn(n+1)+(xk(k+1)*exp(j*2*pi*k*n/N))/N; %IDFT using
formula
    end
end
disp('The IDFT of the sequence is ,');xn
disp('The magnitude sequence is ,'); magxn=abs(xn)
wn=0:1:N-1;
subplot(2,2,3);stem(wn,magxn);% plotting magnitude sequence
xlabel('n');
ylabel('Amplitude');
title('IDFT-Magnitude sequence');
grid on;
```

RESULT

The MATLAB program to to find DFT and IDFT of input sequence is executed and output is obtained.

ALGORITHM

1.Start

2.Enter two sequences

3.Find length of sequences

4.Find length of output sequence

5.Find the fft of first sequence

6.Find the fft of second sequence

7.Find convolution

8.Find ifft for obtaining linear convolution

9.Plot inputs

10.Plot output

11. Stop

EXPERIMENT NO: 8

CONVOLUTION USING FFT

AIM

Write a MATLAB program to find the linear convolution using FFT.

MATLAB FUNCTIONS USED

• FFT: Discrete Fourier Transform. FFT(X) is the discrete Fourier transform(DFT) of vector X. FFT(X,N): is the N-point FFT, padded with zeros if X has less than N points and truncated if it has more.

• IFFT: Inverse discrete Fourier transform. IFFT(X) is the inverse discrete Fourier transform of X. IIF(X,N) is the N-point inverse transform.

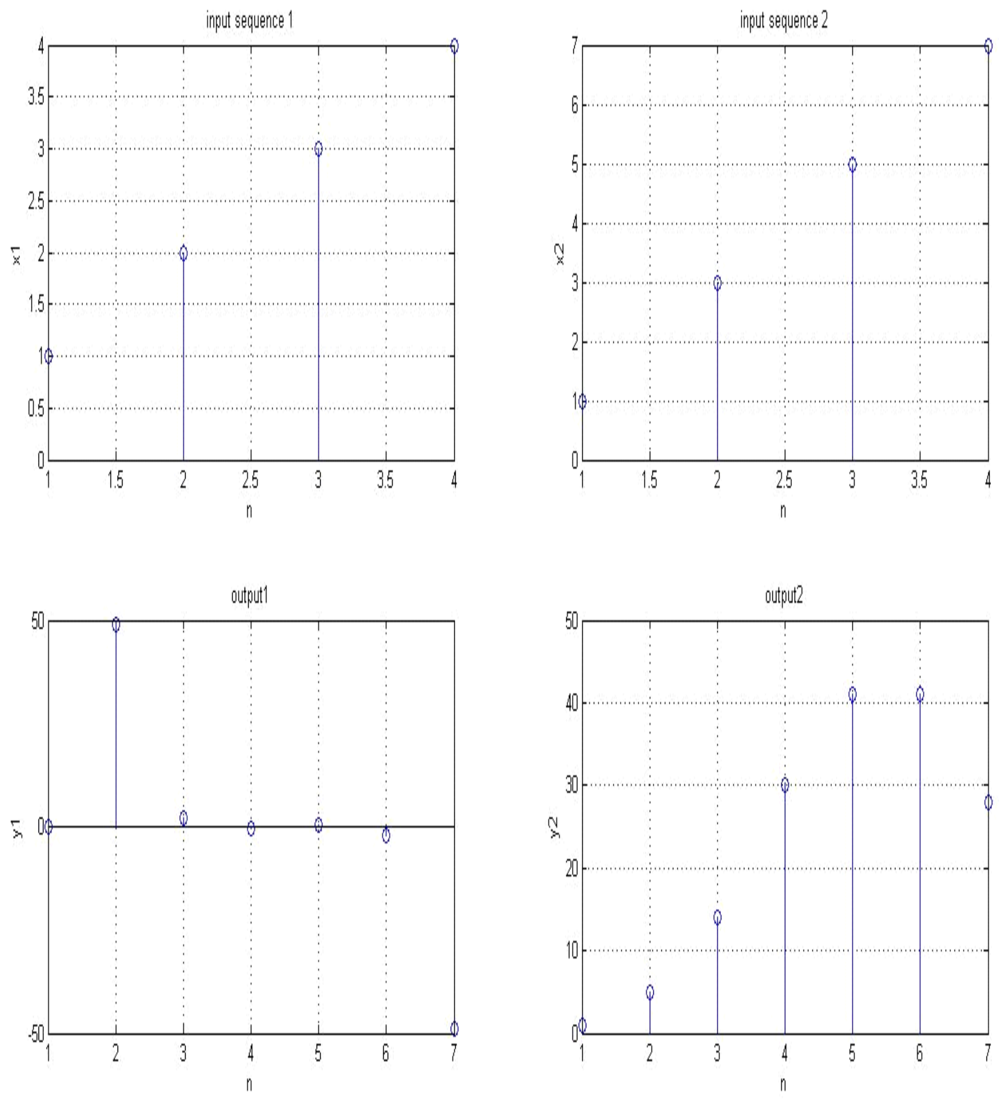
• LENGTH : n = length(X) returns the size of the longest dimension of X. If X is a vector, this is the same as its length.

PROGRAM

```
% program to find linear convolution
clc;
clear all;
close all;
% linear convolution using dft and idft
x1=input('enter the first sequence:');
x2=input('enter the second sequence:');
n1=length(x1);%length of x1 is enter to n1
n2=length(x2);%length of x2 is enter to n2
n=n1+n2-1;%calculate the length of output sequence
x3=fft(x1,n);%find the fft of x1
x4=fft(x2,n);%find the fft of x2
y1=(x3.*x4);%convolution of x3 and x4
```

OUTPUT

```
enter the first sequence:[1 2 3 4]

enter the second sequence:[1 3 5 7]

Linear convolution of 1st and 2nd sequences :

    1.0000    5.0000   14.0000   30.0000   41.0000   41.0000   28.0000
```



```
y2=ifft(y1,n);%calculate ifft of y1
subplot(2,2,1);
stem(x1);%Discrete plot of x1
title('input sequence 1');%title of plot x1
xlabel('n');%variable along x axis
ylabel('x1');%variable along y axis
grid on;
subplot(2,2,2);
stem(x2);%Discrete plot of x2
title('input sequence 2');%title of plot x2
xlabel('n');%variable along x axis
ylabel('x2');%variable along y axis
grid on;
subplot(2,2,3);
stem(y1);%Discrete plot of y1
title('output1');%title of plot y1
xlabel('n');%variable along x axis
ylabel('y1');%variable along y axis
grid on;
subplot(2,2,4);
disp('Linear convolution of 1st and 2nd sequences : ');
disp(y2);
stem(y2);%Discrete plot of y2
title('output2');%title of plot y2
xlabel('n');%variable along x axis
ylabel('y2');%variable along y axis
grid on;
```

RESULT

The MATLAB program to find the linear convolution using FFT was executed and output obtained correctly**.**

ALGORITHM

- Start
- Enter the input sequence
- Find the length of the sequence
- Find DCT using formula
- Find IDCT using formula
- Stop

EXPERIMENT NO: 9

DCT AND IDCT

AIM

Write a MATLAB program to find the DCT and IDCT for the given input.

MATLAB FUNCTIONS USED

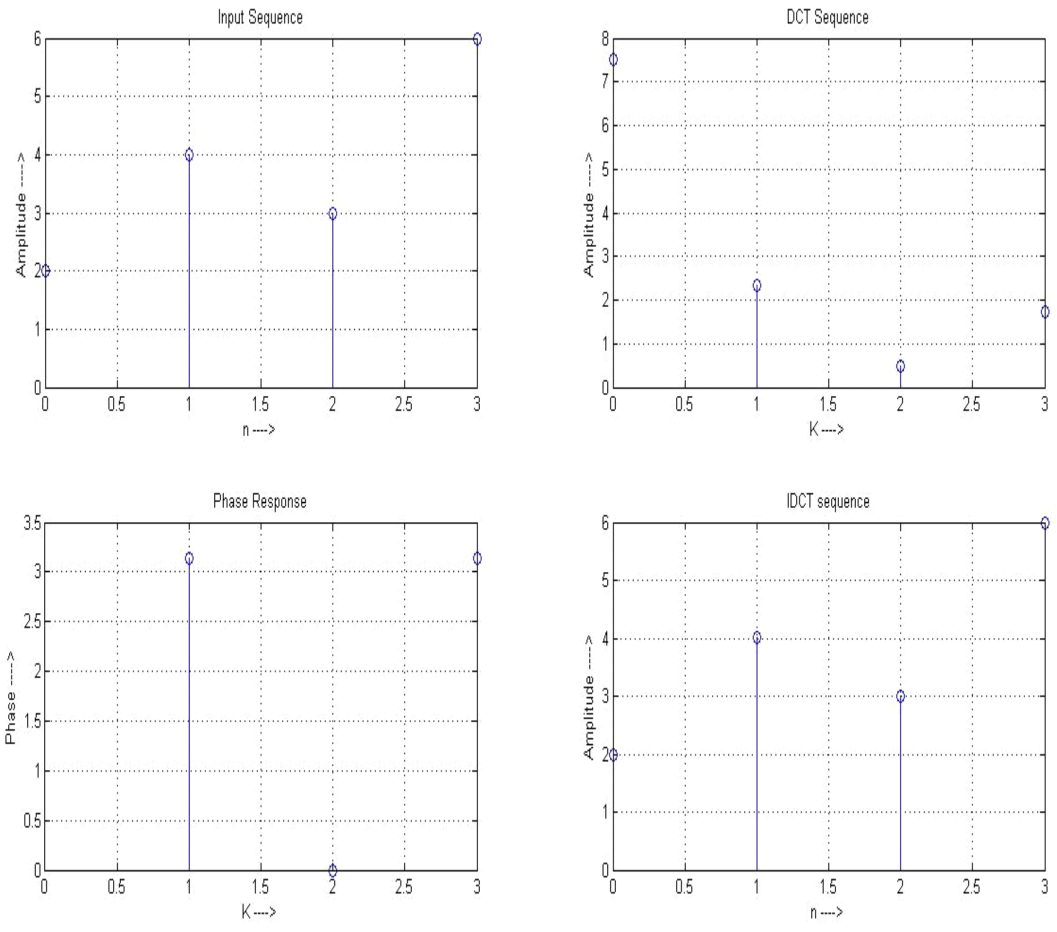
• ABS( ) : abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.
• GRID ON : Grid lines for 2-D and 3-D Plots
• INPUT( ): Request user input
• LENGTH: Length of vector. LENGTH(X) returns the length of vector X.
• ANGLE : Phase angle. P = angle(Z) returns the phase angles, in radians, for each element of complex array Z. The angles lie between +π and –π.

PROGRAM

```
Clc;
x = input('Enter the sequence = ');
l = length(x);%length of the vector
for k = 1:l %value of k varies from 1 to l
   %The series is indexed from n = 1 and k = 1 instead of the usual n = 0
   %and k = 0 because MATLAB® vectors run from 1 to l instead of from 0 to
%l-1.
   y(k) = 0;
if (k==1)
      w(k) = 1/sqrt(l);  %if k=1
   else
```

OUTPUT

```
Enter the sequence = [2 4 3 6]

DCT Sequence =

    7.5000    2.3425    0.5000    1.7357

Phase =

         0    3.1416         0    3.1416

IDCT Sequence =

    2.0000    4.0000    3.0000    6.0000
```



```
w(k) = sqrt(2/l); %if n not equal to 1

    end

    for n = 1:l  %value of n varies from 1 to l

        y(k) = y(k)+x(n)*cos(pi*(2*n-1)*(k-1)/(2*l));%DCT using formula

    end

    y(k) = y(k)*w(k);

end

t = 0:l-1;  %t varies from 0 to l-1

subplot(2,2,1);

stem(t,x);  %plotting input sequence

ylabel('Amplitude ---->');

xlabel('n ---->');

title('Input Sequence');

grid on;

magnitude = abs(y); % Find the magnitudes of individual DCT points

disp('DCT Sequence = ');

disp(magnitude);

%code block to plot the DCT sequence

t = 0:l-1;

subplot(2,2,2);

stem(t,magnitude);

ylabel('Amplitude ---->');

xlabel('K ---->');
```

```
title('DCT Sequence');
grid on;
phase = angle(y); % Find the phases of individual DCT points
disp('Phase = ');
disp(phase);
%code block to plot the phase response
t = 0:l-1;
subplot(2,2,3);
stem(t,phase);
ylabel('Phase ---->');
xlabel('K ---->');
title('Phase Response');
grid on;
for n = 1:l  %value of n varies from 1 to l
    X(n) = 0;
    if (n==1)
        w(n) = 1/sqrt(l);  %if n=1
    else
        w(n) = sqrt(2/l);  %if n not equal to 1
    end
    for k = 1:l
        X(n) = X(n)+w(k)*y(k)*cos(pi*(2*n-1)*(k-1)/(2*l));%IDCT using formula
    end
```

```
end
t = 0:l-1;
subplot(2,2,4);
stem(t,X);%plotting idct sequence
disp('IDCT Sequence = ');
disp(X);
ylabel('Amplitude ---->');
xlabel('n ---->');
title('IDCT sequence');
grid on;
```

RESULT

The MATLAB program to to find the DCT and IDCT for the given input was executed and output obtained correctly.

ALGORITHM

- Start
- Enter the input sequence
- Find the length of the sequence
- Find FFT using formula
- Find IFFT using formula
- Stop

EXPERIMENT NO: 10

FFT AND IFFT

AIM

Write a MATLAB program to find the FFT and IFFT of the given input.

MATLAB FUNCTIONS USED

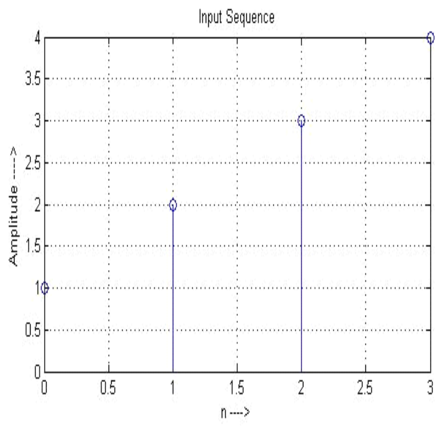
- ABS( ) : abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.
- GRID ON : Grid lines for 2-D and 3-D Plots
- INPUT( ): Request user input
- LENGTH: Length of vector. LENGTH(X) returns the length of vector X.

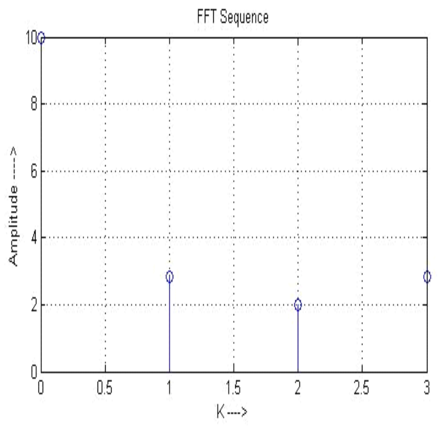
PROGRAM

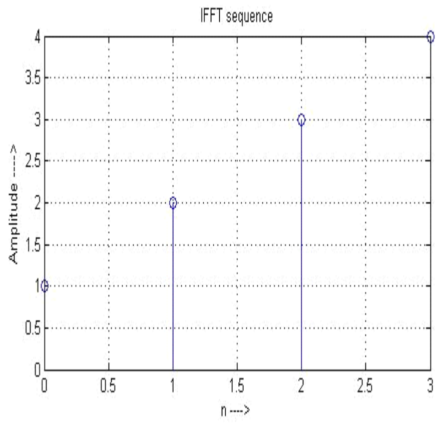
```
clc;
clear all;
x = input('Enter the input sequence = ');
N = length(x); %returns length of x
for k = 1:N  %k varies from 1 to N
y(k) = 0;
for n = 1:N   %n varies from 1 to N
y(k) = y(k)+x(n)*exp(-1i*2*pi*(k-1)*(n-1)/N);%FFT using formula
end
end
%code block to plot the input sequence
t = 0:N-1;
```

OUTPUT

```
Enter the input sequence = [1 2 3 4]


FFT Sequence =

   10.0000    2.8284    2.0000    2.8284


IFFT Sequence =

   1.0000 - 0.0000i   2.0000 - 0.0000i   3.0000 - 0.0000i   4.0000 + 0.0000i
```







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

stem(t,x);

ylabel('Amplitude ---->');

xlabel('n ---->');

title('Input Sequence');

grid on;

magnitude = abs(y); % Find the magnitudes of individual FFT points

disp('FFT Sequence = ');

disp(magnitude);

%code block to plot the FFT sequence

t = 0:N-1;

subplot(2,2,2);

stem(t,magnitude);

ylabel('Amplitude ---->');

xlabel('K ---->');

title('FFT Sequence');

grid on;

R = length(y); %to find the length of y

for n = 1:R  %n varies from 1 to R

x1(n) = 0;

for k = 1:R  %k varies from 1 to R

    x1(n) = x1(n)+(1/R)*y(k)*exp(1i*2*pi*(k-1)*(n-1)/R);%IFFT using formula

end
```

```
end
%code block to plot the IFFT sequence
t = 0:R-1;
subplot(2,2,3);
stem(t,x1);
disp('IFFT Sequence = ');
disp(x1);
ylabel('Amplitude ---->');
xlabel('n ---->');
title('IFFT sequence');
grid on;
```

RESULT

The MATLAB program to find the FFT and IFFT of the given input sequence were executed and results are obtained.

ALGORITHM:

- Start
- Find the passband and stopband edge frequency.
- Find the order and cut-off frequency of butterworth and chebychev.
- Design low pass butterworth and high pass chebychev filter.
- Plot the digital frequency response of both filters.
- Stop

EXPERIMENT NO: 11

IIR FILTER DESIGN

AIM:

Write a MATLAB program to design an IIR Filter(Butterworth and Chebychev)

MATLAB FUNCTIONS USED:

• ABS( ) : abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.

• GRID ON : Grid lines for 2-D and 3-D Plots

• INPUT( ): Request user input

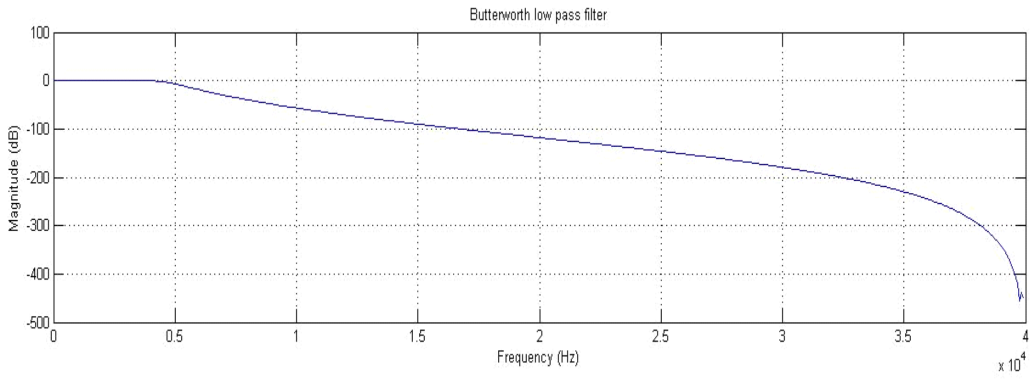
• BUTTORD : Butterworth filter order selection.[N, Wn] = BUTTORD(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Butterworth filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample)

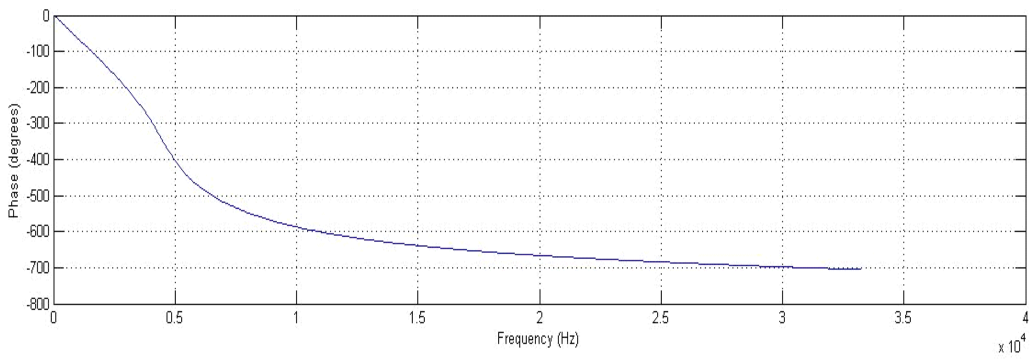
• BUTTER : Butterworth digital and analog filter design. [B,A] = BUTTER(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator)The coefficients are listed in descending powers of z. The cutoff frequency Wn must be 0.0 < Wn < 1.0, with 1.0 corresponding to half the sample rate.

• CHEBY1: Chebyshev Type I digital and analog filter design. [B,A] = CHEBY1(N,R,Wp) designs an Nth order lowpass digital Chebyshev filter with

OUTPUT:

```
Enter the pass band frequency fp   = 4000
Enter the stop band frequency fs   = 8000
Enter the pass band attenuation rp = .5
Enter the stop band attenuation rs = 40
Enter the sampling frequency f     = 80000

n1 =

     8
wn1 =

    0.1151
n2 =

     5
wn2 =

    0.1000
N =

   512
```





R decibels of peak-to-peak ripple in the passband. CHEBY1 returns the filter coefficients in length N+1 vectors B (numerator)

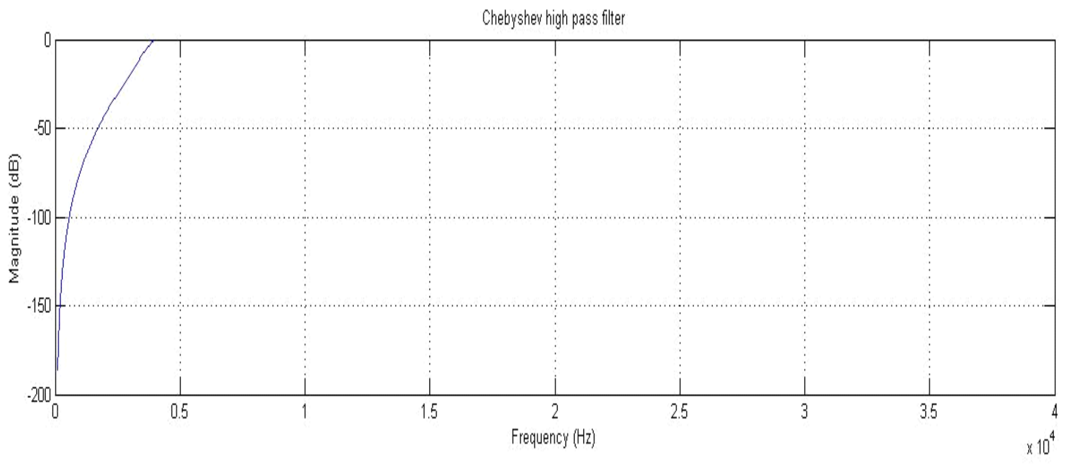
and A (denominator). The passband-edge frequency Wp must be 0.0 < Wp < 1.0 with 1.0 corresponding to half the sample rate.

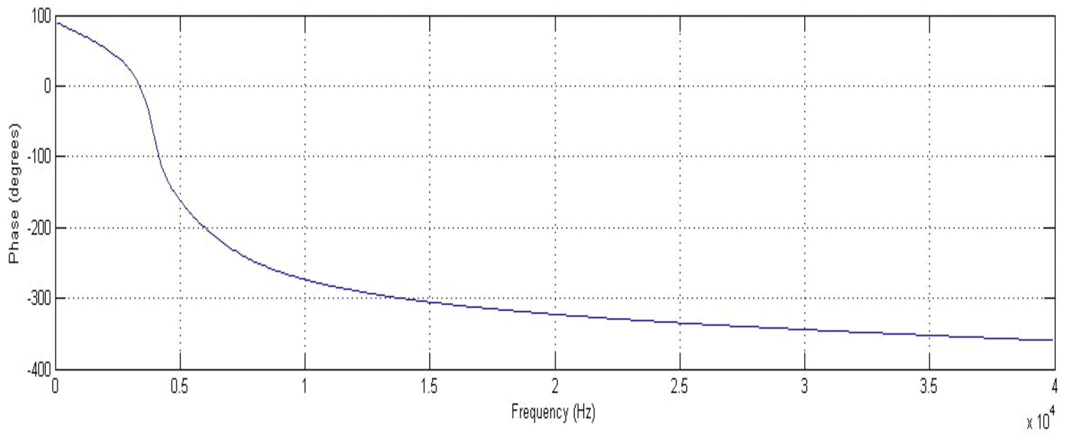
• CHEB1ORD : Chebyshev Type I filter order selection. [N, Wp] = CHEB1ORD(Wp, Ws, Rp, Rs) returns the order N of the lowest order digital Chebyshev Type I filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. Wp and Ws are the passband and stopband edge frequencies, normalized from 0 to 1 (where 1 corresponds to pi radians/sample).

• FREQZ : Digital filter frequency response. [H,W] = FREQZ(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter

PROGRAM:

```
clear all;
close all;
fp=input('Enter the pass band frequency fp   = ');
fs=input('Enter the stop band frequency fs   = ');
rp=input('Enter the pass band attenuation rp = ');
rs=input('Enter the stop band attenuation rs = ');
f=input ('Enter the sampling frequency f     = ');
%fp=4000;
%fs=8000;
%rp=.5
%rs=40;
%f=80000;
```


Chebyshev high pass filter
Magnitude (dB)
0
-50
-100
-150
-200
0
0.5
1
1.5
2
2.5
3
3.5
4
Frequency (Hz)
x 10^4


Phase (degrees)
100
0
-100
-200
-300
-400
0
0.5
1
1.5
2
2.5
3
3.5
4
Frequency (Hz)
x 10^4

```
 wp=2*fp/f;  %passband edge frequency
ws=2*fs/f;  %stopband edge frequency
 [n1, wn1]=buttord(wp,ws,rp,rs)    %returns the order n1 of the lowest order
%digital butterworth filter and the cut-off frequency wn1
[n2, wn2]=cheb1ord(wp,ws,rp,rs)   %returns the order n2 of the lowest order
%digital chebyshev filter and the cut-off frequency wn2
 N=512
 [b, a]=butter(n1,wn1,'low'); %designs an n1th order low pass digital
%butterworth filter and returns the filter co-efficientsin length n1+1 vectors
%b(numerator) and a(denominator)
 [c, d]=cheby1(n2,rp,wn2,'high');%designs an n2th order high pass digital
%chebyshev filter with rp decibels of peak-to-peak ripple in passband.
 %cheby1 returns the filter co-efficientsin length n2+1 vectors c(numerator) and
%d(denominator
figure;
freqz(b,a,N,f); %digital filter frequency response
title('Butterworth low pass filter');
 figure;
 freqz(c,d,N,f);  %digital filter frequency response
title('Chebyshev high pass filter');
```

RESULT:

Designed Butterworth Low Pass and Chebychev High Pass Filters.

ALGORITHM:

• Start

• Generate the ideal impulse response (Sinc function).

• Call window functions for - Rectangular, Hamming , Hanning and Kaiser windows

• Multiply them with ideal impulse response.

• Plot the frequency responses of these windowed sequences to obtain the FIR filter responses of the respective windows.

• Stop

EXPERIMENT NO: 12

FIR FILTER DESIGN

AIM:

Write a MATLAB program to design FIR Filter using Window method.

MATLAB FUNCTIONS USED:

• LENGTH: Length of vector. LENGTH(X) returns the length of vector X.

• FFT: Discrete Fourier Transform. FFT(X) is the discrete Fourier transform(DFT) of vector X. FFT(X,N) is the N-point FFT, padded with zeros if X has less than N points and truncated if it has more.

• ABS( ) : abs(X) returns an array Y such that each element of Y is the absolute value of the corresponding element of X.

• HANN : Hanning window. HANN(N) returns the N-point symmetric Hanning window in a column vector.

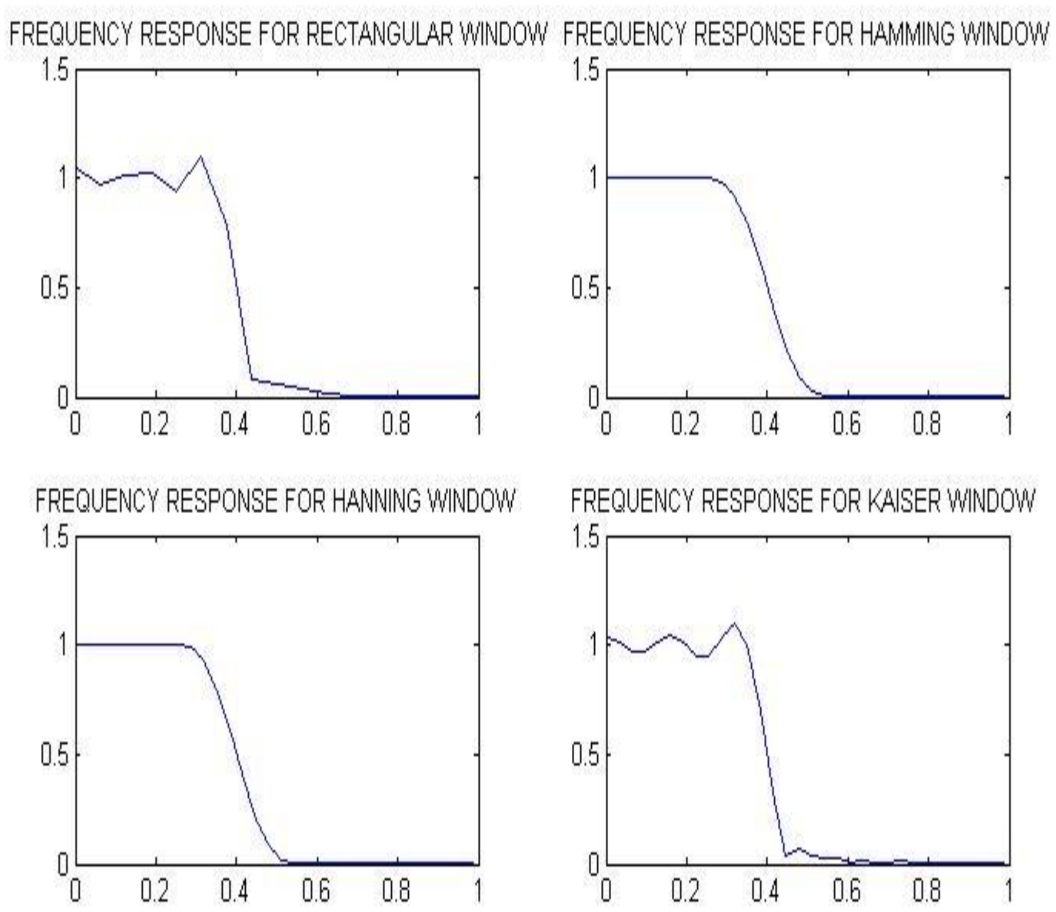
• HAMMING: Hamming window. HAMMING(N) returns the N-point symmetric Hamming window in a column vector.

• RECTWIN: Rectangular window. W = RECTWIN(N) returns the N-point rectangular window.

PROGRAM:

```
%Design a low pass filter with cutoff frequency 0.4Pi and order 25 using

%rectangular, hanning, hamming and Kaiser window.

 clear all;

clc;

wc=0.4*pi;
```

OUTPUT:


FREQUENCY RESPONSE FOR RECTANGULAR WINDOW
FREQUENCY RESPONSE FOR HAMMING WINDOW
FREQUENCY RESPONSE FOR HANNING WINDOW
FREQUENCY RESPONSE FOR KAISER WINDOW
1.5
1
0.5
0
0.2
0.4
0.6
0.8

```
N=25;
e=.001;% to avoid indetrminate  form
a=(N-1)/2;
n=0:1:N-1;
hd=sin(wc*(n-a+e))./(pi*(n-a+e));

wr=rectwin(N);
hn=hd.*wr';
w=0:pi/16:pi;
h=freqz(hn,1,w);
subplot(2,2,1);plot(w/pi,abs(h));
title(' FREQUENCY RESPONSE FOR RECTANGULAR WINDOW');

wh=hamming(N);
hn2=hd.*wh';
w2=0:.1:pi;
h2=freqz(hn2,1,w2);subplot(2,2,2);
plot(w2/pi,abs(h2));
title(' FREQUENCY RESPONSE FOR HAMMING WINDOW ');

whn=hanning(N);
hn3=hd.*whn';
w3=0:.1:pi;
h3=freqz(hn3,1,w3);
subplot(2,2,3);plot(w3/pi,abs(h3));
title(' FREQUENCY RESPONSE FOR HANNING WINDOW ');

wk=kaiser(N);
hn4=hd.*wk';
```

```
w4=0:.1:pi;
h4=freqz(hn4,1,w4);;subplot(2,2,4);
plot(w4/pi,abs(h4));
title('FREQUENCY RESPONSE FOR KAISER WINDOW ');
```

RESULT:

Designed FIR filter using window methods like rectangular,hamming and hanning windows.

EXPERIMENT NO: 13

FILTER DESIGN USING FILTER DESIGN TOOL

AIM:

FIR and IIR filter design using Filter Design Toolbox.

Filter Design and Analysis using FDATool of MATLAB:

The Filter Design and Analysis Tool (FDATool) is a powerful user interface for designing and analyzing filters quickly. FDATool enables you to design digital FIR or IIR filters by setting filter specifications, by importing filters from your MATLAB workspace, or by adding, moving or deleting poles and zeros. FDATool also provides tools for analyzing filters, such as magnitude and phase response and pole-zero plots. FDATool seamlessly integrates additional functionality from other MathWorks products.

Use FDATOOL in matlab.

If you type

```
>>fdatool
```

in command window, FDAtool will be opened. There you can select FIR or IIR filter, order of filter and cutoff frequency of a filter (either HPF, LPF or BPF). That code will automatically generate .m file for you.

GETTING STARTED:

Type fdatool at the MATLAB command prompt:

```
>>fdatool
```

A Tip of the Day dialog displays with suggestions for using FDATool. Then, the GUI diplays with a default filter.

The GUI has three main regions:

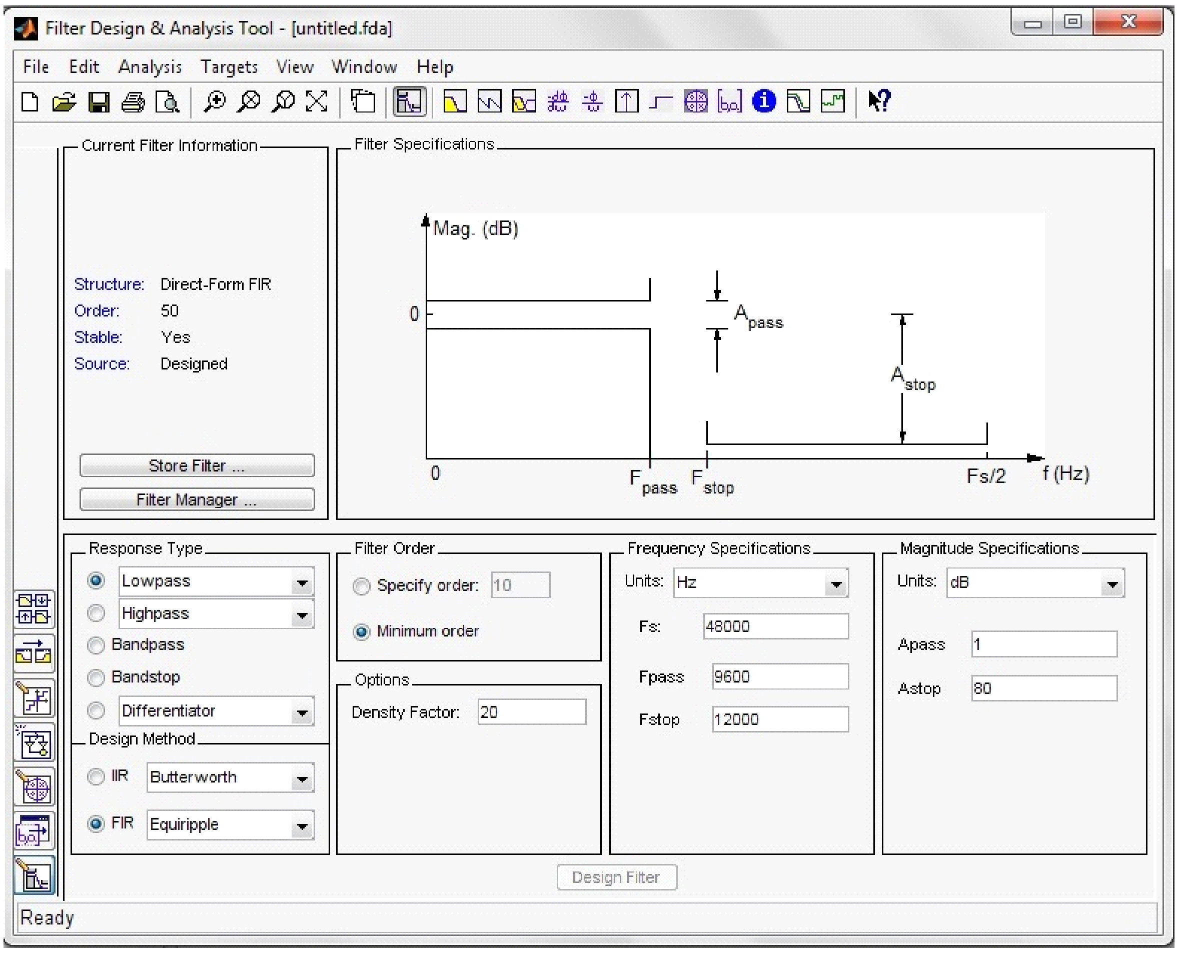
1.The Current Filter Information region

2.The Filter Display region and

3.The Design panel

The upper half of the GUI displays information on filter specifications and responses for the current filter. The Current Filter Information region, in the upper left, displays filter properties, namely the filter structure, order, number of sections used and whether the filter is stable or not. It also provides access to the Filter manager for working with multiple filters.

The Filter Display region, in the upper right, displays various filter responses, such as, magnitude response, group delay and filter coefficients.The lower half of the GUI is the interactive portion of FDATool. The Design Panel, in the lower half is where you define your filter specifications. It controls what is displayed in the other two upper regions. Other panels can be displayed in the lower half by using the sidebar buttons.



Current Filter Information

Structure: Direct-Form FIR
Order: 50
Stable: Yes
Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: IIR Butterworth [✓ FIR Equiripple]

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Density Factor: 20

Frequency Specifications: Units: Hz; Fs: 48000; Fpass: 9600; Fstop: 12000

Magnitude Specifications: Units: dB; Apass: 1; Astop: 80

DESINING FILTER:

- **IIR FILTER**

We will design a butterworth low pass filter with following specifications:

Pass band frequency: 4000

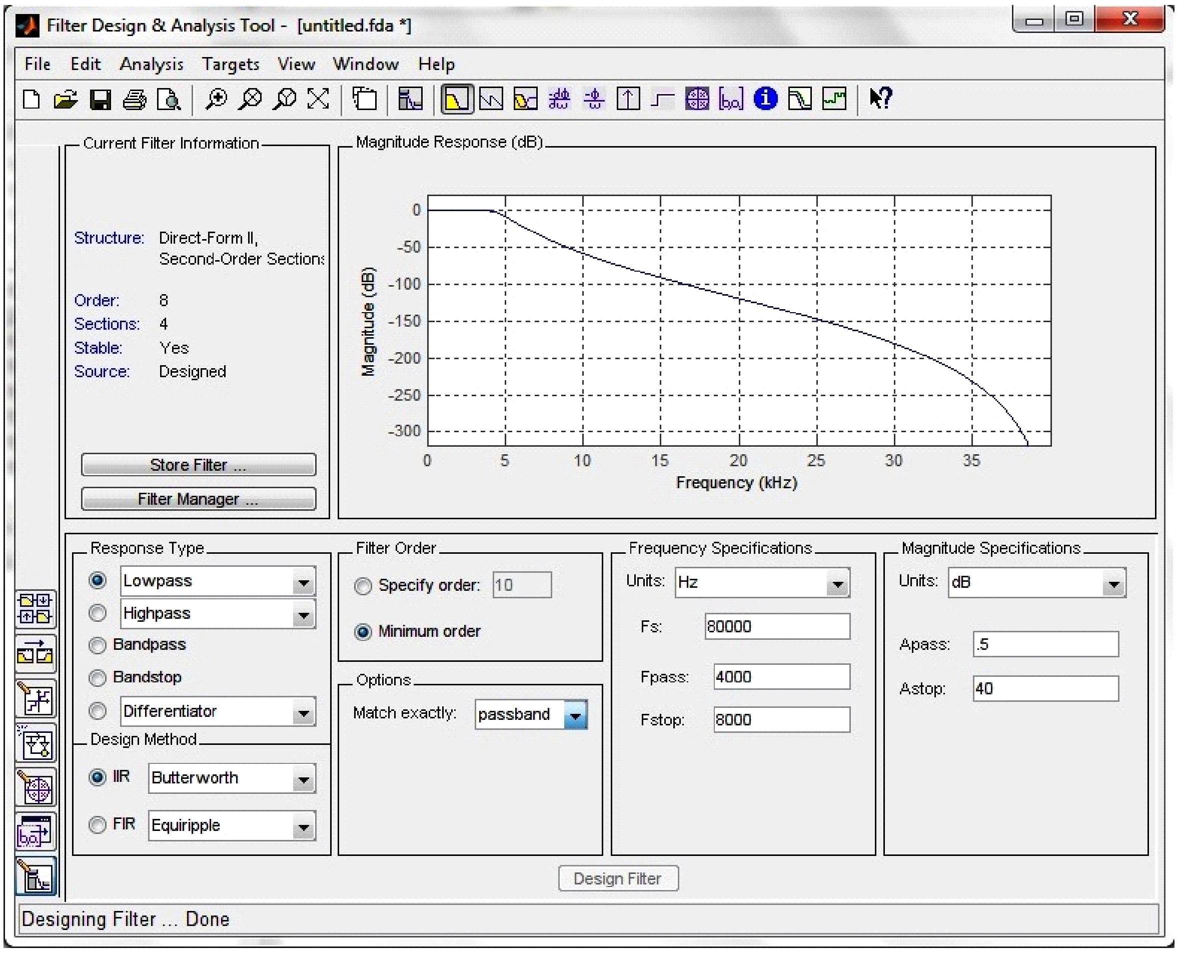
Stop band frequency: 8000

Pass band ripple: .5

Stop band ripple: 40

Sampling Frequency: 80000

To implement this design, we will use the following specifications:



Current Filter Information

Structure: Direct-Form II, Second-Order Sections

Order: 8

Sections: 4

Stable: Yes

Source: Designed

Response Type: [✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method: [✓ IIR] Butterworth; FIR Equiripple

Filter Order: Specify order: 10 [✓ Minimum order]

Options: Match exactly: passband

Frequency Specifications: Units: Hz; Fs: 80000; Fpass: 4000; Fstop: 8000

Magnitude Specifications: Units: dB; Apass: .5; Astop: 40

Designing Filter ... Done

1. Select **Lowpass** from the dropdown menu under **Response Type** and **BUTTERWORTH** under **IIR Design Method**. In general, when you change the Response Type or Design Method, the filter parameters and Filter Display region update automatically.

2. Select **Minimum order** in the **Filter Order.**

3. Select **Hz** in the Units pull down menu in the **Frequency Specifications** area. Then input sampling frequency,Pass band frequency and stop band frequency.

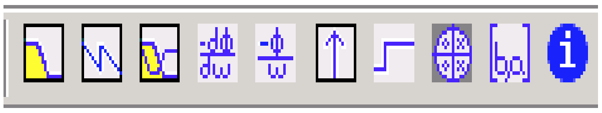
4. Select Db in the units pull down menu in the Magnitude Specifications area.Enter 0.5 for Apass and 40 for Astop in the Magnitude Specifications area.

5. After setting the design specifications, click the Design Filter button at the bottom of the GUI to design the filter.

The magnitude response of the filter is displayed in the Filter Analysis area after the coefficients are computed

Viewing other Analyses:

Once you have designed the filter, you can view the following filter analyses in the display window by clicking any of the buttons on the toolbar:



In order from left to right, the buttons are

1. Magnitude response
2. Phase response
3. Magnitude and Phase responses
4. Group delay response
5. Phase delay response
6. Impulse response
7. Step response
8. Pole-zero plot
9. Filter Coefficients
10. Filter Information

- **FIR FILTER**

We will design a FIR low pass filter using hanning window with following specifications:

Sampling Frequency: 80000

Cut-off Frequency : 4000

To implement this design, we will use the following specifications:

1. Select **Lowpass** from the dropdown menu under **Response Type** and **Window** under **FIR Design Method**. In general, when you change the Response Type or Design Method, the filter parameters and Filter Display region update automatically.

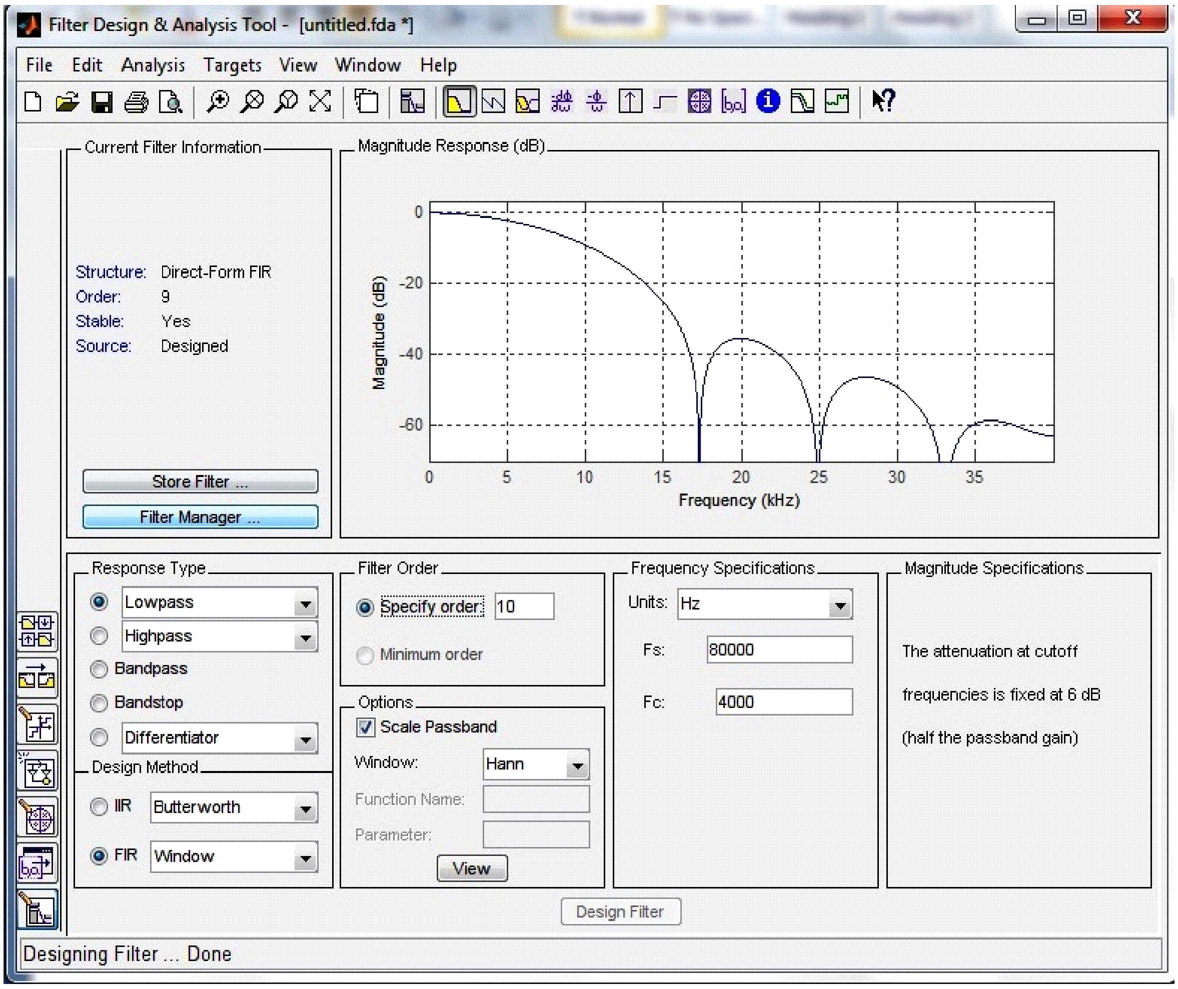
2. Enter **10** in **Specify order** under the **Filter Order.**

3. Select **Hann** in **Window.**

4. Select **Hz** in the Units pull down menu in the **Frequency Specifications** area. Then input sampling frequency and cut-off frequency.

5. After setting the design specifications, click the **Design Filter** button at the bottom of the GUI to design the filter.

The magnitude response of the filter is displayed in the Filter Analysis area after the coefficients are computed.



Structure: Direct-Form FIR
Order: 9
Stable: Yes
Source: Designed

Response Type: Lowpass
Design Method: FIR Window
Specify order: 10
Scale Passband ✓
Window: Hann
Units: Hz
Fs: 80000
Fc: 4000

The attenuation at cutoff frequencies is fixed at 6 dB (half the passband gain)

Designing Filter ... Done

RESULT: Designed FIR and IIR filter Using Filter Design Tool.