DSP LAB MANUAL

Prepared by

Dr. Md. Ekramul Hamid

Department of CSE, RU

CONTENTS

Sl. No . Experiment

1. Introduction to MATLAB and its basic commands

2. To develop programs for generating elementary signal functions like unit sample, unit step, exponential, ramp sequences, sinusoidal, random and periodic signal.

3. Generation of basic signals and illustration of sampling process using Matlab

4. Introduction to different operations on sequences

5. Understanding of aliasing effect of discrete time signals in MATLAB.

6. To develop the program for finding the convolution between two sequences.

7. To develop the program for finding the Correlation of two sequences.

8. To develop the program for finding the DFT.

9. To develop the program for finding the magnitude and phase response of system described by system function H(s).

10. To find the frequency response of analog LP/HP filter.

11. To develop the program for designing Low pass Type 1 Chebyshev filter having passband defined from 0-40 Hz and stopband in the range of 150-500Hz having less than 3 dB of ripple in the passband and atleast 60dB of attenuation in the stopband.

12. Design FIR filter using windowing technique.

13. Design IIR filter

14. Power density spectrum of a sequence.

15. The objective of this program is To Perform upsampling on the Given Input Sequence.

16. The objective of this program is To Perform Decimation on the Given Input Sequence.

17. Analysis of Z transform and Inverse Z Transform.

18. Application on speech signal processing (students will prepare project based on this experiment)

(a) Read Speech sound file

(b) Show the effect of sampling, e.g. over, under, aliasing effect

(c) Show the effect of filtering- low pass, windowing

(d) Reconstruction of signal

(e)Add white and color noise to speech at particular SNR- show waveform, spectrogram, etc

(f)Show the FFT with changing different parameters.

(g) Show the effect of filters on noisy speech- adaptive

(h) Calculation of SNR

19. Experiment with MDA DSP kit..

EXPERIMENT NO: 01

Title: Introduction to MATLAB and its basic commands

Objective: MATLAB: This is a very important tool used for making long complicated calculations and plotting graphs of different functions depending upon our requirement. Using MATLAB an m-file is created in which the basic operations are performed which leads to simple short and simple computations of some very complicated problems in no or very short time. To familiarise with MATLAB software, general functions and signal processing toolbox functions.

Task: Learn Matlab and Practice Matlab Commands

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Reference: DSP by Prokis

Some important commands in MATLAB:

Help	List topics on which help is available
Help command name	Provides help on the topic selected
Demo	Runs the demo program
Who	Lists variables currently in the workspace
Whos	Lists variables currently in the workspace with their size
Clear	Clears the workspace, all the variables are removed
Clear x,y,z	Clears only variables x,y,z
Quit	Quits MATLAB

Some very important functions performed by MATLAB are given as follows:

_ Matrix computations
_ Vector Analysis
_ Differential Equations computations
_ Integration is possible
_ Computer language programming
_ Simulation
_ Graph Plotation
_ 2-D & 3-D Plotting

Benefits:

Some Benefits of MATLAB are given as follows:
_ Simple to use
_ Fast computations are possible
_ Wide working range
_ Solution of matrix of any order
_ Desired operations are performed in matrices
_ Different Programming languages can be used
_ Simulation is possible

Basic Commands:

Some basic MATLAB commands are given as follows:

Addition:
A+B

Subtraction:
A-B

Multiplication:
A*B

Division:
A/B

Power:
A^B

Power Of each Element individually:
A.^B

Range Specification:
A:B

Square-Root:
A=sqrt(B)

Where A & B are any arbitrary integers

Basic Matrix Operations:

This is a demonstration of some aspects of the MATLAB language.

Creating a Vector:

Let_s create a simple vector with 9 elements called a.

```
a = [1 2 3 4 6 4 3 4 5]
a =
 1    2    3    4    6    4    3    4    5
```

Now let's add 2 to each element of our vector, a, and store the result in a new vector.

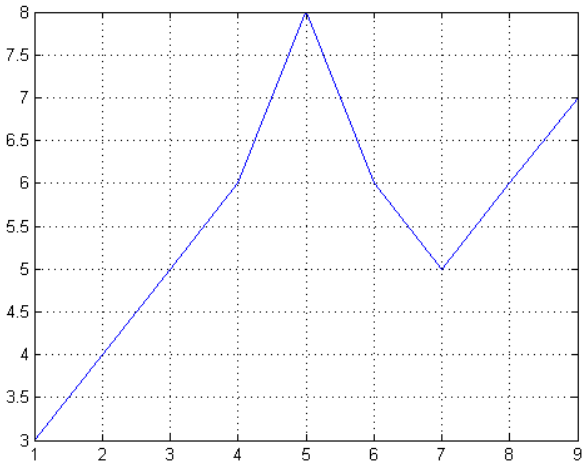
Notice how MATLAB requires no special handling of vector or matrix math.

Adding an element to a Vector:

```
b = a + 2
b =
 3    4    5    6    8    6    5    6    7
```

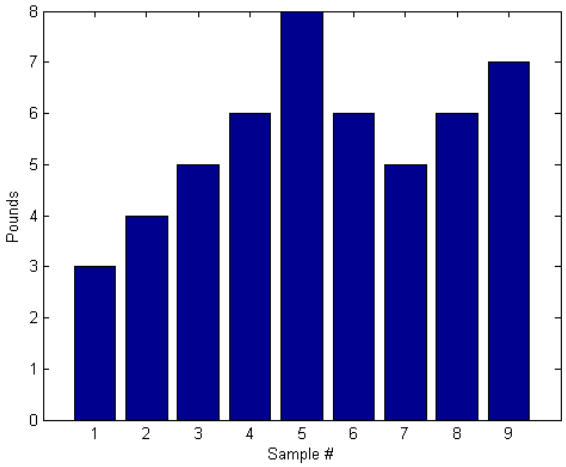
Plots and Graphs:

Creating graphs in MATLAB is as easy as one command. Let's plot the result of our vector addition with grid Lines.



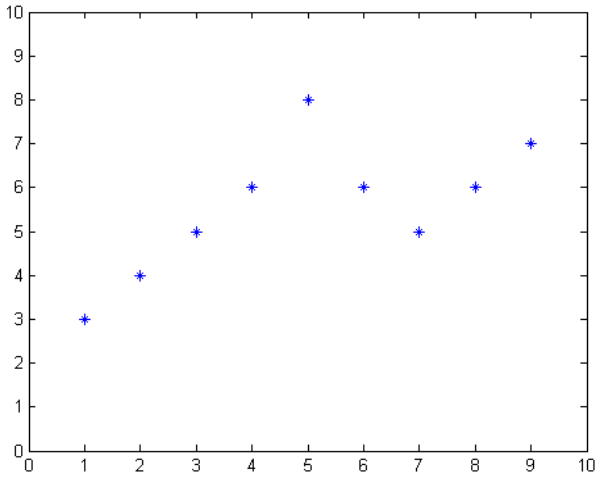

```
Plot (b)
grid on
```

MATLAB can make other graph types as well, with axis labels.




```
bar(b)
xlabel('Sample #')
ylabel('Pounds')
```

MATLAB can use symbols in plots as well. Here is an example using stars to mark the points. MATLAB offers a variety of other symbols and line types.




```
plot(b,'*')
axis([0 10 0 10])
```

One area in which MATLAB excels is matrix computation.

Creating a matrix:
Creating a matrix is as easy as making a vector, using semicolons (;) to separate the rows of a matrix.

```
A = [1 2 0; 2 5 -1; 4 10 -1]
A =
    1     2     0
    2     5    -1
    4    10    -1
```

Adding a new Row:

```
B(4,:)=[7 8 9]
ans=
    1     2     0
    2     5    -1
    4    10    -1
    7     8     9
```

Adding a new Column:

```
C(:,4)=[7 8 9]
ans=
    1     2     0   7
    2     5    -1   8
    4    10    -1   9
```

Transpose:
We can easily find the transpose of the matrix A.

```
B = A'
B =
    1     2     4
    2     5    10
    0    -1    -1
```

Matrix Multiplication:
Now let's multiply these two matrices together.

Note again that MATLAB doesn't require you to deal with matrices as a collection of numbers. MATLAB knows when you are dealing with matrices and adjusts your calculations accordingly.

```
C = A * B
C =
    5    12    24
   12    30    59
   24    59   117
```

Matrix Multiplication by corresponding elements:
Instead of doing a matrix multiply, we can multiply the corresponding elements

of two matrices or vectors using the ” .* ” operator.

```
C = A .* B

C =
     1     4     0
     4    25   -10
     0   -10     1
```

Inverse:

Let's find the inverse of a matrix ...

```
X = inv(A)
X =
     5     2    -2
    -2    -1     1
     0    -2     1
```

... and then illustrate the fact that a matrix times its inverse is the identity matrix.

```
I = inv(A) * A
I =
     1     0     0
     0     1     0
     0     0     1
```

MATLAB has functions for nearly every type of common matrix calculation.

Eigen Values:

There are functions to obtain eigenvalues ...

```
eig(A)
ans =
    3.7321
    0.2679
    1.0000
```

Singular Value Decomposition:

The singular value decomposition

```
svd(A)
ans =
   12.3171
    0.5149
    0.1577
```

Polynomial coefficients:

The "poly" function generates a vector containing the coefficients of the characteristic polynomial.

The characteristic polynomial of a matrix A is

```
p = round(poly(A))
p =
    1    -5     5    -1
```

We can easily find the roots of a polynomial using the roots function. These are actually the eigenvalues of the original matrix.

```
roots(p)
ans =
   3.7321
   1.0000
   0.2679
```

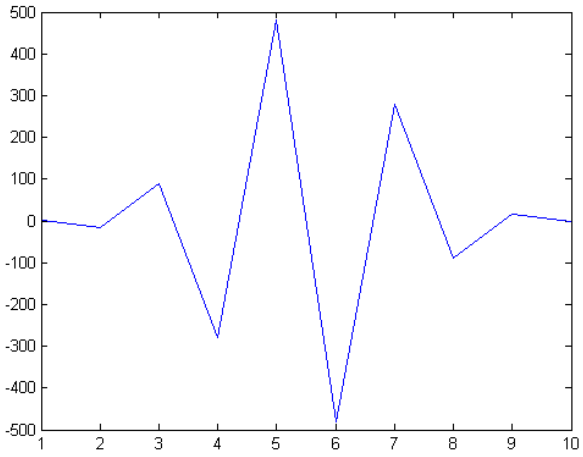
MATLAB has many applications beyond just matrix computation.

Vector Convolution:
To convolve two vectors ...

```
q = conv(p,p)
q =
    1   -10    35   -52    35   -10     1
```

... Or convolve again and plot the result.

```
r = conv(p,q)
plot(r);
r =
    1   -15    90  -278   480  -480   278   -90    15    -1
```



Matrix Manipulation:

We start by creating a magic square and assigning it to the variable A.
A = magic(3)
A =

8	1	6
3	5	7
4	9	2

Some of the frequently used built-in-functions in Signal Processing Toolbox

clc
(Remove items from workspace, freeing up system memory) clears all input and output from the Command Window display, giving "clean screen." After using clc, the scroll bar cannot be used to see the history of functions, but still the up arrow can be used to recall statements from the command history.

close
(Remove specified figure): close deletes the current figure or the specified figure(s). It optionally returns the status of the close operation.

xlabel, ylabel, zlabel (Label x-, y-, and z-axis) : 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.

xlabel('string') labels the x-axis of the current axes.

ylabel(...) and zlabel(...) label the y-axis and z-axis, respectively, of the current axes.

title(Add title to current axes) : Each axes graphics object can have one title. The title is located at the top and in the center of the axes.
title('string') outputs the string at the top and in the center of the current axes.
figure (create figure graphics object) : figure creates figure graphics objects. Figure objects are the individual windows on the screen in which MATLAB displays graphical output.

subplot (Create axes in tiled positions): 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 pane.

h = subplot(m,n,p) or subplot(mnp) breaks the figure window into an m-by-n matrix of small axes, selects the pth axes object for the current plot, and returns the axes handle. The axes are counted along the top row of the figure window, then the second row, etc. For example,
subplot(2,1,1), plot(income)

subplot(2,1,2), plot(outgo) plots income on the top half of the window and outgo on the bottom half.

stem (Plot discrete sequence data) : 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.
stem(Y) Plots the data sequence Y as stems that extend from equally spaced and automatically generated values along the x-axis. When Y is a matrix, stem plots all elements in a row against the same x value.
stem(X,Y) plots X versus the columns of Y. X and Y must be vectors or matrices of the same size. Additionally, X can be a row or a column vector and Y a matrix with length(X) rows.

bar(Plot bar graph (vertical and horizontal)) : A bar graph displays the values in a vector or matrix as horizontal or vertical bars.

bar(Y) draws one bar for each element in Y. If Y is a matrix, bar groups the bars produced by the elements in each row. The x-axis scale ranges from 1 up to length(Y) when Y is a vector, and 1 to size(Y,1), which is the number of rows, when Y is a matrix.

barh(...) and h = barh(...) create horizontal bars. Y determines the bar length. The vector x is a vector defining the y-axis intervals for horizontal bars.

plot (2-D line plot) :

plot(Y) Plots the columns of Y versus their index if Y is a real number. If Y is complex, plot(Y) is equivalent to plot(real(Y),imag(Y)). In all other uses of plot, the imaginary component is ignored.

plot(X1,Y1,...) Plots all lines defined by Xn versus Yn pairs. If only Xn or Yn is a matrix, the vector is plotted versus the rows or columns of the matrix,
depending on whether the vector's row or column dimension matches the matrix. If Xn is a scalar and Yn is a vector, disconnected line objects are created and plotted as discrete points vertically at Xn.

input (Request user input) : The response to the input prompt can be any MATLAB expression, which is evaluated using the variables in the current workspace.

user_entry = input('prompt') Displays prompt as a prompt on the screen, waits for input from the keyboard, and returns the value entered in user_entry. user_entry = input('prompt', 's') returns the entered string as a text variable rather than as a variable name or numerical value.

zeros (Create array of all zeros) :
B = zeros(n) Returns an n-by-n matrix of zeros. An error message appears if n is not a scalar.

ones (Create array of all ones) :
Y = ones(n) Rreturns an n-by-n matrix of 1s. An error message appears if n is not a scalar.

exp (Exponential) :
Y = exp(X) The exp function is an elementary function that operates element-wise on arrays. Its domain includes complex numbers.
Y = exp(X) returns the exponential for each element of X.

disp (Display text or array) :
disp(X) Displays an array, without printing the array name. If X contains a text string, the string is displayed. Another way to display an array on the screen is to type its name, but this prints a leading "X=," which is not always desirable. Note that disp does not display empty arrays.

conv (Convolution and polynomial multiplication) :
w = conv(u,v) convolves vectors u and v. Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of u and v.

xcorr (Cross-correlation) :
c = xcorr(x,y) returns the cross-correlation sequence in a length 2*N-1 vector, where x and y are length N vectors (N>1). If x and y are not the same length, the shorter vector is zero-padded to the length of the longer vector.

filter (1-D digital filter) :

y = filter(b,a,X) filters the data in vector X with the filter described by numerator coefficient vector b and denominator coefficient vector a. If a(1) is not equal to 1, filter normalizes the filter coefficients by a(1). If a(1) equals 0, filter returns an error.

poly (Polynomial with specified roots) :

r = roots(p) which returns a column vector whose elements are the roots of the polynomial specified by the coefficients row vector p. For vectors, roots and poly are inverse functions of each other, up to ordering, scaling, and round off error.

tf(Convert unconstrained MPC controller to linear transfer function) :

sys=tf(MPCobj) The tf function computes the transfer function of the linear controller ss(MPCobj) as an LTI system in tf form corresponding to the MPC controller when the constraints are not active. The purpose is to use the linear equivalent control in Control System Toolbox for sensitivity and other linear analysis.

freqz (Frequency response of 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 (Absolute value and complex magnitude) :

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

fft (Discrete Fourier transform) :

Y = fft(X) Y = fft(X) returns the discrete Fourier transform (DFT) of vector X, computed with a fast Fourier transform (FFT) algorithm.

mod (Modulus after division) :

M = mod(X,Y) returns X - n.*Y where n = floor(X./Y). If Y is not an integer and the quotient X./Y is within round off error of an integer, then n is that integer. The inputs X and Y must be real arrays of the same size, or real scalars.

sqrt (Square root) :

B = sqrt(X) returns the square root of each element of the array X. For the elements of X that are negative or complex, sqrt(X) produces complex results.

ceil (Round toward infinity) :

B = ceil(A) rounds the elements of A to the nearest integers greater than or equal to A. For complex A, the imaginary and real parts are rounded independently.

fir1(Window-based finite impulse response filter design) :

b = fir1(n,Wn) returns row vector b containing the n+1 coefficients of an order n lowpass FIR filter. This is a Hamming-window based, linear-phase filter with normalized cutoff frequency Wn. The output filter coefficients, b, are ordered in descending powers of z.

buttord (Butterworth filter order and cutoff frequency) :

[n,Wn] = buttord(Wp,Ws,Rp,Rs) returns the lowest order, n, of the digital Butterworth filter that loses no more than Rp dB in the passband and has at least Rs dB of attenuation in the stopband. The scalar (or vector) of corresponding cutoff frequencies, Wn, is also returned. Use the output arguments n and Wn in butter.

fliplr (Flip matrix left to right) :

B = fliplr(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.If A is a row vector, then fliplr(A) returns a vector of the same length with the order of its elements reversed. If A is a column vector, then fliplr(A) simply returns A.

min (Smallest elements in array) :

C = min(A) returns the smallest elements along different dimensions of an array. If A is a vector, min(A) returns the smallest element in A.If A is a matrix, min(A) treats the columns of A as vectors, returning a row vector containing the minimum element from each column. If A is a multidimensional array, min operates along the first nonsingleton dimension.

max (Largest elements in array) :

C = max(A) returns the largest elements along different dimensions of an array.If A is a vector, max(A) returns the largest element in A.If A is a matrix, max(A) treats the columns of A as vectors, returning a row vector containing the maximum element from each column. If A is a multidimensional array, max(A) treats the values along the first non-singleton dimension as vectors, returning the maximum value of each vector.

find (Find indices and values of nonzero elements) :

ind = find(X) locates all nonzero elements of array X, and returns the linear indices of those elements in vector ind. If X is a row vector, then ind is a row vector; otherwise, ind is a column vector. If X contains no nonzero elements or is an empty array, then ind is an empty array.

residuez (z-transform partial-fraction expansion) :

residuez converts a discrete time system, expressed as the ratio of two polynomials, to partial fraction expansion, or residue, form. It also converts the partial fraction expansion back to the original polynomial coefficients.

[r,p,k] = residuez(b,a) finds the residues, poles, and direct terms of a partial fraction expansion of the ratio of two polynomials, b(z) and a(z). Vectors b and a specify the coefficients of the polynomials of the discrete-time system b(z)/a(z) in descending powers of z.

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.

log (Natural logarithm) :

Y = log(X) returns the natural logarithm of the elements of X. For complex or negative z, where z = x +y*i , the complex logarithm is returned.

EXPERIMENT NO: 02

Title: - To develop programs for generating elementary signal functions like unit sample, unit step, exponential, ramp sequences, sinusoidal, random and periodic signal.

Objective: To familiarise with MATLAB software, general functions and signal processing toolbox functions.

Task: generating elementary signal functions like unit sample, unit step, exponential, ramp sequences, sinusoidal, random and periodic signal. Practice on different operations on it.

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Reference: DSP by Prokis

Program: -

% program for generation of unit sample

```
clc;clear all;close all;

t = -3:1:3;

y = [zeros(1,3),ones(1,1),zeros(1,3)];

subplot(2,2,1);stem(t,y);

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

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

title('Unit Impulse Signal');
```

% program for genration of unit step of sequence [u(n)- u(n)-N]

```
t = -4:1:4;

y1 = ones(1,9);

subplot(2,2,2);stem(t,y1);

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

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

```
title('Unit step');
```

% program for generation of ramp signal

```
n1 = input('Enter the value for end of the seqeuence ');   %n1 = <any value>7 %
x = 0:n1;
subplot(2,2,3);stem(x,x);
ylabel('Amplitude------>');
xlabel('(c)n ------>');
title('Ramp sequence');
```

% program for generation of exponential signal

```
n2 = input('Enter the length of exponential seqeuence ');   %n2 = <any value>7 %
t = 0:n2;
a = input('Enter the Amplitude');    %a=1%
y2 = exp(a*t);
subplot(2,2,4);stem(t,y2);
ylabel('Amplitude------>');
xlabel('(d)n ------>');
title('Exponential sequence');
disp('Unit impulse signal');y
disp('Unit step signal');y1
disp('Unit Ramp signal');x
disp('Exponential signal');x
```

Output :

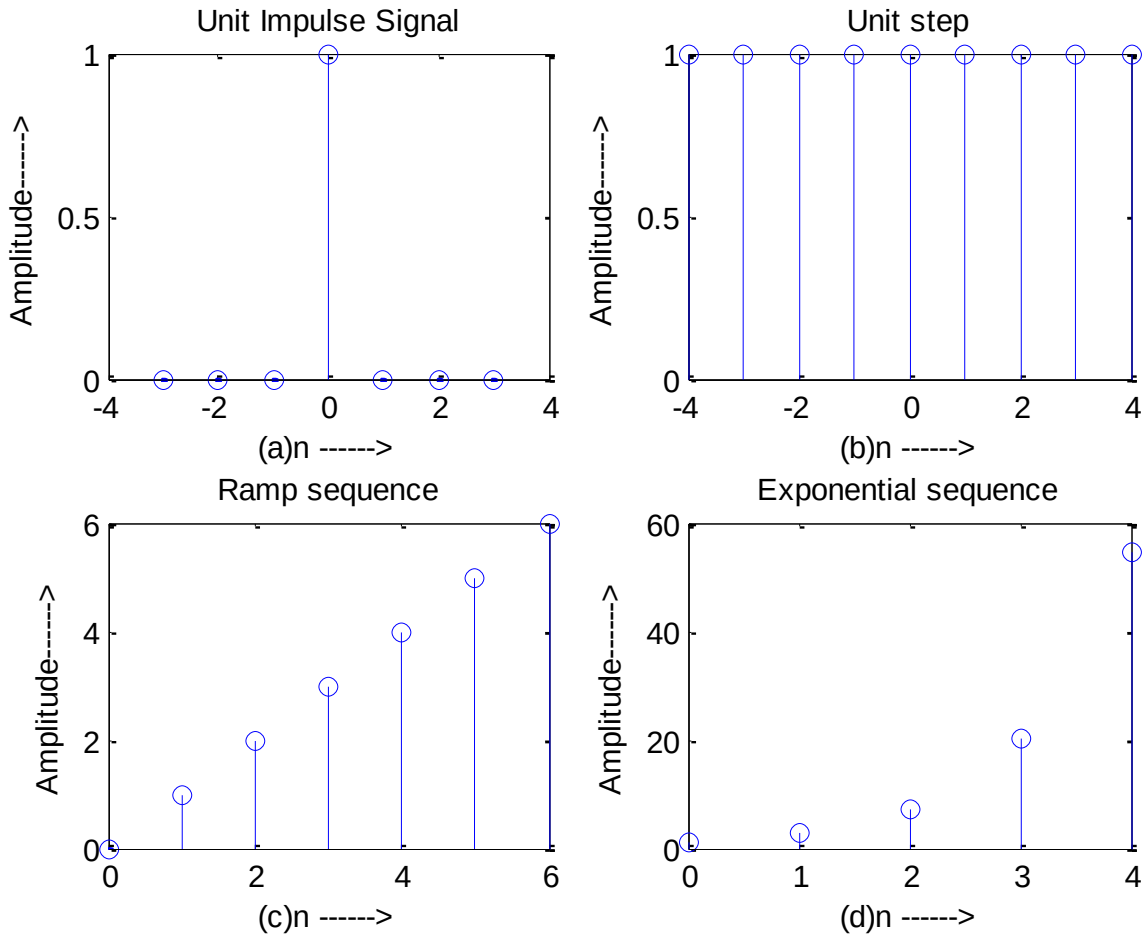
```
Enter the value for end of the seqeuence 6
Enter the length of exponential seqeuence 4
Enter the Amplitude1
```

Unit impulse signal y = 0 0 0 1 0 0 0

Unit step signal y1 = 1 1 1 1 1 1 1 1 1

Unit Ramp signal x = 0 1 2 3 4 5 6

Exponential signal x = 0 1 2 3 4 5 6



Unit Sample Sequence

```
>>function[x, n] =impseq (n0, n1, n2)
% Generates x[n] =delta (n-n0) ; n1<=n<=n2

% n1 is lower limit of required sequence;n2 is upper limit of required sequence
>>n=n1:n2;x=(n-n0)==0;
>>stem(n,x);
>>title(_Delayed Impulse_);
>>xlabel(_n_);
>>ylabel(_x[n]_);
```

Practice:

Use above function to plot unit sample sequence that has a value at n=-9 in a range from n=-14 to n=-2. Use zeros(1,N) command to generate above unit sample sequence function_

MATLAB CODE:

```
>> n0=-9;
>> n1=-14;
>> n2=-2;
>> n=n1:n2;
>> x=(n-n0)==0;
>> stem(n,x)
>> title('Delayed Impulse Sequence')
>> xlabel('n')
>> ylabel('x[n]')
```

Practice:

Use above function to plot unit step sequence having range between -5 and 15, shifted at n=-3. Use zeros(1,N) and ones(1,N) commands to generate above unit step sequence function…

MATLAB CODE:

```
>> n0=-3;

>> n1=-5;

>> n2=15;

>> n=n1:n2;

>> x=(n-n0)>=0;

>> stem(n,x)

>> title('Delayed Step Sequence')

>> xlabel('n')

>> ylabel('x[n]')
```

Sinusoidal Sequence:

$$X[n]=\cos(w_0 n+\theta), \text{ for all } n$$

MATLAB function sin or cos is used to generate sinusoidal sequences.

Practice:

Generate a sinusoidal signal given below in MATLAB and also plot the output…

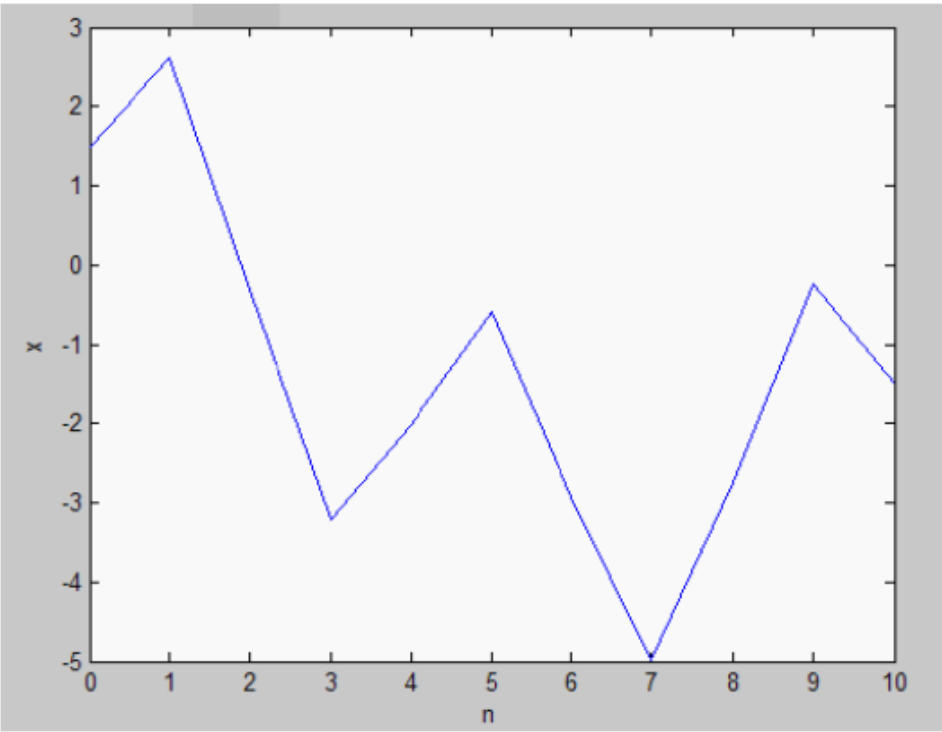
>>x[n]=3cos(0.1n∏+∏/3)+2sin(0.5n∏)

MATLAB CODE:

```
>> n=0:10;

>> a=3*cos((0.1*n*pi)+pi/3);

>> b=2*sin(0.5*n*pi);
```

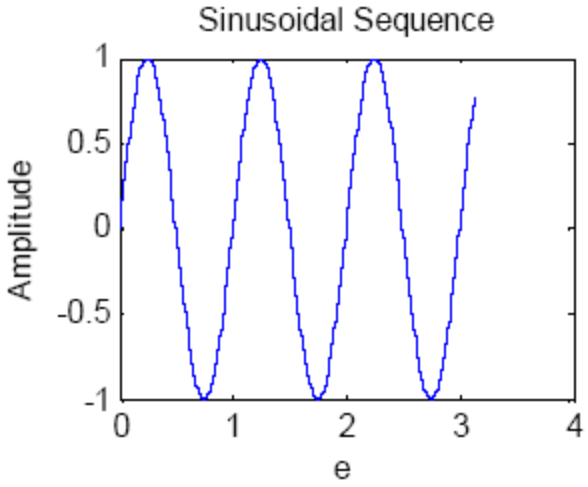
```
>> x=a+b;

>> plot(n,x)
```

PLOT:



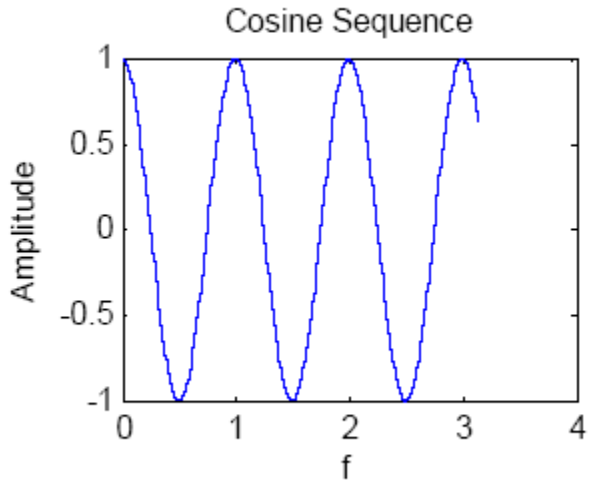
%Sinusoidal sequence:-

```
t=0:0.01:pi;
y=sin(2*pi*t);
subplot(2,2,1);
plot(t,y);
ylabel('Amplitude');
xlabel('e');
title('Sinusoidal Sequence');
```



% Cosine Sequence:-

```
t=0:0.01:pi;
y=cos(2*pi*t);
subplot(2,2,1);
plot(t,y);
ylabel('Amplitude');
xlabel('f');
title('Cosine Sequence');
```



Random Signals:

Many practical sequences cannot be described by mathematical expressions like those above. These signals are called random or stochastic signals and are characterized by parameters of associated probability density functions or their statistical moments. In MATLAB rand (1, N) generates a length N random sequence whose elements are uniformly distributed between [0, 1].randn (1, N) generates a length N Guassian random sequence with mean 0 and variance 1. Above mentioned functions can be transformed to generate other random sequences.

EXAMPLE:

```
>> rand(1,5)
```

RESULT:

```
ans =

0.0975    0.2785    0.5469    0.9575    0.9649
```

Periodic Signals:

Nncopy can be used to produce a vector with repeating sequence. If we want a vector 'x' in below example to repeat for three times, we can use,

```
>>x=[1 2 3];

>>y=nncopy(x,1,3);
```

1 2 3 1 2 3 1 2 3

Practice:

Write MATLAB code for generating a periodic sequence using only 'ones' command.

CODE:

```
>> x=ones(1,2);

>> y=nncopy(x,3,3)
```

RESULT:

```
y =     1     1     1     1     1     1

        1     1     1     1     1     1

        1     1     1     1     1     1
```

EXPERIMENT NO: 03

Title: Generation of basic signals and illustration of sampling process using Matlab

Objective: To familiarise with MATLAB software, general functions and signal processing toolbox functions.

Task: Learn how to generate sinusoidal signal and how to sample it in to discrete time sequences.

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Reference: DSP by Prokis

ALGORITHM:-

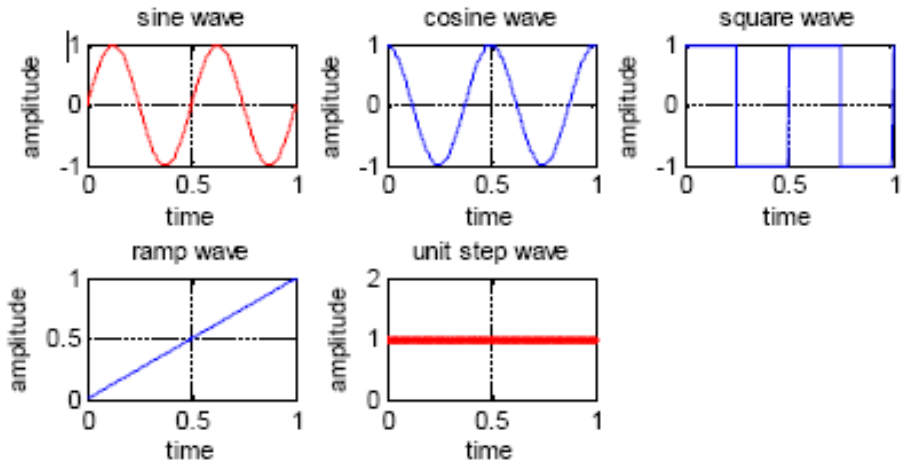
- Get the amplitude and frequency of the signal
- Use ‘sin’, ’cos’ ,’square’ matlab built in functions
- Using ‘plot’ function plot the signal

MATLAB CODE:-

```
clc;
clear all;
close all;
t=0:.001:1;
f=input('Enter the value of frequency');
a=input('Enter the value of amplitude');
subplot(3,3,1);
y=a*sin(2*pi*f*t);
plot(t,y,'r');
xlabel('time');
ylabel('amplitude');
title('sine wave')
grid on;
subplot(3,3,2);
z=a*cos(2*pi*f*t);
plot(t,z);
xlabel('time');
ylabel('amplitude');
title('cosine wave')
grid on;
subplot(3,3,3);
s=a*square(2*pi*f*t);
plot(t,s);
xlabel('time');
ylabel('amplitude');
```

```
title('square wave')
grid on;
subplot(3,3,4);
plot(t,t);
xlabel('time');
ylabel('amplitude');
title('ramp wave')
grid on;
subplot(3,3,5);
plot(t,a,'r');
xlabel('time');
ylabel('amplitude');
title('unit step wave')
grid on;
```

FIGURE:-



SAMPLE INPUT:-
Enter the value of frequency2
Enter the value of amplitude1

RESULTS:- Thus the generation of continues time signals using matlab was verified

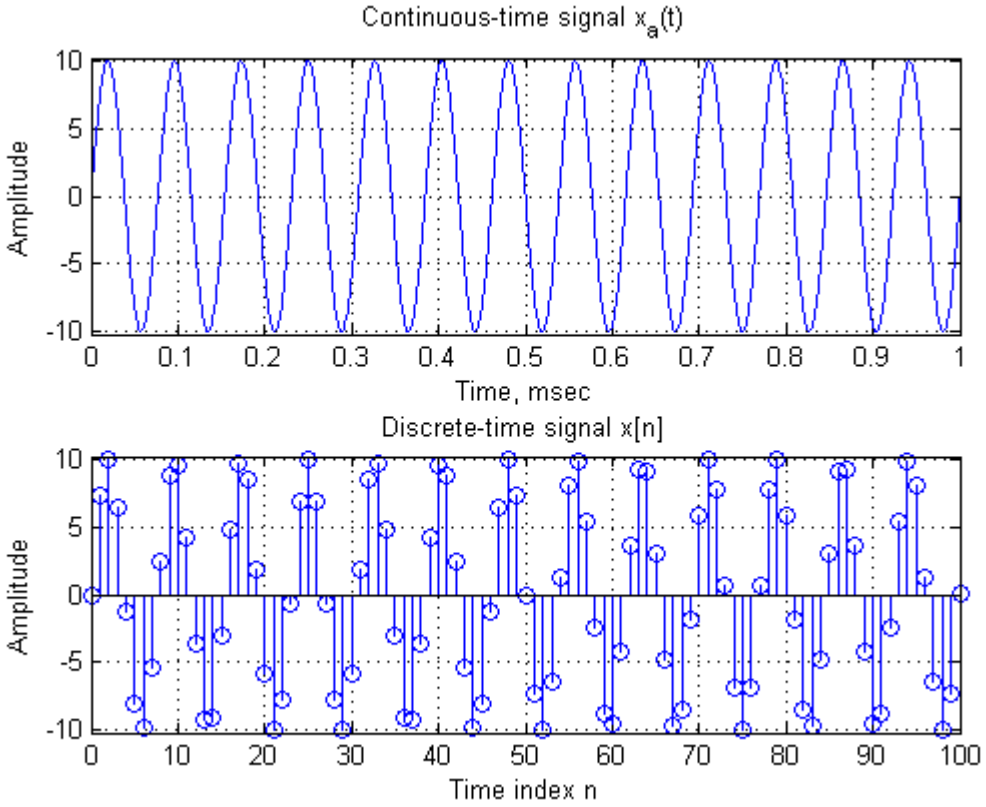
PROGRAM :

```
% Generation of Sine Wave & Illustration of the Sampling Process in the Time
Domain
clc;
```

```
t = 0:0.0005:1;
a = 10
f = 13;
xa = a*sin(2*pi*f*t);
subplot(2,1,1)
plot(t,xa);grid
xlabel('Time, msec');
ylabel('Amplitude');
title('Continuous-time signal x_{a}(t)');
axis([0 1 -10.2 10.2])
subplot(2,1,2);
T = 0.01;
n = 0:T:1;
xs = a*sin(2*pi*f*n);
k = 0:length(n)-1;
stem(k,xs);
grid
xlabel('Time index n');
ylabel('Amplitude');
title('Discrete-time signal x[n]');
axis([0 (length(n)-1) -10.2 10.2])
```

a = 10

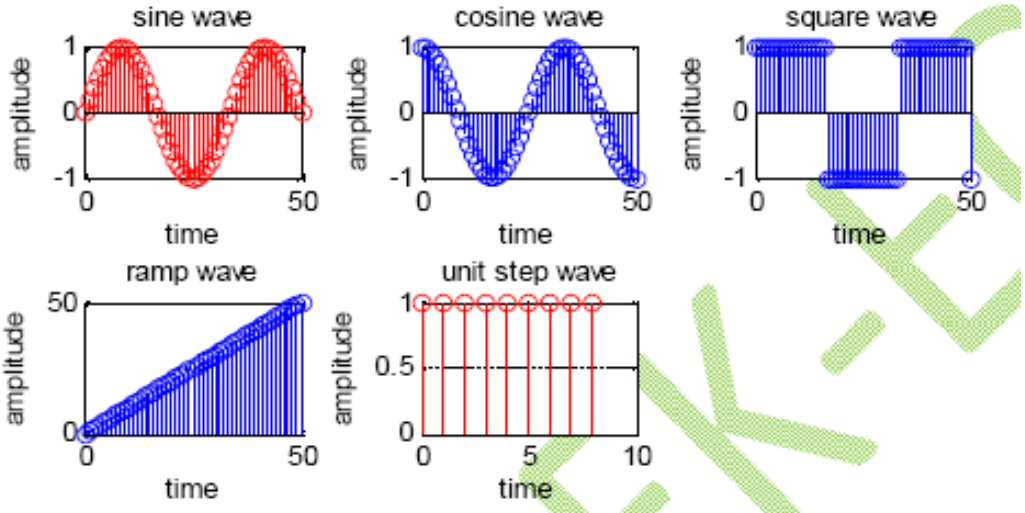
EXPECTED GRAPH:



%%% Generation of discrete signal

```
clc;
clear all;
close all;
n=0:1:50;
f=input('Enter the value of frequency');
a=input('Enter the value of amplitude');
N=input('Enter the length of unit step');
subplot(3,3,1);
y=a*sin(2*pi*f*n);
stem(n,y,'r');
xlabel('time');
ylabel('amplitude');
title('sine wave')
grid on;
subplot(3,3,2);
z=a*cos(2*pi*f*n);
stem(n,z);
xlabel('time');
ylabel('amplitude');
title('cosine wave')
grid on;
subplot(3,3,3);
s=a*square(2*pi*f*n);
stem(n,s);
xlabel('time');
ylabel('amplitude');
title('square wave')
grid on;
subplot(3,3,4);
stem(n,n);
xlabel('time');
ylabel('amplitude');
title('ramp wave')
grid on;
x=0:N-1;
d=ones(1,N);
subplot(3,3,5);
stem(x,d,'r');
xlabel('time');
ylabel('amplitude');
title('unit step wave')
grid on;
```

FIGURE:-



Viva Questions:

1. Define sinusoidal signal
2. Define C.T.S
3. Define D.T.S.
4. Compare C.T.S & D.T.S
5. Define

Stem, Plot, Plot3,fplot, ezplot, linspace, flyplr, grid,mesh and legend

6. Draw the C.T.S & D.T.S diagrams

EXERCISE QUESTIONS:

1. Write program to get Discrete time Sinusoidal Signal
2. Write program to get Fourier Transform of Sinusoidal Signal
3. Write program to get Inverse Fourier Transform of Sinusoidal Signal
4. Write Program for the following Function

$$Y=\exp(-2*\prod*f*t)+\exp(-8*\prod*f*)$$

$$Y=((\exp(-1.56\prod f)*\mathrm{Sin}(2\prod f)+\cos(2\prod f)$$

EXPERIMENT NO: 04

Title: - Introduction to different operations on sequences

Objectives: To familiarise with MATLAB software, general functions and signal processing toolbox functions and demonstrate different operations.

Task: Learn how to generate sinusoidal signal and how to sample it in to discrete time sequences.

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Operations on sequences:

Signal addition:

In MATLAB, two sequences are added sample by sample using arithmetic operator "+". However if lengths of sequences are different or if sample positions are different for equal-length sequences, then we can not directly use "+" operator. In this case, we have to augment the sequences so that they have same position vector _n_(and hence the same length). Following is the code for sequence addition keeping in view above mentioned facts.

```
Function[y,n]=sigadd(x1,n1,x2,n2)
%implemnts y[n]=x1[n]+x2[n]
%y=sum sequence over n which includes n1 and n2
%x1=first sequence over n1
%x2=second sequence over n2(n2 can be different from n1)
n=min(min(n1),min(n2)):max(max(n1),max(n2));
y1=zeroes(1,length(n));y2=y1;
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
y=y1+y2;
```

Practice:
x1[n]=[3,11,7,0,-1,4,2] and x2[n]=[2,3,0,-5,2,11]
Use above function to add two signals.

MATLAB CODE:

```
x1=[3 11 7 0 -1 4 2];
x2=[2 3 0 -5 2 11];
n1=4;%initial element number of x1
n2=3;%initial element number of x2
if n2<n1
```

```
x2=[zeros(1,length(n1-n2)) x2];
x1=x1;
if length(x1)>length(x2)
x2=[x2 zeros(1,length(x1)-length(x2))]
x1=x1
x=x1+x2
elseif length(x1)<length(x2)
x1=[x1 zeros(1,length(x2)-length(x1))]
x2=x2
x=x1+x2;
elseif length(x1)==length(x2)
x=x1+x2;
end
elseif n2>n1
x1=[zeros(1,length(n2-n1)) x1];
x2=x2;
if length(x1)>length(x2)
x2=[x2 zeros(1,length(x1)-length(x2))]
x1=x1
x=x1+x2
elseif length(x1)<length(x2)
x1=[x1 zeros(1,length(x2)-length(x1))]
x2=x2
x=x1+x2;
elseif length(x1)==length(x2)
x=x1+x2;
end
elseif n2==n1
if length(x1)>length(x2)
x2=[x2 zeros(1,length(x1)-length(x2))]
x1=x1
x=x1+x2
elseif length(x1)<length(x2)
x1=[x1 zeros(1,length(x2)-length(x1))]
x2=x2
x=x1+x2;
elseif length(x1)==length(x2)
x=x1+x2;
end
end
```

MATLAB RESULTS:

```
x2 = 2 3 0 -5 2 11 0 0
x1 = 0 3 11 7 0 -1 4 2
x = 2 6 11 2 2 10 4 2
```

Signal multiplication:

In MATLAB, two signals are multiplied sample by sample using array operator "*". Similar instructions regarding position vector imply for multiplication as for addition.

Practice:
Write a MATLAB function sigmult to multiply two signals x1[n] and x2[n], where x1[n] and x2[n] may have different durations. Call this function to multiply any two signals.

MATLAB RESULTS:
x2 = 2 3 0 -5 2 11 0 0
x1 = 0 3 11 7 0 -1 4 2
x = 0 9 0 -35 0 -11 0 0

Signal shifting:
In this operation each sample of x[n] is shifted by an amount k to obtain a shifted sequence y[n].
y[n]=x[n-k]
This operation has no effect on vector x but vector n is changed by adding k to each element.

```
Function[y,n]=sigshift(x,m,n0)
%implements y[n]=x[n-n0]
n=m+n0;
y=x;
```

Signal Folding:

In this operation each sample of x[n] is flipped around n=0 to obtain a folded sequence y[n].
y[n]=x[-n]

In MATLAB, this function is implemented by flipr(x) function for sample values and by _flipr(n) function for sample positions.

```
Function[y,n]=sigfold(x,n)
%implements y[n]=x[-n]
Y=flipr(x);
n=-flipr(n)
```

Sample Summation and Sample Product:

It adds all sample values of x[n] between n1 and n2. It is implemented by

sum(x(n1:n2)).
Sample multiplication multiplies all sample values of x[n] between n1 and n2. It is implemented by prod(x(n1:n2)).

MATLAB CODE FOR SUMMATION:

```
x=[1 2 3 4 5 6];
n1=x(1,3);
n2=x(1,5);
sum(x(n1:n2))
```

RESULT:

```
ans =12
```

MATLAB CODE FOR PRODUCT:

```
x=[1 2 3 4 5 6];
n1=x(1,3);
n2=x(1,5);
prod(x(n1:n2))
```

RESULT:

```
ans =60
```

Signal Energy:

Energy of a finite duration sequence x[n] can be computed in MATLAB using

```
>>Ex=sum(x.*conj(x));
```

or

```
>>Ex=sum(abs(x).^2);
```

MATLAB CODE:

```
x=[1 2 3];
Ex=sum(abs(x).^2)
```

RESULT:

```
Ex =14
```

EXPERIMENT NO: 05

Title: - Understanding of aliasing effect of discrete time signals in MATLAB

Outline: Aliasing is the phenomenon that results in a loss of information when a signal is reconstructed from its sampled signal. In Principle, the analogue can be reconstructed from the samples, provided that the sampling rate is sufficiently high to avoid the problem called Aliasing.

Task: Illustrate aliasing effect..

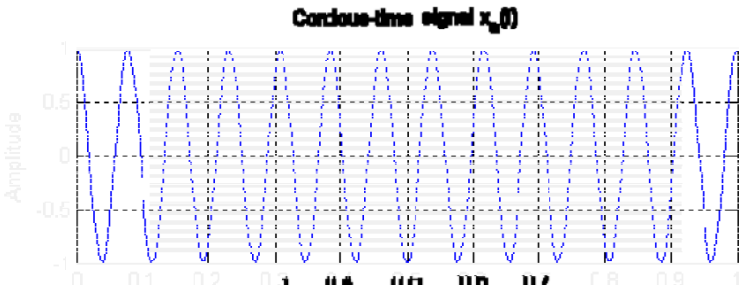
Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

MATLAB CODE:

```
%Illustration of Aliasing Effect in the Time-Domain
cls;
t=0:0.0005:1;
f=13;
xa=cos(2*pi*f*t);
subplot(2,1,1)
plot(t,xa);
grid
Xlabel('Time,msec');
Ylabel('Amplitude');
title('Continous=time signal x_{a}(t)');
axis([0 1 -1.2 1.2])
  subplot(2,1,2);

  T=0.1;

  F=13;

  n=0:T:1;

  xs=cos(2*pi*f*n);

  t=linspace(-0.5,5,500);

  ya=sinc((1/T)*t(:,ones(size(n)))-(1/T)*n(:,ones(size(t))))*xs;

  plot(n,xs,'0',t,ya);

  grid;
```

```
xlabel('Time,msec');

ylabel('Amplitude');

title('Reconstructed continuous-time signal y_{a}(t)');

axis([0 1 -
1.2 1.2]);
```

RESULT:



EXPERIMENT NO: 06

Title: To develop the program for finding the convolution between two sequences.

Outline: To familiarise with MATLAB software, general functions and signal processing toolbox functions and demonstrate different operations.

Task: illustration of convolution between two sequences..

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

ALGORITHM:-

- Read the input sequence x[n] ,and plot
- Read the impulse sequence h[n] , and plot
- Use the matlab function ‘conv’
- Convolve the two sequence and plot the result

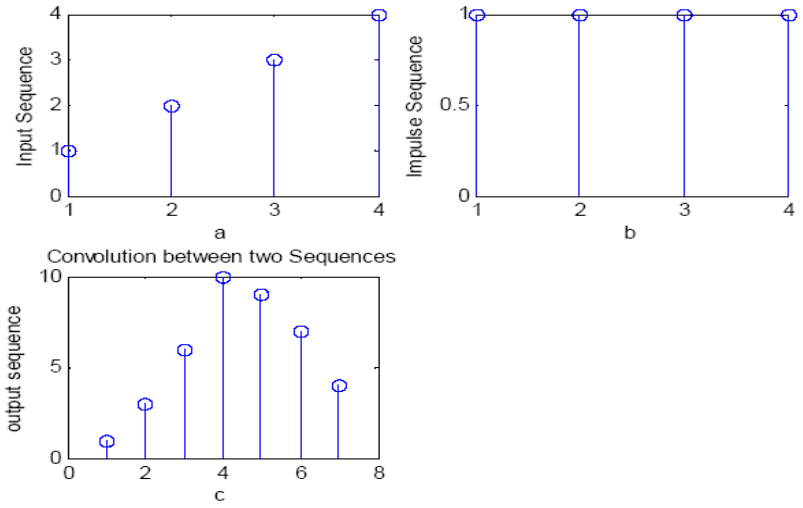
Program:-

```
x=input('enter the first sequence')
enter the first sequence[1 2 3 4]
h=input('enter the second sequence')
enter the second sequence[1 1 1 1]
y=conv(x,h);
subplot(2,2,1);
stem(x);
xlabel('a');
ylabel('Input Sequence');
subplot(2,2,2);
stem(h);
xlabel('b');
ylabel('Impulse Sequence');
subplot(2,2,3);
stem(y);
xlabel('c');
ylabel('output sequence');
title('Convolution between two Sequences');
```

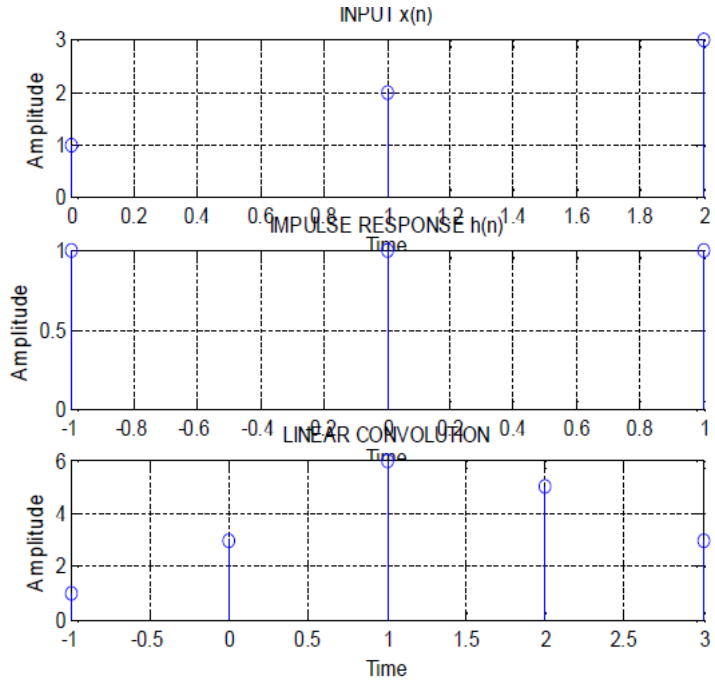
Program 2:

```
clc;
close all;
clear all;
x=[1,2,1,1]; %first signal 0r input signal
h=[1,-1,1,-1]; %second signal
```

```
N1=length(x);
N2=length(h);
X=[x,zeros(1,N2)]; %padding of N2 zeros
H=[h,zeros(1,N1)]; %padding of N1 zeros
   for i=1:N1+N2-1
      y(i)=0;
      for j=1:N1
         if(i-j+1>0)
            y(i)=y(i)+X(j)*H(i-j+1);
         else
         end
      end
   end
   stem(y);
   ylabel('y[n]');
   xlabel('------>n');
   title('convolution of two signal');
```



Practice:



SAMPLE INPUT:--

```
Enter the starting point of x(n)=0
Enter the starting point of h(n)=-1
Enter the co-efficient of x(n)=[1 2 3]
Enter the co-efficient of h(n)=[1 1 1]
     1     3     6     5     3
```

Use what is in the noisyC script to generate a noisy sine wave:

```
fs = 1e4;
t = 0:1/fs:5;
sw = sin(2*pi*262.62*t); % Middle C
n = 0.1*randn(size(sw));
swn = sw + n;
```

Use a simple lowpass (averaging) Filter:

```
b=[.25 .25 .25 .25];
a=[1 0 0 0];
y=filter(b,a,swn);
```

```
figure, plot(t,y), axis([0 0.04 -1.1 1.1])
h=impz(b,a);
y2=conv(swn,h);
figure, plot(t,y2(1:end-3)), axis([0 0.04 -1.1 1.1])
```

Q. How do the two outputs (y and y2) compare?

EXPERIMENT NO: 07

Title: To develop the program for finding the Correlation of two sequences.

Outline: To familiarise with MATLAB software, general functions and signal processing toolbox functions and demonstrate different operations.

Task: illustration of correlation between two sequences..

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Program :

```
% correlation of two sequences
% x(n)=[3,11,7,0,-1,4,2], x=[-3:3]
% y(n)= x(n-2)+w(n), where w(n) is a random number
clc;
x=[3, 11, 7, 0, -1, 4, 2]; nx=[-3:3]; % given x(n)
% signal shift
ny=nx+2; y=x; % implement y(n)=x(n-2)
%%%%%%%%%%%%%%
w=rand(1,length(y)); nw=ny; % generate white noise w(n)
% signal addition
n=min(min(nx),min(nw)):max(max(nx),max(nw))% duration of y(n)
y1=zeros(1,length(n)); y2=y1; % initialization
y1(find((n>=min(nx))&(n<=max(nx))==1))=y; % y with duration of y
y2(find((n>=min(nw))&(n<=max(nw))==1))=w; % w with duration of y
ww=y1+y2; % sequence addition
%%%%%%%%%%%%%%
% signal folding
x=fliplr(x); nx=-fliplr(nx);
% convolution of arbitrary support sequences
```

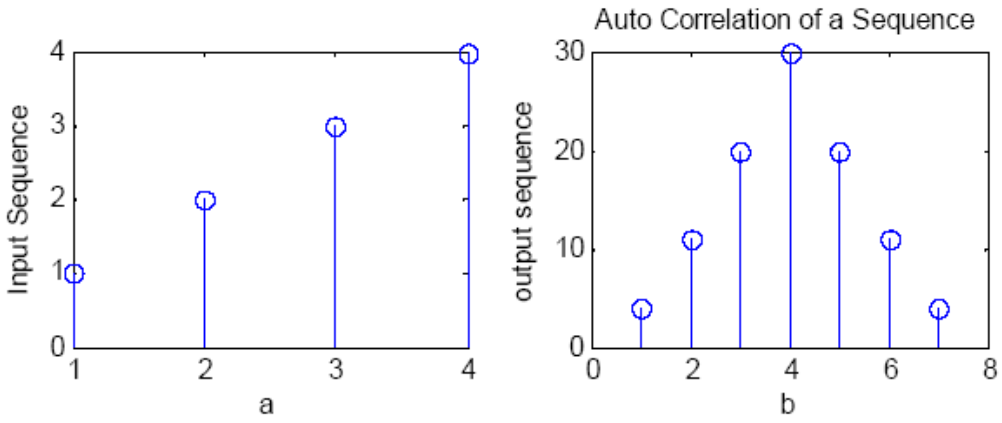
```
nyb=nx(1)+n(1);
nye=nx(length(x))+n(length(ww));
ny=[nyb:nye];
%%%%%%%%%%%%%%%%
% convolution
N1=length(x);
N2=length(ww);
X=[x,zeros(1,N2)];% padding of N2 zeros
H=[ww,zeros(1,N1)];% padding of N1 zeros
   for i=1:N1+N2-1
      y(i)=0;
     for j=1:N1
        if(i-j+1>0)
      y(i)=y(i)+X(j)*H(i-j+1);
        else
        end
     end
   end
%%%%%%%%%%%%%%%%
%subplot(1,1,1); subplot(2,1,1);
stem(ny,y);
%axis([-4,8,-50,250]); xlebel('lag variable 1')
%ylebel('ny');
%title('cross correlation with noise sequence')
```

Problem: Autocorrelation of a sequence

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

enter the first sequence[1 2 3 4]

```
y=xcorr(x,x);
subplot(2,2,1);
stem(x);
xlabel('a');
ylabel('Input Sequence');
subplot(2,2,2);
stem(y);
xlabel('b');
ylabel('output sequence');
title('Auto Correlation of a Sequence ');
```



EXPERIMENT NO: 08

Title:- To develop the program for finding the DFT

Outline: To familiarise with MATLAB software, general functions and signal processing toolbox functions and demonstrate different operations.

Task: illustration of Discrete Fourier Transform.

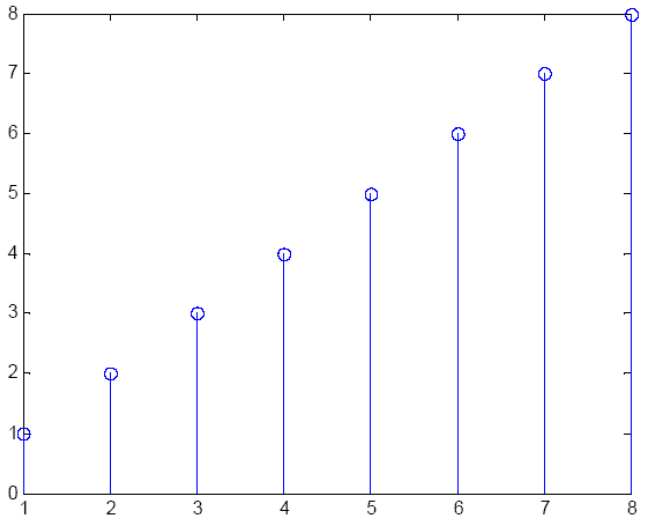
Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

ALGORITHM:-

- Enter the input sequence x[n]
- Enter the length of sequence,N
- Use the matlab function ‘fft’
- Plot the input and output sequence

Program:-

```
x=input('enter the input sequence');
enter the input sequence[1 2 3 4 5 6 7 8]
n=input('enter the length of sequence');
enter the length of sequence8
X=fft(x,n);
stem(x);
plot(x);
plot(X);
title('DFT of the Input Sequence');
```

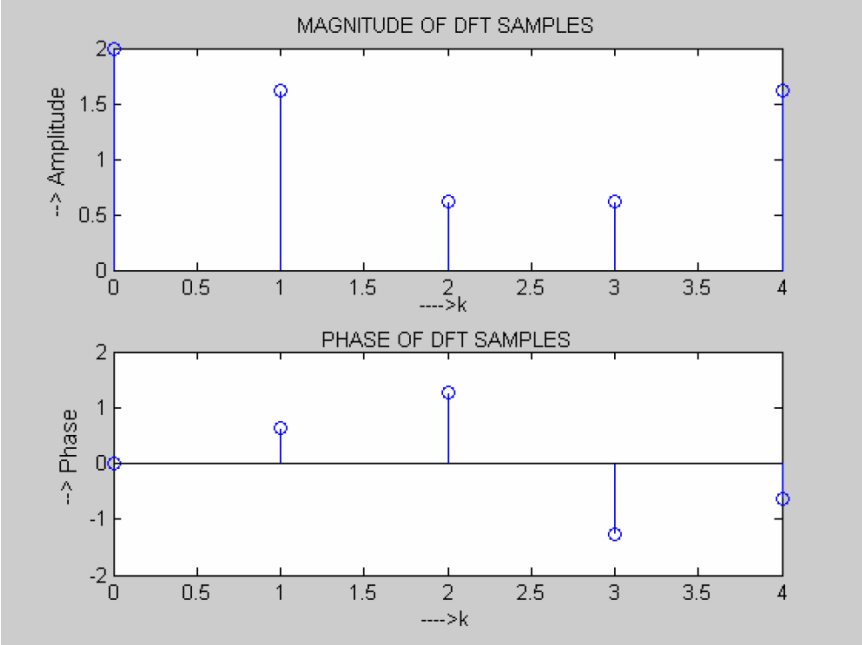


Program:

```
% N-point FFT algorithm
N=input ('enter N value=');
Xn=input ('type input sequence=');
k=0:1: N-1;
L=length (xn)
if (N<L)
   error ('N MUST BE>=L');
end;
x1= [xn zeros (1, N-L)]
for c=0:1:N-1;
   for n=0:1:N-1;
      p=exp (-i*2*pi*n*C/N);
      x2(c+1, n+1) =p;

end;
MagXk=abs (Xk);
angXk=angle (Xk);
subplot (2, 1, 1);
stem (k, magXk);
title ('MAGNITUDE OF DFT SAMPLES');
xlabel ('---->k');
ylabel ('-->Amplitude');
subplot (2, 1, 2);
stem (k, angXk);
title ('PHASE OF DFT SAMPLES');
xlabel ('---->k');
ylabel ('-->Phase');
disp (‘abs (Xk) =');
disp (magXk)
disp ('angle=');
disp (angXk)
```

Output:

```
enter N value=5
type input sequence= [1 1 0]
L= 3
x1 = 1     1     0     0     0
abs (Xk) = 2.0000    1.6180    0.6180    0.6180    1.6180
angle= 0    0.6283    1.2566   -1.2566   -0.6283
```



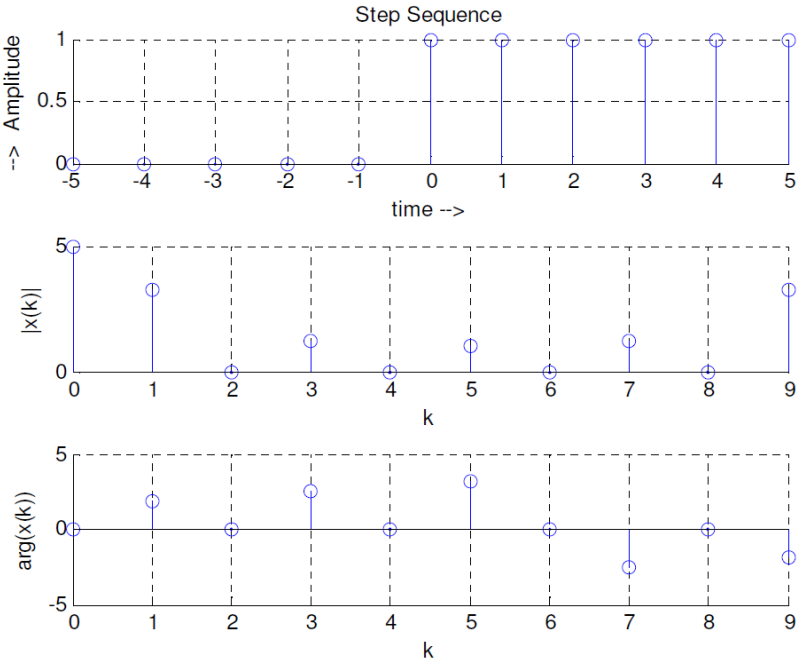
%To compute the FFT of the step sequence and plot magnitude and phase response

```
clc;
clear all;
close all;
%Step Sequence
s=input ('enter the length of step sequence');
t=-s:1:s;
y=[zeros(1,s) ones(1,1) ones(1,s)];
subplot(3,1,1);
stem(t,y);
grid
input('y=');
disp(y);
```

```
title ('Step Sequence');
xlabel ('time -->');
ylabel ('--> Amplitude');
xn=y;
N=input('enter the length of the FFT sequence: ');
xk=fft(xn,N);
magxk=abs(xk);
angxk=angle(xk);
k=0:N-1;
subplot(3,1,2);
stem(k,magxk);
grid
xlabel('k');

ylabel('|x(k)|');
subplot(3,1,3);
stem(k,angxk);
disp(xk);
grid
xlabel('k');
ylabel('arg(x(k))');
```

outputs:

```
enter the length of step sequence: 5
y=   0    0    0    0    0    1    1    1    1    1    1
enter the length of the FFT sequence:  10

  5.0000          -1.0000 + 3.0777i       0          -1.0000 + 0.7265i       0          -1.0000                 0
  -1.0000 - 0.7265i       0          -1.0000 - 3.0777i
```


Step Sequence
--> Amplitude
time -->
|x(k)|
k
arg(x(k))
k

Practice:

A common use of Fourier transforms is to find the frequency components of a signal buried in a noisy time domain signal. In this exercise we will create a sound signal sampled at 1000 Hz containing 50 Hz and 120 Hz and corrupt it with some zero-mean random noise.

• Use SIN function to create a 5 second sound signal with 50Hz and 120 Hz

t = 0:0.001:5; number of sampling points at 1000Hz for 5 seconds
x = sin(2*pi*50*t)+sin(2*pi*120*t); Create a sound signal with 50Hz and 120Hz

• Use RANDN to create random noise to be added to the sound signal.
y = x + 2*randn(size(t));

• Use PLOT command to plot the corrupted sound signal
plot(1000*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')

• It is difficult to identify the frequency components by looking at the original signal. Using discrete Fourier transform to convert to the signal to frequency domain. We will perform a 512-point fast Fourier transform (FFT), calculate and plot the power spectrum.

1. Calculate FFT
Y = fft(y,512);

2. Calculate power spectrum (a measurement of the power at various frequencies)

Power spectrum = Y.* conj(Y) / 512;

3. Graph the first 257 points (the other 255 points are redundant) on a meaningful frequency axis:

f = 1000*(0:256)/512;
plot(f,Pyy(1:257))
title('Frequency content of y')
xlabel('frequency (Hz)')

EXPERIMENT NO: 09

Title:- To develop the program for finding the magnitude and phase response of system described by system function H(s).

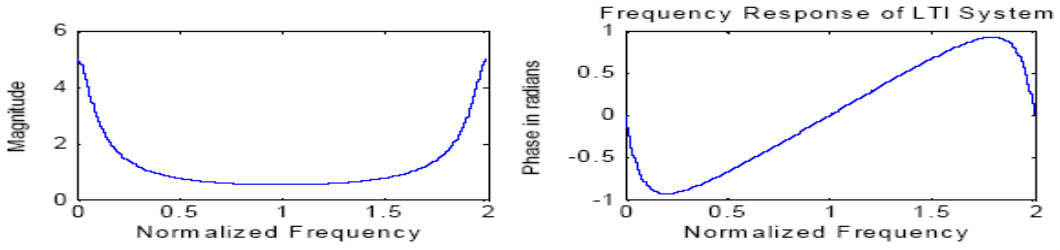
Outline: To familiarise with MATLAB software, general functions and signal processing toolbox functions and demonstrate different operations.

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Program:-

```
b=[1];
a=[1,-0.8];
h=freqz(b,a,w);
w=0:0.01:2*pi;
[h]=freqz(b,a,w);
subplot(2,2,1);
plot(w/pi,abs(h));
xlabel('Normalized Frequency');
ylabel('Magnitude');
subplot(2,2,2);
plot(w/pi,angle(h));
xlabel('Normalized Frequency');
ylabel('Phase in radians');
title('Frequency Response of LTI System');
```



EXPERIMENT NO: 10

Title:- To find the frequency response of analog LP/HP filter

Outline: To familiarise with MATLAB software, general functions and signal processing toolbox functions and demonstrate different operations.

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

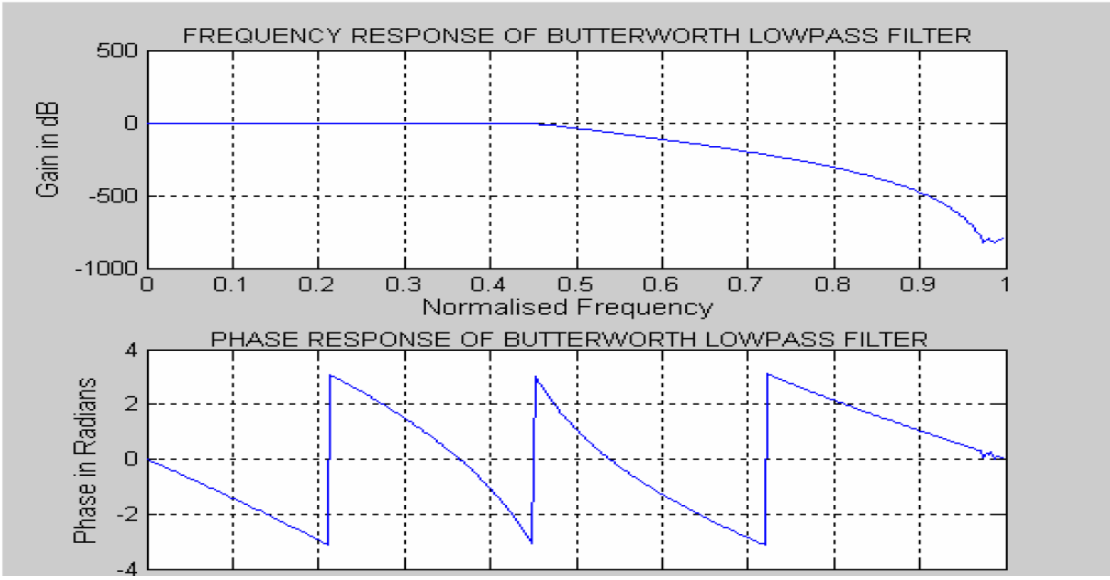
Program:

```
%Design of Butterworth lowpass filter
alphap=input ('enter the pass band attenuation=');
alphas=input ('enter the stop band attenuation=');
fp=input ('enter the passband frequency=');
fs=input ('enter the stop band frequency=');
F=input ('enter the sampling frequency=');
omp=2*fp/F;
oms=2*fs/F;
[N, wn]=buttord (omp, oms, alphap, alphas,'s')
[b, a]=butter (n, wn)
w=0:0.01: pi;
[h, ph]=freqz (b, a, w);
m=20*log (abs (h));
an=angle (h);
subplot (2, 1, 1);
plot (ph/pi, m);
grid on;
ylabel ('Gain in dB');
xlabel ('Normalised Frquency');
title ('FREQUENCY RESPONSE OF BUTTERWORTH LOWPASS FILTER');
```

```
subplot (2, 1, 2);
plot (ph/pi, an);
grid on;
ylabel ('Phase in Radians');
xlabel ('Normalisd Frequency');
title ('PHASE RESPONSE OF BUTTERWORTH LOWPASS FILTER');
```

Output:

```
Enter the pass band attenuation=0.4
 Enter the stop band attenuation=60
 Enter the pass band frequency=400
 Enter the stop band frequency=800
 Enter the sampling frequency=2000
n = 12
Wn = 0.4499
b = 0.0003    0.0040    0.0220    0.0735    0.1653    0.2644    0.3085
0.2644
0.1653    0.0735    0.0220    0.0040    0.0003
a = 1.0000   -1.1997    2.2442   -1.7386    1.5475   -0.7945    0.4106
-0.1362
0.0412   -0.0080    0.0013   -0.0001    0.0000
```

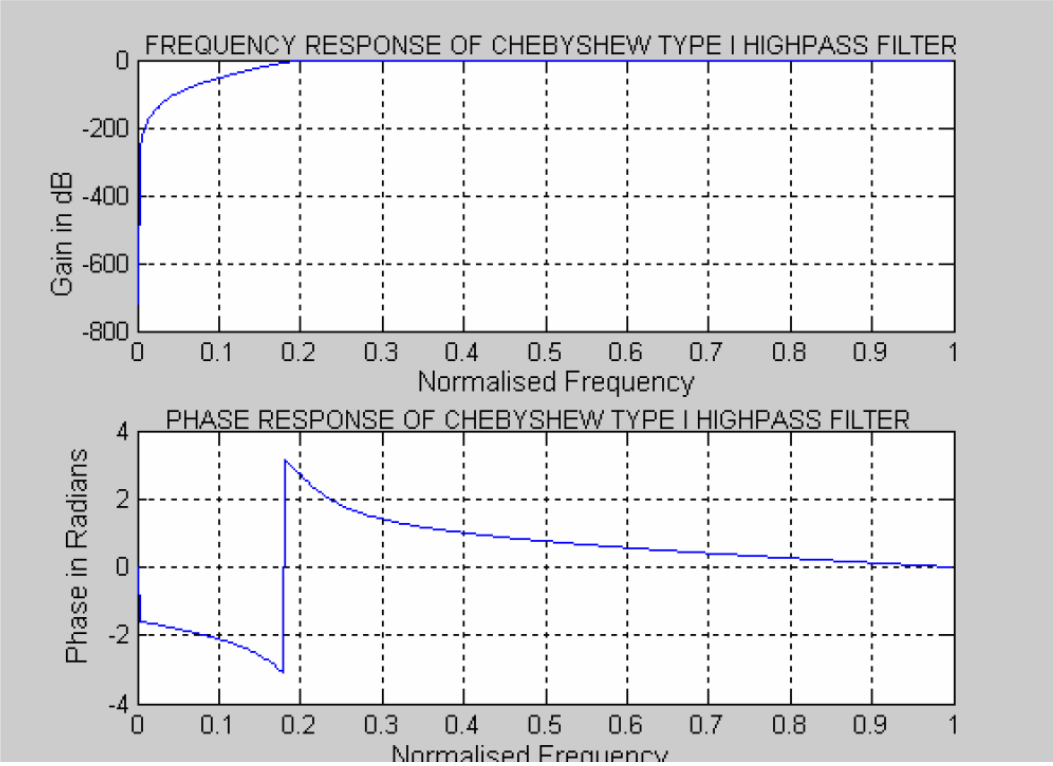


Program:

```
%Design of IIR Chebyshew type I high pass filter
alphap=input ('enter the pass band attenuation=');
alphas=input ('enter the stop band attenuation=');
wp=input ('enter the pass band frequency=');
ws=input ('enter the stop band frequency=');
[n, wn]=cheb1ord (wp/pi, ws/pi, alphap, alphas,'s')
[b, a]=cheby1 (n, alphap, wn,'high')
w=0:0.01: pi;
[h, ph]=freqz (b, a, w);
m=20*log (abs (h));
an=angle (h);

subplot (2, 1, 1);
plot (ph/pi, m);
grid on;
ylabel ('Gain in dB');
xlabel ('Normalised Frquency');
title ('FREQUENCY RESPONSE OF CHEBYSHEW TYPE I HIGHPASS
      FILTER');
subplot (2, 1, 2);
plot (ph/pi, an);
grid on;
ylabel ('Phase in Radians');
xlabel ('Normalisd Frequency');
title ('PHASE RESPONSE OF CHEBYSHEW TYPE I HIGHPASS FILTER');
```

Output:

```
Enter the pass band attenuation=1
Enter the stop band attenuation=15
Enter the pass band frequency=0.2*pi
Enter the stop band frequency=0.1*pi
n =3
wn = 0.2000
b =0.4759   -1.4278    1.4278   -0.4759
a = 1.0000   -1.6168    1.0366   -0.1540
```

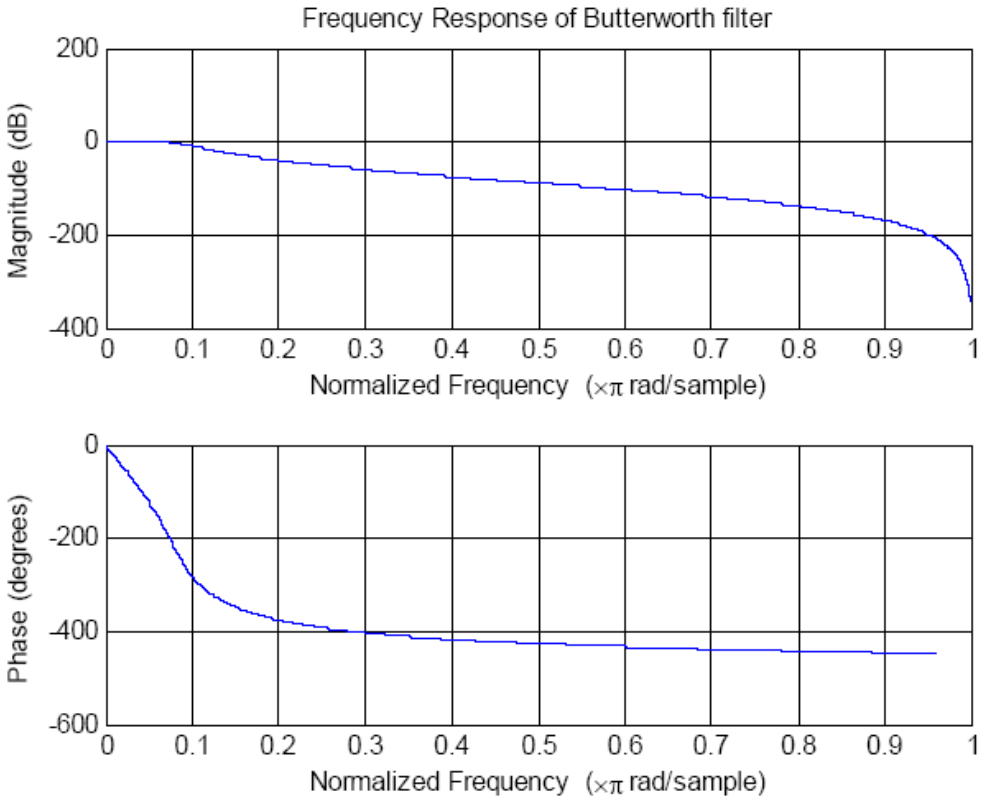


Practice:

To develop the program for designing Low Pass Butterworth filter having passband defined from 0-40 Hz and stopband in the range of 150-500Hz having less than 3 dB of ripple in the passband and atleast 60dB of attenuation in the stopband.

Program:-

```
wp=40/500;
ws=150/500;
[n,wn]=buttord(wp,ws,3,60);
n =
5
wn =
0.0810
(b,a)=butter(n,wn);
freqz(b,a)
title('Frequency Response of Butterworth filter')
```



EXPERIMENT NO: 11

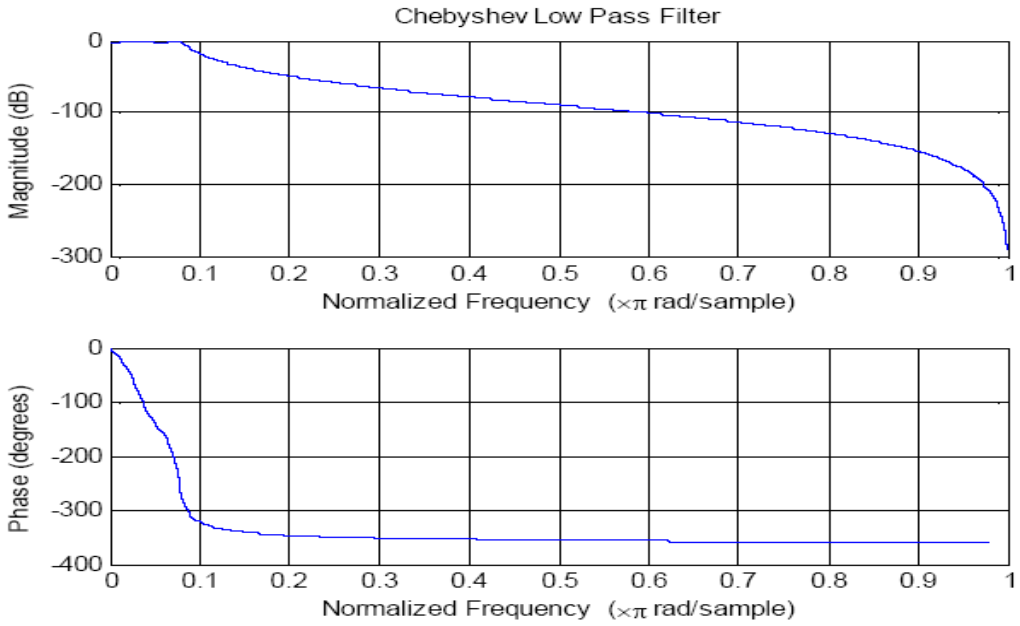
Title:- To develop the program for designing Low pass Type 1
Chebyshev filter having passband defined from 0-40 Hz and stopband in the range of 150-500Hz having less than 3 dB of ripple in the passband and atleast 60dB of attenuation in the stopband.

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Program: -

```
wp=40/500;
ws=150/500;
[n,wn]=cheb1ord(wp,ws,3,60)
n = 4
wn =
0.0800
[b,a]=cheby1(n,3,wn);
freqz(b,a)
title('Chebyshev Low Pass Filter');
```



EXPERIMENT NO: 12

Title: Design FIR filter using windowing technique.

Objective: Illustration of different types of window function.

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

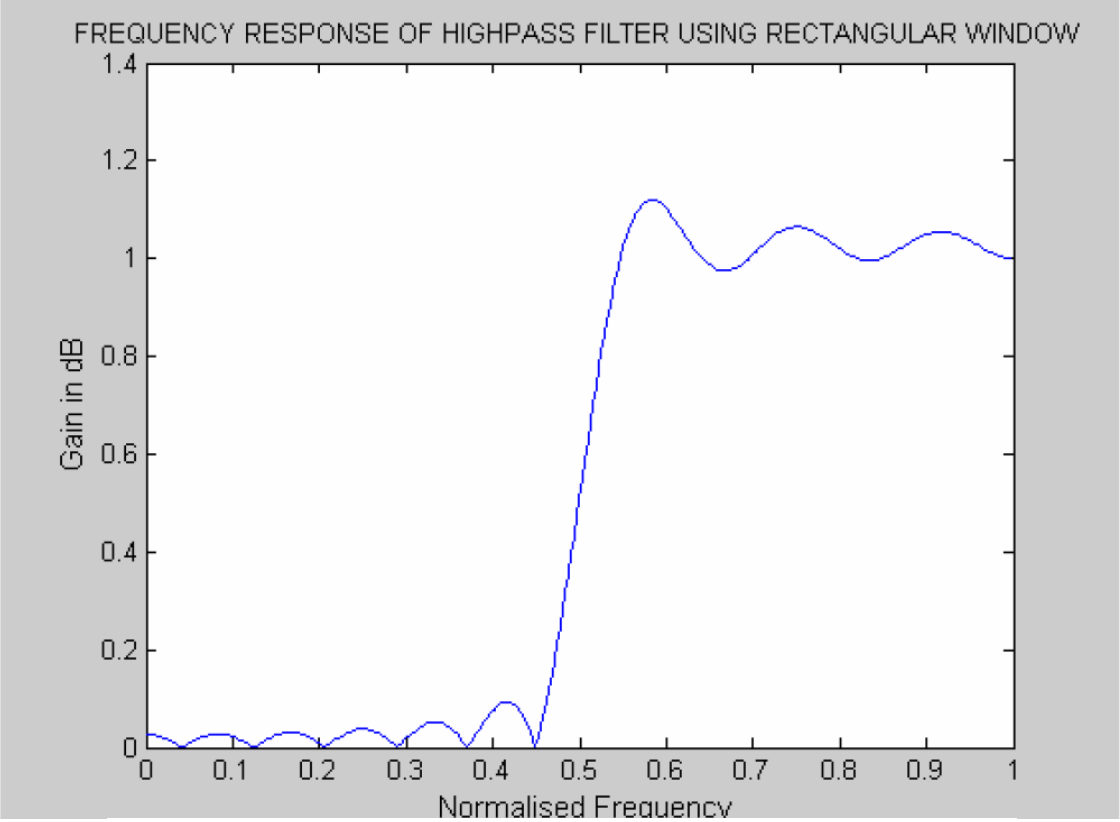
Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Program:

```
%Design of high pass filter using rectangular window
WC=0.5*pi;
N=25;
b=fir1 (N-1, wc/pi,'high', rectwin (N));
disp ('b=');
disp (b)
w=0:0.01: pi;
h=freqz (b, 1, w);
plot (w/pi, abs (h));
xlabel ('Normalised Frequency');
ylabel ('Gain in dB')
title ('FREQUENCY RESPONSE OF HIGHPASS FILTER USING
                                    RECTANGULAR WINDOW');
```

Output:

```
b= Columns 1 through 17
-0.0000   0.0297  -0.0000  -0.0363  -0.0000   0.0467  -0.0000  -0.0654  -
0.0000   0.1090  -0.0000  -0.3269   0.5135  -0.3269  -0.0000   0.1090  -
0.0000
Columns 18 through 25
-0.0654  -0.0000   0.0467  -0.0000  -0.0363  -0.0000   0.0297  -0.0000
```



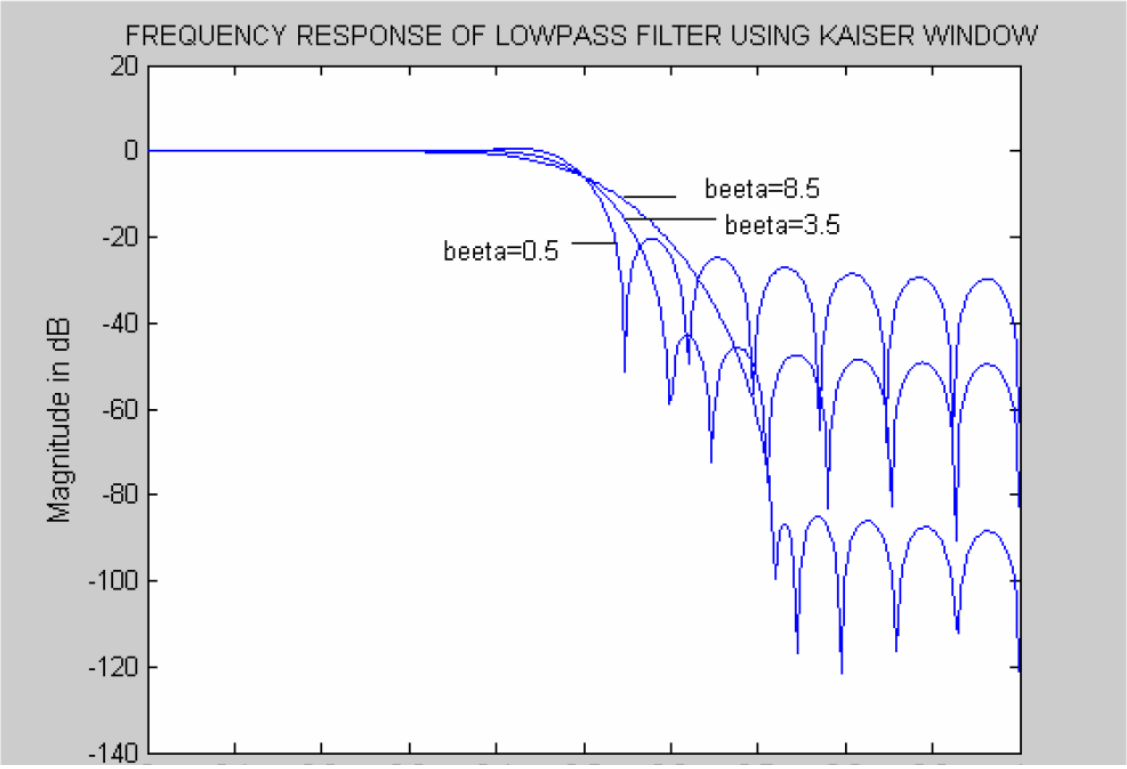
Program:

```
%Design of FIR lowpass filter using Kaiser Window
WC=0.5*pi;
N=25;
b=fir1 (N, wc/pi, kaiser (N+1, 0.5))
w=0:0.01: pi;
h=freqz (b, 1, w);
plot (w/pi, 20*log10 (abs (h)));
hold on
b=fir1 (N, wc/pi, kaiser (N+1, 3.5))
w=0:0.01: pi;
h=freqz (b, 1, w);
plot (w/pi, 20*log10 (abs (h)));
hold on
```

```
b=fir1 (N, wc/pi, kaiser (N+1, 3.5))
w=0:0.01: pi;
h=freqz (b, 1, w);
plot (w/pi, 20*log10 (abs (h)));
hold on
b=fir1 (N, wc/pi, kaiser (N+1, 8.5))
w=0:0.01: pi;
h=freqz (b, 1, w);
plot (w/pi, 20*log10 (abs (h)));
xlabel ('Normalised Frequency');
ylabel ('Magnitude in dB');
title ('FREQUENCY RESPONSE OF LOWPASS FILTER USING
KAISER  WINDOW');
hold off
```

Output:

```
b = Columns 1 through 17
0.0170   -0.0186   -0.0206    0.0229    0.0258   -0.0294   -0.0341    0.0405
0.0497   -0.0641   -0.0899    0.1501    0.4507    0.4507    0.1501   -0.0899   -
0.0641

Columns 18 through 26
0.0497    0.0405   -0.0341   -0.0294    0.0258    0.0229   -0.0206   -0.0186
0.0170
b = Columns 1 through 17
0.0024   -0.0041   -0.0062    0.0089    0.0124   -0.0169   -0.0227    0.0305
0.0412   -0.0573   -0.0849    0.1471    0.4496    0.4496    0.1471   -0.0849   -
0.0573
Columns 18 through 26
0.0412    0.0305   -0.0227   -0.0169    0.0124    0.0089   -0.0062   -0.0041
0.0024
b = Columns 1 through 17
0.0000   -0.0002   -0.0006    0.0015    0.0032   -0.0062   -0.0109    0.0182
0.0293   - 0.0467   -0.0766    0.1416    0.4473    0.4473    0.1416   -0.0766   -
0.0467
```

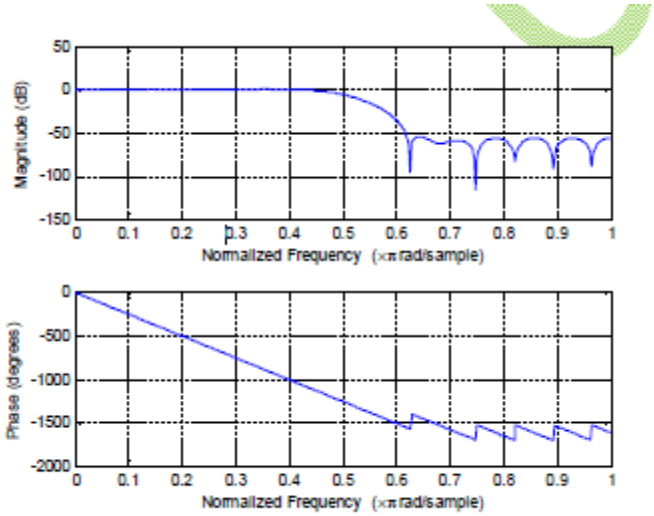


Practice:

% FIR LOW PASS FILTER USING HAMMING WINDOW

```
clc;
clear all;
close all;
N=input('Enter the value of N:');
wc=input('Enter cutoff frequency:');
h=fir1(N,wc/pi,hamming(N+1));
freqz(h);
```

FIGURE:-



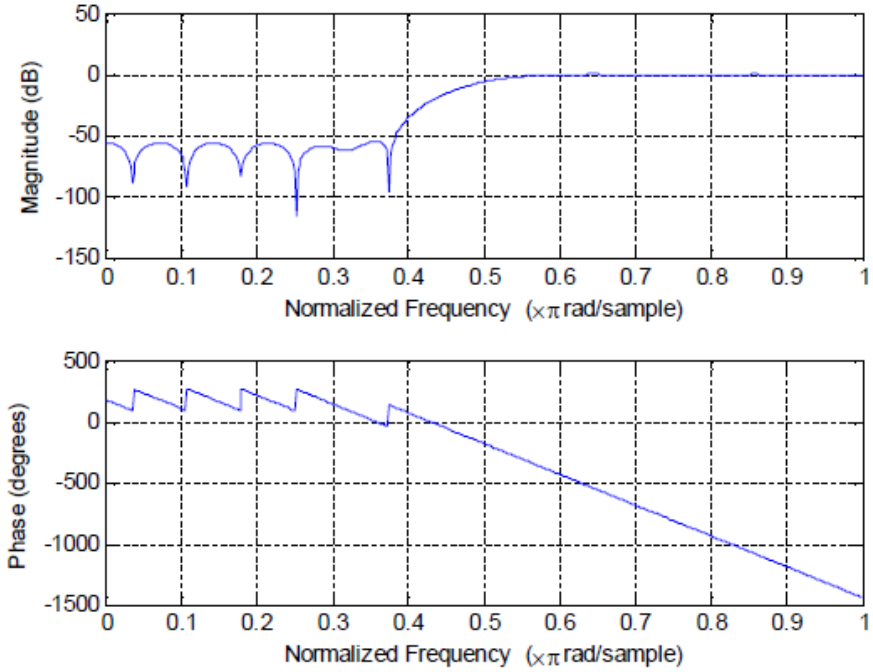
SAMPLE INPUT:-

Enter the value of N:28

Enter cutoff frequency:0.5*pi

% **FIR HIGHPASS FILTER USING HAMMING WINDOW**

```
clc;
clear all;
close all;
N=input('Enter the value of N:');
wc=input('Enter cutoff frequency:');
h=fir1(N,wc/pi,'high',hamming(N+1));
freqz(h);
```



SAMPLE INPUT:-

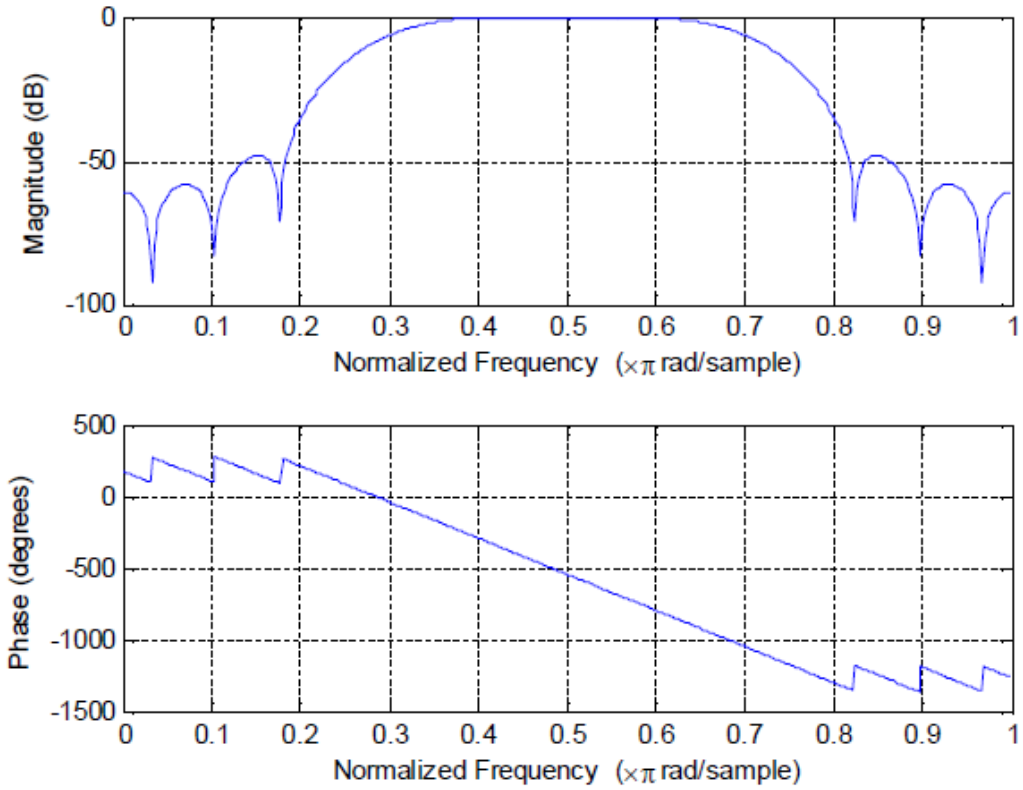
Enter the value of N:28

Enter cutoff frequency:0.5*pi

% **FIR BAND PASS FILTER USING HAMMING WINDOW**

```
clc;
clear all;
close all;
N=input('Enter the value of N:');
wc=input('Enter cutoff frequency:');
h=fir1(N,wc/pi,hamming(N+1));
freqz(h);
```

FIGURE:-



SAMPLE INPUT:-

Enter the value of N:28

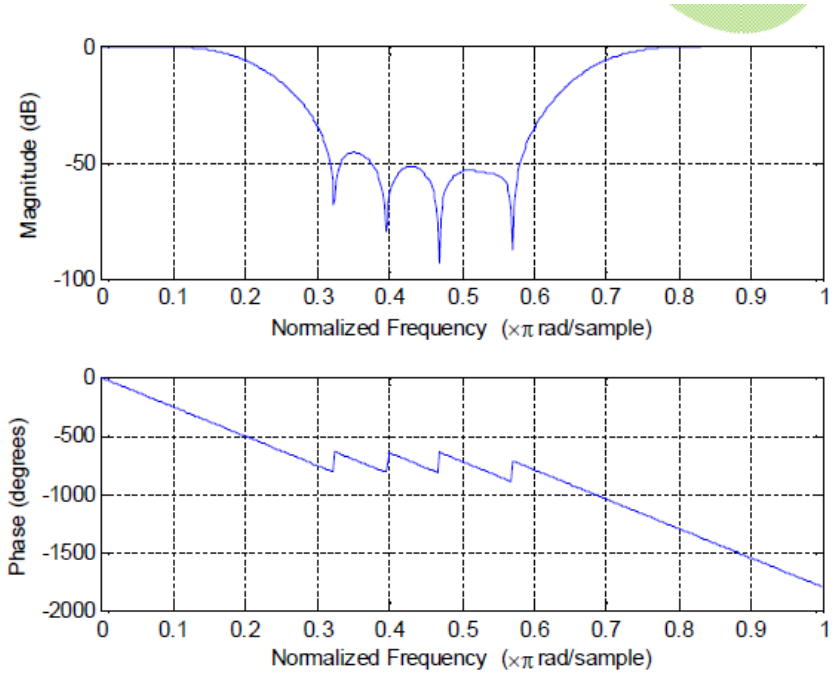
Enter cutoff frequency:[0.3*pi,0.7*pi]

% **FIR BAND STOP FILTER USING HAMMING WINDOW**

```
clc;
clear all;
close all;
N=input('Enter the value of N:');
wc=input('Enter cutoff frequency:');
h=fir1(N,wc/pi,'stop',hamming(N+1));
```

freqz(h);

FIGURE:-



SAMPLE INPUT:-

Enter the value of N:28

Enter cutoff frequency:[0.2*pi,0.7*pi]

EXPERIMENT NO: 13

Title: Design IIR filter

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

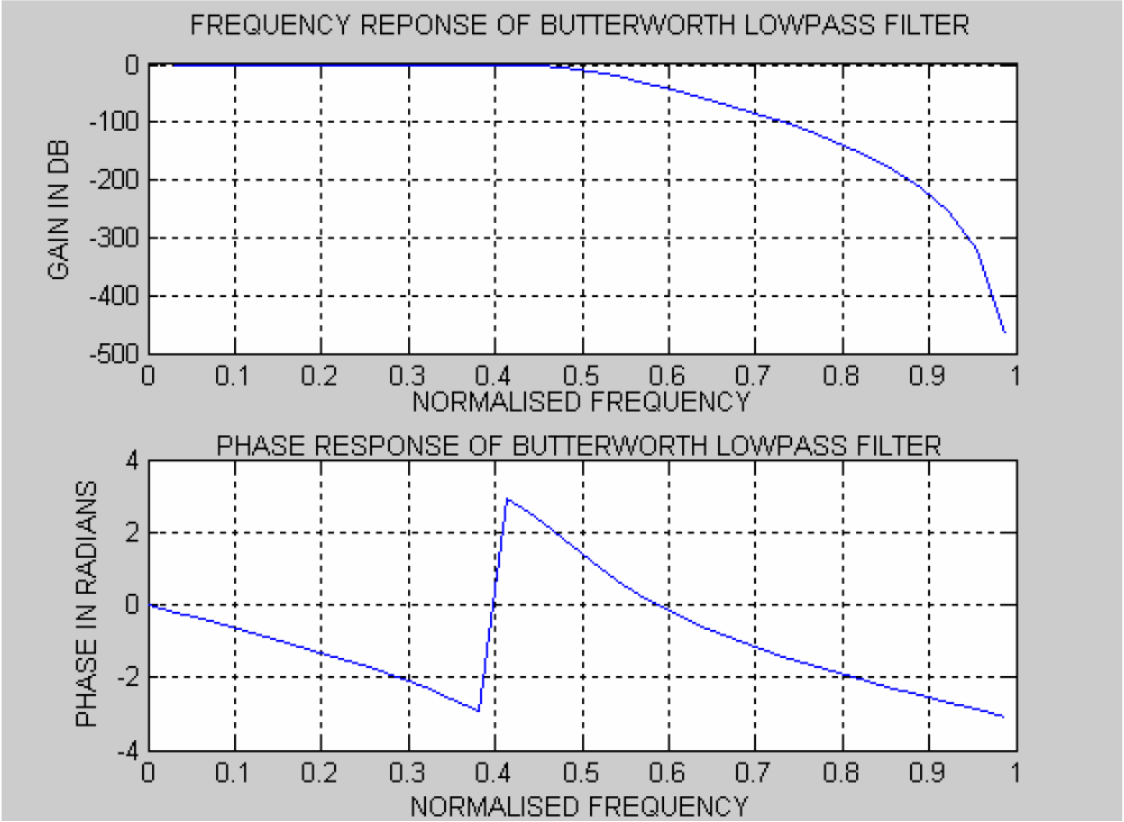
Program:

```
%Design of IIR Butterworth lowpass filter
alphap=input ('enter the pass band attenuation=');
alphas=input ('enter the stop band attenuation=');
fp=input ('enter the passband frequency=');
fs=input ('enter the stop band frequency=');
F=input ('enter the sampling frequency=');
omp=2*fp/F;
oms=2*fs/F;
[n, wn]=buttord (omp, oms, alphap, alphas)
[b, a]=butter (n, wn)
w=0:0.1: pi;
[h, ph]=freqz (b, a, w);

m=20*log (abs (h));
an=angle (h);
subplot (2, 1, 1);
```

```
plot (ph/pi, m);
grid on;
ylabel ('Gain in dB');
xlabel ('Normalised Frequency');
title ('FREQUENCY RESPONSE OF BUTTERWORTH LOWPASS
subplot (2, 1, 2);
title ('PHASE RESPONSE OF BUTTERWORTH LOWPASS FILTER');
plot (ph/pi, an);
grid on;
ylabel ('Phase in Radians');
xlabel ('Normalised Frequency');
```

Output:

```
Enter the pass band attenuation=0.4
Enter the stop band attenuation=60
Enter the pass band frequency=400
Enter the stop band frequency=800
Enter the sampling frequency=2000
n = 6
Wn =0.4914
b =0.0273   0.1635   0.4089   0.5452   0.4089   0.1635   0.0273
a = 1.0000  -0.1024   0.7816  -0.0484   0.1152  -0.0032   0.0018
```



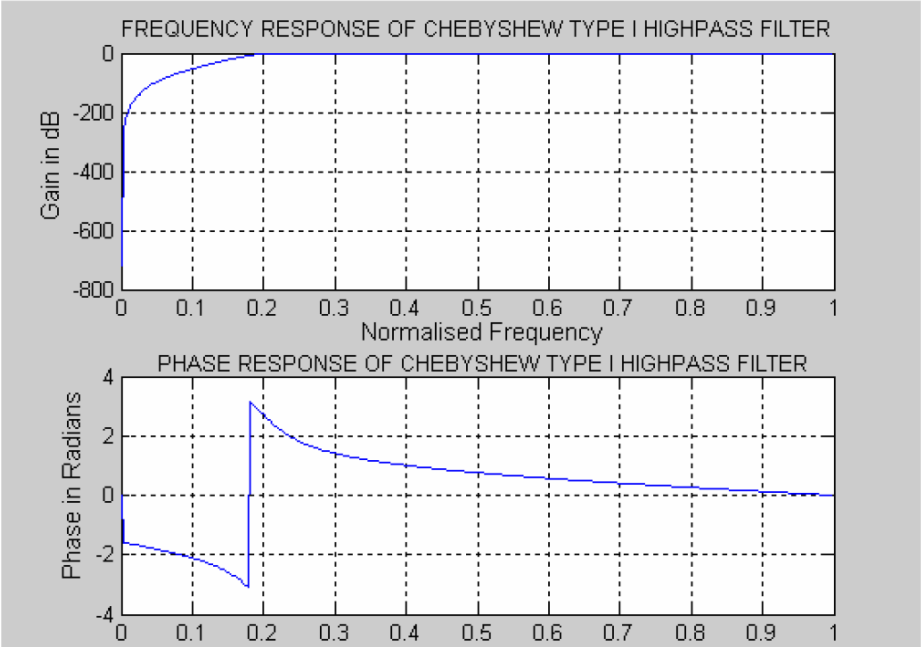
Program:

```
%Design of IIR Chebyshew type I high pass filter
alphap=input ('enter the pass band attenuation=');
alphas=input ('enter the stop band attenuation=');
fp=input ('enter the pass band frequency=');
fs=input ('enter the stop band frequency=');
F=input ('enter the sampling frequency=');
omp=2*fp/F;
oms=2*fs/F;
[n, wn]=cheb1ord (omp, oms, alphap, alphas)
[b, a]=cheby1 (n, wn, high)
w=0:0.1: pi;
[h, ph]=freqz (b, a, w);
m=20*log (abs (h));
an=angle (h);
```

```
subplot (2, 1, 1);
plot (ph/pi, m);
grid on;
ylabel ('Gain in dB');
xlabel ('Normalised Frequency');
title ('FREQUENCY RESPONSE OF CHEBYSHEW TYPE I HIGH PASS FILTER');
subplot (2, 1, 2);
plot (ph/pi, an);
grid on;
ylabel ('Phase in Radians');
xlabel ('Normalised Frequency');
title ('PHASE RESPONSE OF CHEBYSHEW TYPE I HIGH PASS FILTER');
```

Output:

```
enter the pass band attenuation=1
enter the stop band attenuation=15
enter the passband frequency=0.2*pi
enter the stop band frequency=0.1*pi
n = 3
Wn =0.2000
b = 0.4759   -1.4278    1.4278   -0.4759
a = 1.0000   -1.6168    1.0366   -0.1540
```


FREQUENCY RESPONSE OF CHEBYSHEW TYPE I HIGHPASS FILTER
Gain in dB
0
-200
-400
-600
-800
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
Normalised Frequency
PHASE RESPONSE OF CHEBYSHEW TYPE I HIGHPASS FILTER
Phase in Radians
4
2
0
-2
-4
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

EXPERIMENT NO: 14

Title:- Power density spectrum of a sequence.

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

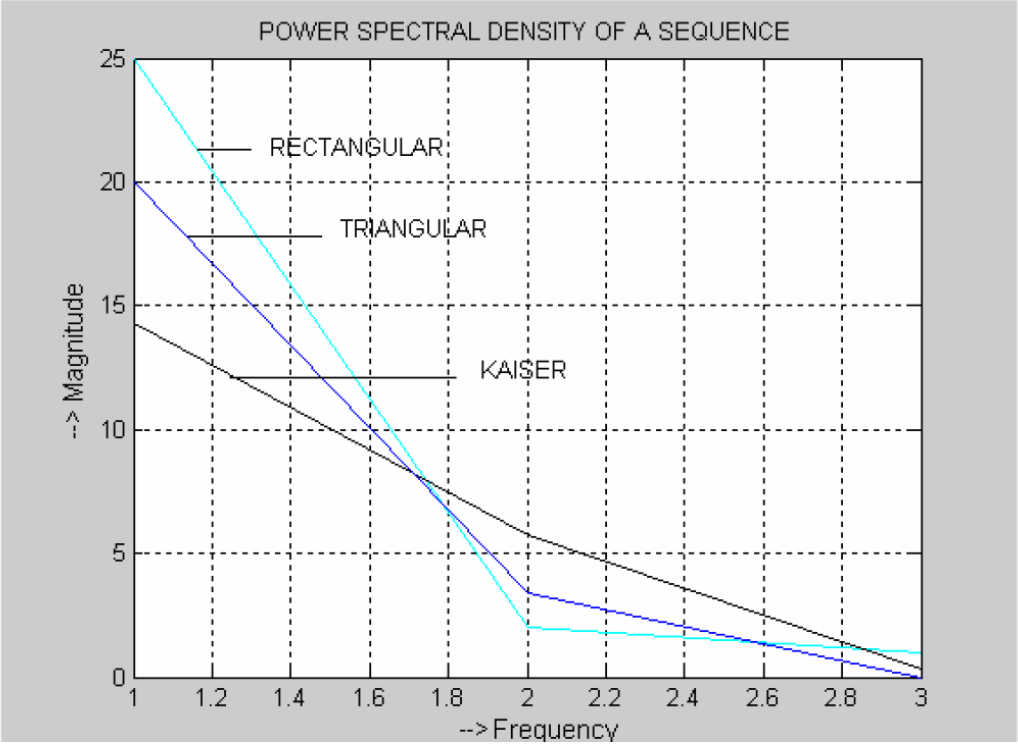
Program:

```
%Power density spectrum of a sequence
x=input ('enter the length of the sequence=');
y=input ('enter the sequence=');
fs=input ('enter sampling frequency=');
N=input ('enter the value of N=');
q=input ('enter the window name1=');
m=input ('enter the window name2=');
s=input ('enter the window name3=');
pxx1=psd(y, N, fs, q)
```

```
plot (pxx1,'c');
hold on;
grid on;
pxx2=psd(y, N, fs, m)
plot (pxx2,'k');
hold on;
pxx3=psd(y, N, fs, s)
plot (pxx3,'b');
hold on;
xlabel ('-->Frequency');
ylabel ('-->Magnitude');
title ('POWER SPECTRAL DENSITY OF A SEQUENCE');
hold off;
```

Output:

```
Enter the length of the sequence=5
Enter the sequence= [1 2 3 4 5]
Enter sampling frequency=20000
Enter the value of N=4
Enter the window name1=rectwin (N)

Enter the window name2=Kaiser (N, 4.5)
Enter the window name3=triang (N)
```


POWER SPECTRAL DENSITY OF A SEQUENCE
RECTANGULAR
TRIANGULAR
KAISER
--> Magnitude
--> Frequency
25
20
15
10
5
0
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3

EXPERIMENT NO: 15

Title:- The objective of this program is To Perform upsampling on the Given Input Sequence.

Outline: Up sampling on the Given Input Sequence and Interpolating the sequence.

Task: illustration for finding the magnitude and phase response of system described by system function H(s).

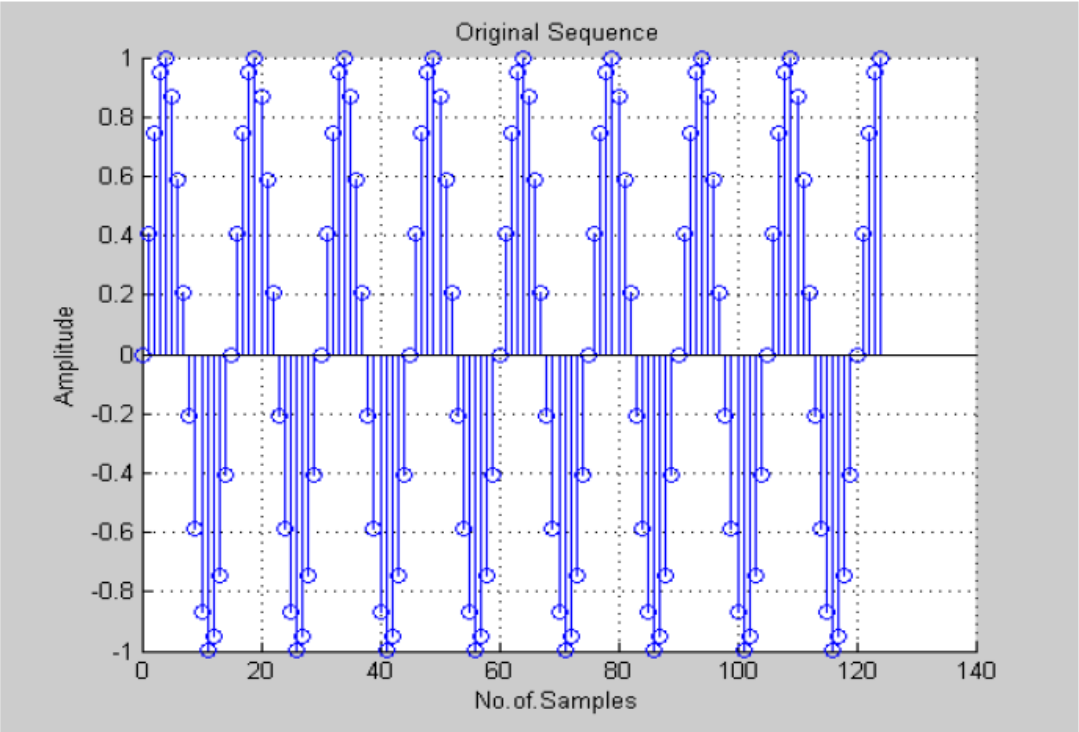
Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

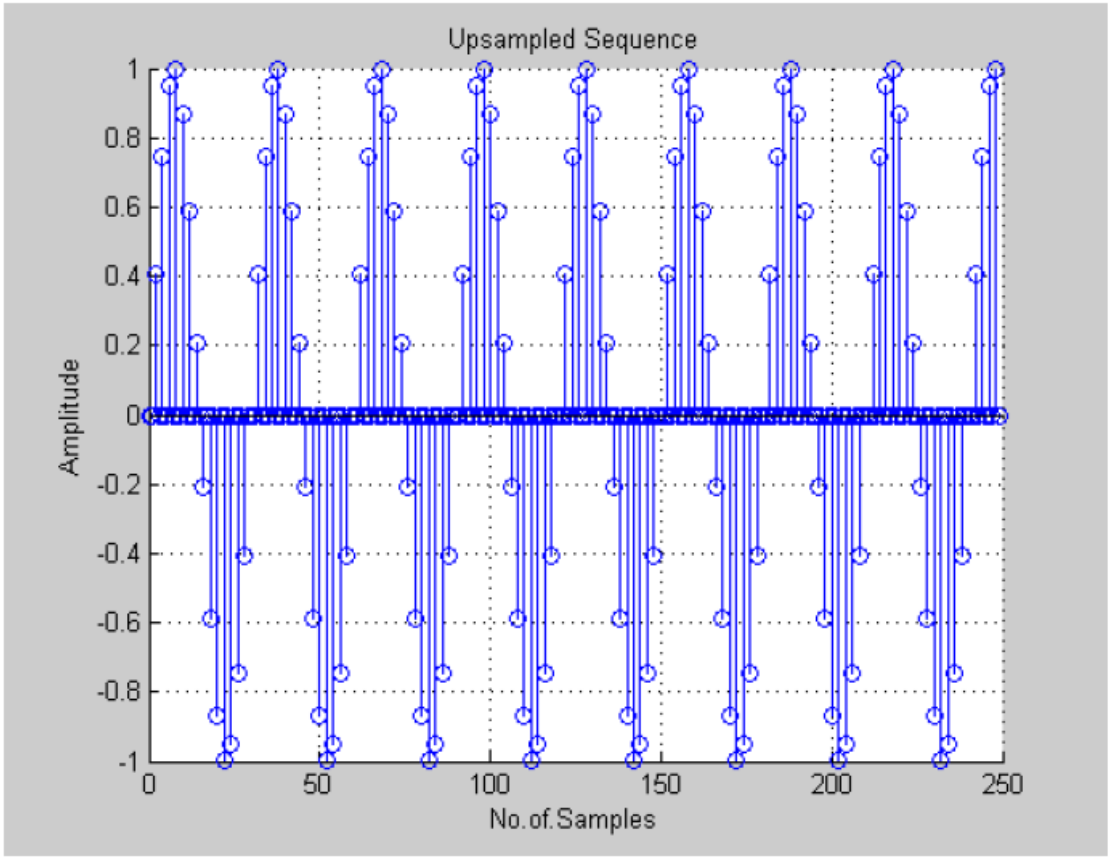
PROGRAM:

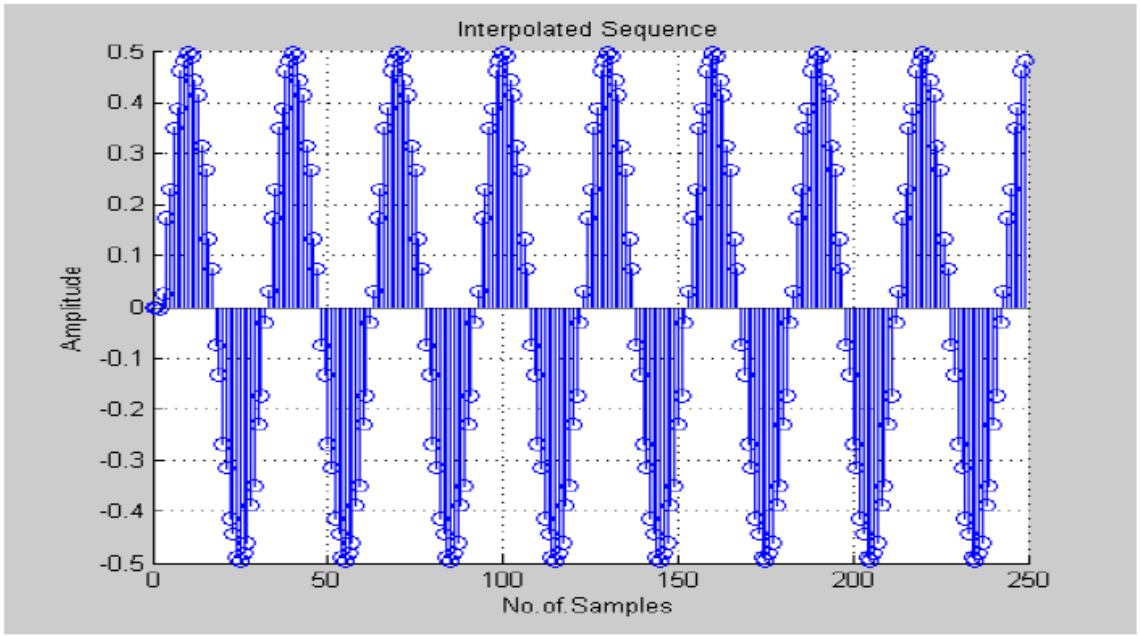
```
clc;
clear all;
close all;
N=125;
n=0:1:N-1;
x=sin(2*pi*n/15);
L=2;
figure(1)
stem(n,x);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Original Sequence');
x1=[zeros(1,L*N)];
n1=1:1:L*N;
j =1:L:L*N;
x1(j)=x;
figure(2)
stem(n1-1,x1);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Upsampled Sequence');
a=1;
b=fir1(5,0.5,'Low');
y=filter(b,a,x1);
figure(3)
stem(n1-1,y);
grid on;
xlabel('No.of.Samples');
```

```
ylabel('Amplitude');
title('Interpolated Sequence');
```

% **Graph:**


Original Sequence
Amplitude
No.of.Samples
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140





EXPERIMENT NO: 16

EXPERIMENT NO: 16

Title:- The objective of this program is To Perform Decimation on the Given Input Sequence.

Objective: Decimation on the Given Input Sequence by using filter with filter-coefficients a and b.

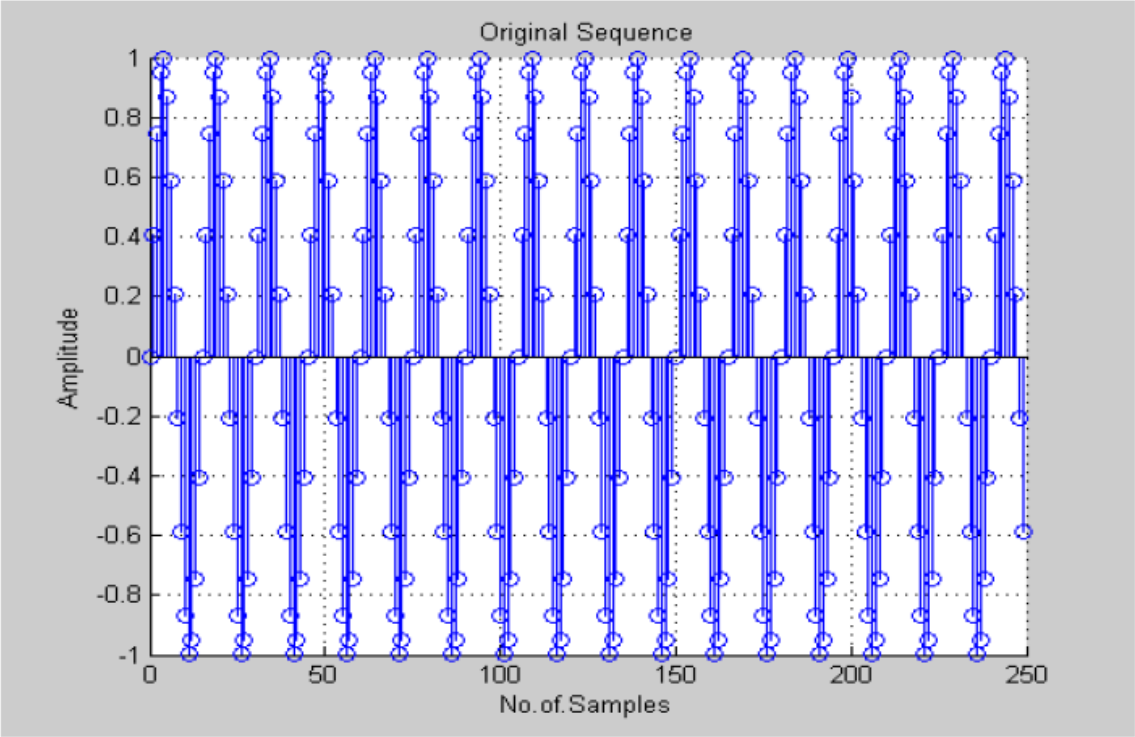
Task: illustration for finding the magnitude and phase response of system described by system function H(s).

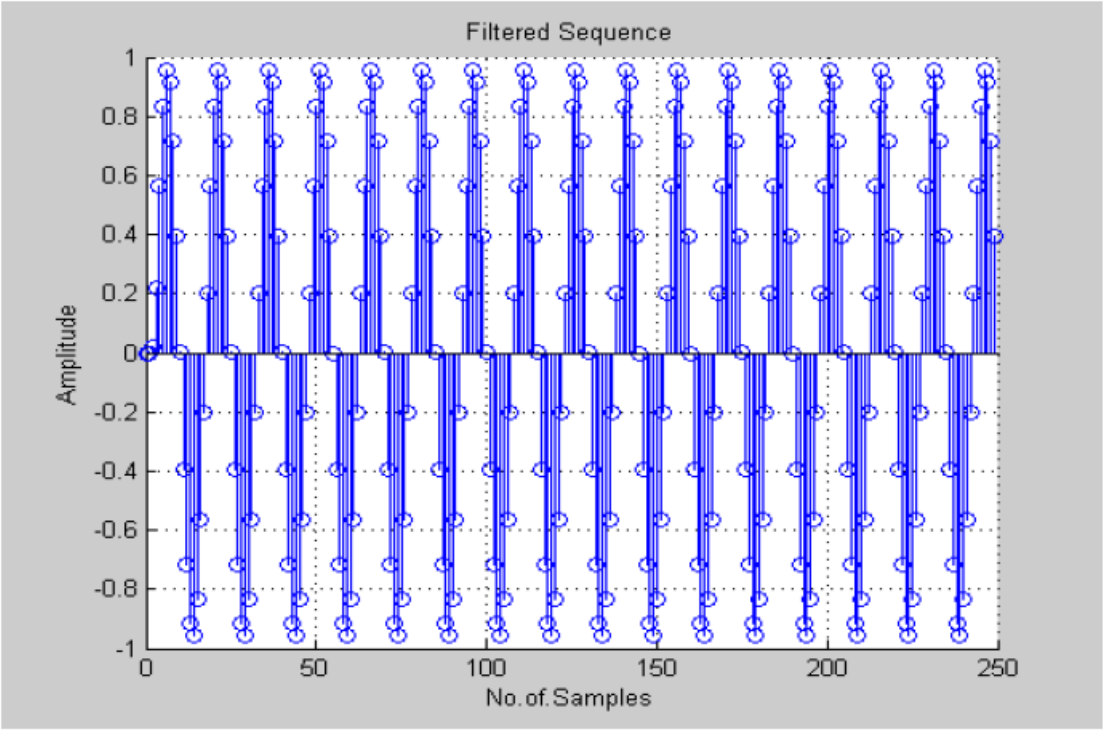
Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

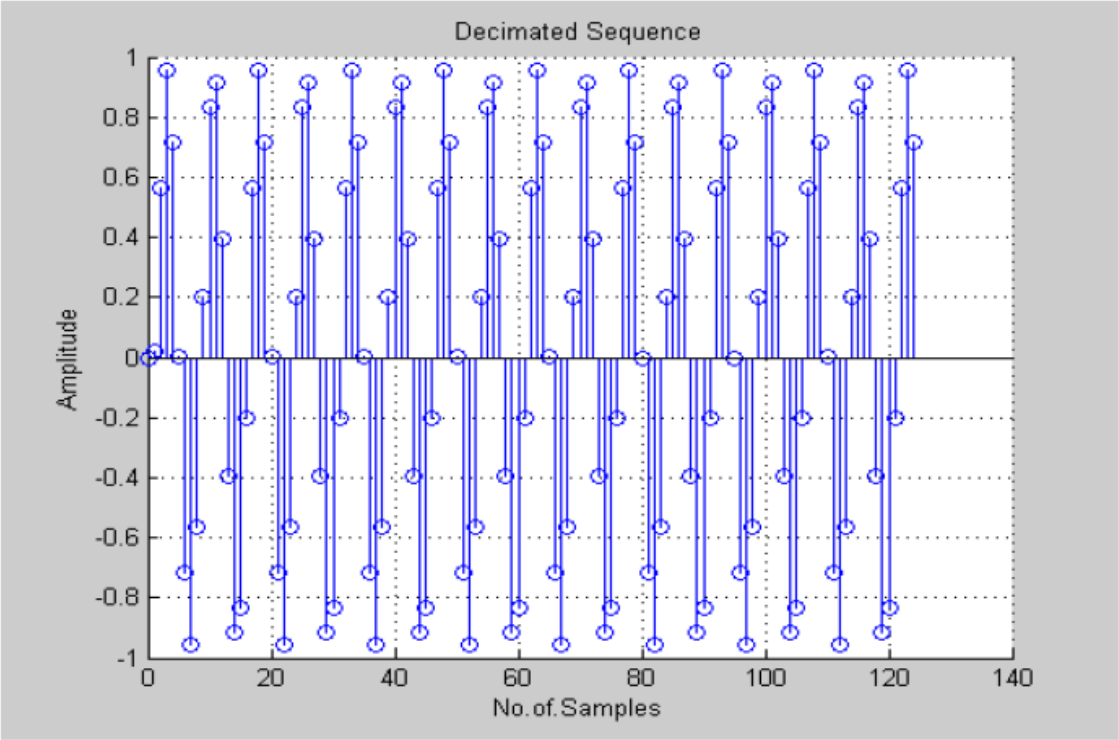
PROGRAM:

```
clc;
clear all;
close all;
N=250 ;
n=0:1:N-1;
x=sin(2*pi*n/15);
M=2;
figure(1)
stem(n,x);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Original Sequence');
a=1;
b=fir1(5,0.5,'Low');
y=filter(b,a,x);
figure(2)
stem(n,y);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Filtered Sequence');
x1=y(1:M:N);
n1=1:1:N/M;
figure(3)
stem(n1-1,x1);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Decimated Sequence');
```

% Graph:


Original Sequence
Amplitude
No. of Samples
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250


Filtered Sequence
Amplitude
No. of Samples
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
50
100
150
200
250


Decimated Sequence
Amplitude
No.of.Samples
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140

Experiment:-17

Title:- Analysis of Z transform and Inverse Z Transform.

Task: Analysis of Z-Transforms and Inverse Z Transform.

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

```
%Analysis of Z-Transforms %Definition of numerators and denominator coefficients
num=[5 6 -44 21 32];
den=[5 13 15 18 -12]; %Conversion from rational to Factored form
[z,p,k]=tf2zp(num,den);
disp('Zeros are at');disp(z);
disp('Poles are at');disp(p);
disp('Gain Constant');disp(k); %Determination of radius of the poles
radius=abs(p);
disp('Radius of the poles');
disp(radius); %Pole Zero Map using numerator and denominator coefficients
zplane(num,den) %Conversion from factored to secomd ordered factored
sos=zp2sos(z,p,k)
disp('Second Order Sections');disp(sos);
```

```
%Inverse Z-Transform using impz
%definition of numerator and denominator coefficients
num=[0.1+.4*i 5 .05];
den=[1 .9+0.3*i .12];
%Finding the inverse z transform of G(z)
[a,b]=impz(num,den);
%Evaluating on Unit Circle i.e. Fourier Transform
[c,d]=freqz(num,den);
% Plotting of x[n] and it's fourier transform
subplot(2,2,1)
stem(b,real(a))
title('Real Part of g[n]')
xlabel('Samples'); ylabel('Magnitude')
grid on
subplot(2,2,2)
stem(b,imag(a))
title('Imaginary Part of g[n]')
xlabel('Samples'); ylabel('Magnitude')
grid on
subplot(2,2,3)
```

```
plot(d/pi,abs(c))
title('Magnitude Spectrum of g[n]')
xlabel('\omega/\pi'); ylabel('Magnitude')
grid on
subplot(2,2,4)
plot(d/pi,angle(c))
title('Phase Spectrum of g[n]')
xlabel('\omega/\pi'); ylabel('Phase, radians')
grid on
```

Experiment:-18

Title:- To develop an application with speech signal .

Task: Application of DSP on real world signals.

Mode of Evaluation: Presentation, Implementation & Testing, and Viva.

Program : Add noise at a particular SNR to a speech signal.

```
clc
clear
%%%%%%%%%read of clean speech file%%%%%%%%

[x, fs]=wavread('2.wav');  %fs=sampling frequency.  x=sample point(the sampled data in y)
block=256;%floor(0.023*fs); block=number of sample data in each block.
n=floor(length(x)/block);% number of blocks

x=x(1:block*n);% x will start from 1 and go upto blockno*no of block=total data

figure,plot(x);
title('Original signal');

%%% Generate noise using random number (white noise) %%%%%%%%%%

db=0; % SNR
%signal=x/max(abs(x));
sig_var=cov(x); % covariance of x
randn('state',0); % use default initial stage of random number
v1=randn(1,size(x)); % generate random number (noise), length is 1 to size of x
noise_var=sig_var*10^(-db/10); % find noise variance for the particular SNR
v=sqrt(noise_var)*v1; % modify noise with noise variance
xn=x+v'; % add noise with clean speech we have noisy speech
wavwrite(xn,11000,'nois.wav');% making wav file of noisy speech
figure,plot(xn);
```