DIGITAL SIGNAL PROCESSING LABORATORY

MASTER MANUAL

DEPARTMENT

OF

ELECTRONICS & COMMUNICATION ENGINEERING

SCHOOL OF STUDIES OF ENGINEERING AND TECHNOLOGY

GURU GHASIDAS VISHWAVIDYALAYA BILASPUR

(CENTRAL UNIVERSITY)

KONI, BILASPUR-495009

B. Tech. III Year V Semester

EC205PPC08 DIGITAL SIGNAL PROCESSING LAB

LIST OF EXPERIMENTS:

1. To generate the random sequences and determine the correlation.
2. To verify linear and circular convolutions.
3. To compute DFT of sequence and its spectrum analysis.
4. To implement 8-point FFT algorithm.
5. To design of FIR filters using rectangular window techniques.
6. To design of FIR filters using triangular window techniques.
7. To design of FIR filters using kaiser window.
8. To design of butterworth IIR filter.
9. To design of chebyshev IIR filter.
10. To generate the down sample (decimation) by an integer factor,
11. To generate the up sample (interpolation) by an integer factor
12. To remove the noise in 1-D and 2-D signals

1. PROGRAM TO GENERATE RANDOM SEQUENCES AND PERFORM CORRELATION OF 2 SEQUENCES

AIM:

To generate a random sequence and perform cross correlation between two sequences.

APPARATUS:

PC with MATLAB

THEORY:

(a) Generation of Random Sequences

Matlab has two functions for generating random numbers, which can be added to signals to model noise.

- A = rand(m,n)-generates an mxn array of random numbers from the uniform distribution on the interval[0,1].
- r = randi(imax,n) returns an n-by-n matrix containing pseudorandom integers drawn from the discrete uniform distribution on the interval [1,imax].
- R = randn(sz,arraytype) creates a matrix with underlying class of double, with random values in all elements both positive and negative.
- p = randperm(n,k) returns a row vector containing k unique integers selected randomly from 1 to n numbers.

(b) Correlation of 2 sequences

Correlation:

It is a mathematical tool used to compare 2 signals.It has a significant role in signal processing.We have 2 types of correlations.

Auto Correlation:

Autocorrelation, also known as **s**erial correlation, is the correlation of a signal with itself at different points in time. Informally, it is the similarity between observations as a function of the time lag between them. It is a mathematical tool for finding repeating patterns, such as the presence of a periodic signal obscured by noise, or identifying the missing fundamental frequency in a signal implied by its harmonic frequencies.

It is given by $R_{xx}(l)= \sum x(n)*x(n-l)$

Cross Correlation:

In signal processing, cross-correlation is a measure of similarity of two series as a function of the displacement of one relative to the other. Cross correlation between a pair of signals x(n) and y(n) is given by,

- $R_{xy}(l)= \sum x(n)*y(n-l)$

The index l is called lag parameter and the subscript xy indicates that x(n) is reference sequence that remains unshifted in time and y(n) is shifted l units in time with respect to x(n).

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory

- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

(b) Generation of Random Sequences

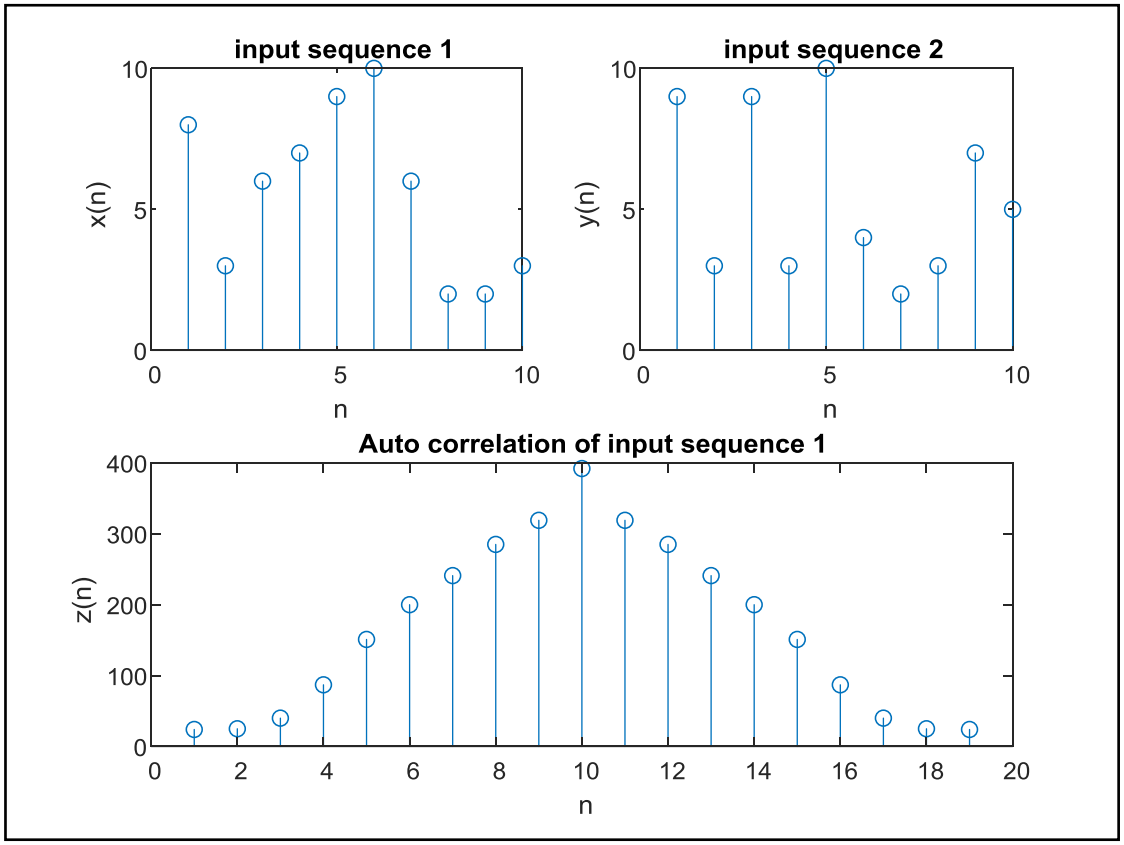
```
% Generate the random sequences and determine the
correlation.

clc;
close all;
clear all;
% two input sequences
l=input('enter input sequence length')
x = randi(10,1,l);
y = randi(10,1,l);
subplot(2,2,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence');

subplot(2,2,2);
stem(y);
xlabel('n');
ylabel('y(n)');
title('input sequence');
% cross correlation of input sequence
z=xcorr(x,y);
disp('The values of z are = '); disp(z);
subplot(2,1,2);
stem(z);
xlabel('n');
ylabel('z(n)');
title('Cross correlation of input sequence');

z1=xcorr(x,x);
disp('The values of z are = '); disp(z1);
figure(2)
subplot(2,2,1);
stem(x);
xlabel('n');
ylabel('x(n)');
title('input sequence');

subplot(2,2,2);
stem(y);
xlabel('n');
ylabel('y(n)');
```

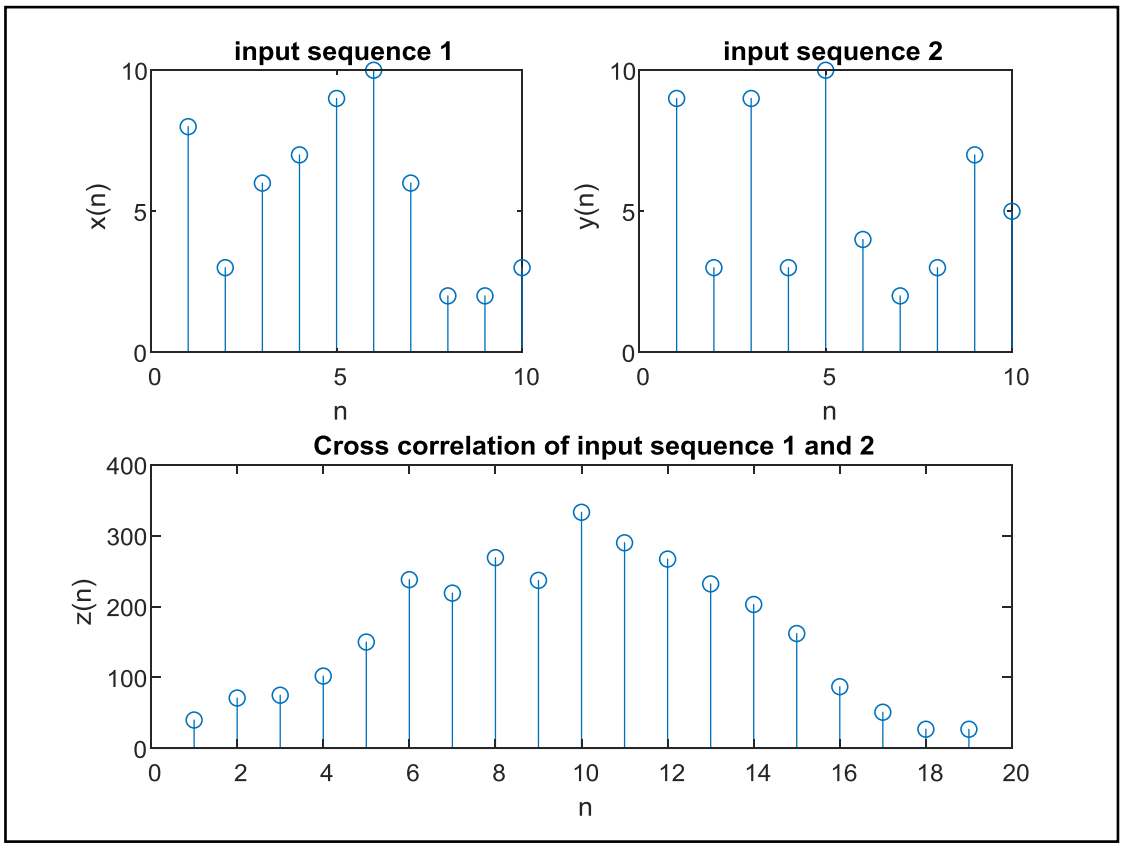
```
title('input sequence');

subplot(2,1,2);
stem(z1);
xlabel('n');
ylabel('z(n)');
title('Auto correlation of input sequence');
```

INPUT:

Enter the sequence length: 10.

OUTPUT:





RESULT:

Two Random sequence, and its auto correlation, cross correlation is generated in MATLAB and figures are plotted showing all the specifications.

2. LINEAR CONVOLUTION AND CIRCULAR CONVOLUTION

(a) LINEAR CNVOLUTION

AIM:

To perform linear convolution of two sequences without using in-built function.

APPARATUS:

PC with MATLAB

THEORY:

Convolution is a formal mathematical operation, just as multiplication, addition, and integration. Addition takes two *numbers* and produces a third *numbe*r, while convolution takes two *signals* and produces a third *signa*l. Convolution is used in the mathematics of many fields, such as probability and statistics. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal.

$$y(n) = \sum_{k=0}^{N-1} x_1(k)x_2(n-k) \quad 0 < n < N-1$$

In Linear convolution if $x_1(n)$ has L samples and $x_2(n)$ has M samples then the linear convoluted output y(n) has a total of L+M-1 number of samples.

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

```
% To verify Linear convolutions.

close all;
clear all;
clc;
clearvars
x=input('Enter x:      ')
```

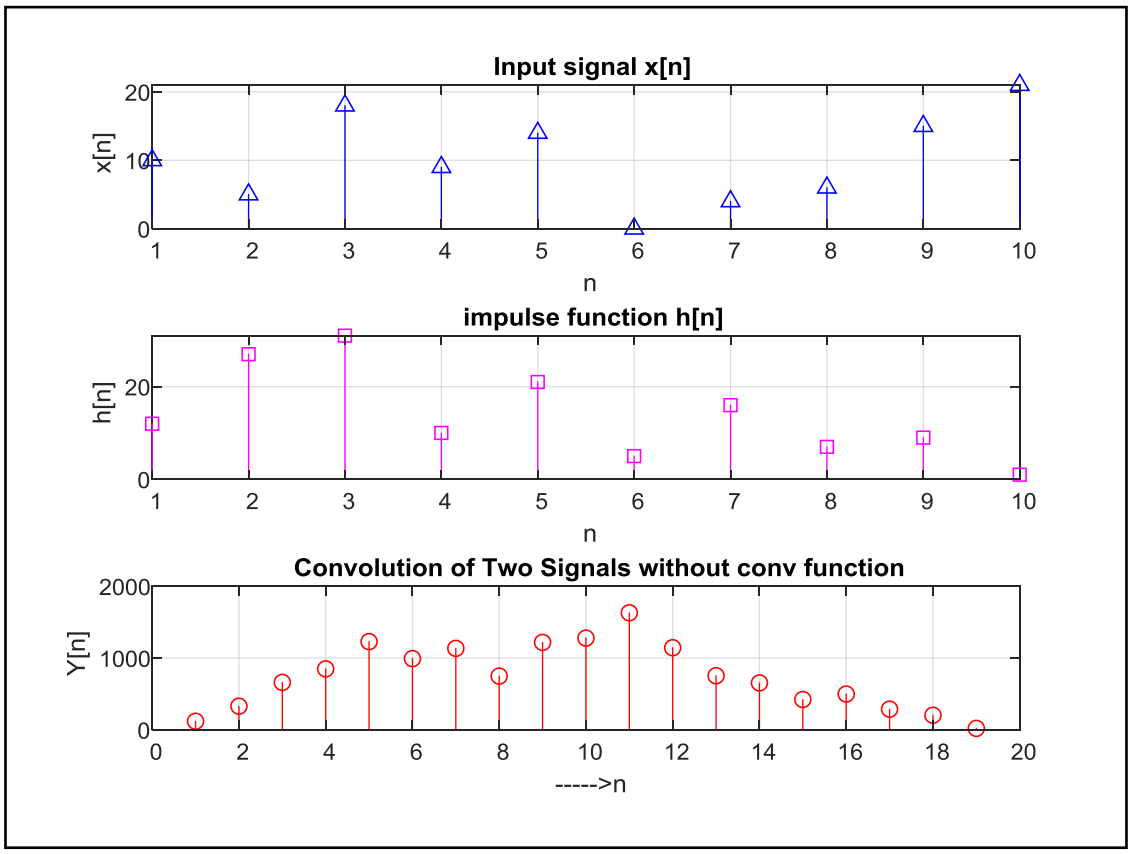
```
% x=sin(2*pi*0.1.*(1:1:11));
h=input('Enter h:    ')
% h=[1 2 3 4 5 3 1 -1];
% convolution
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
for i=1:n+m-1
    Y(i)=0;
    for j=1:m
        if(i-j+1>0)
            Y(i)=Y(i)+X(j)*H(i-j+1);
        else
        end
    end
end
% plot results
figure;
subplot(3,1,1); stem(x, '-b^'); xlabel('n');
ylabel('x[n]'); grid on;
title('Input signal x[n]');
subplot(3,1,2); stem(h, '-ms');
xlabel('n'); ylabel('h[n]'); grid on;
title('impulse function h[n]');
subplot(3,1,3); stem(Y, '-ro');
ylabel('Y[n]'); xlabel('----->n'); grid on;
title('Convolution of Two Signals without conv function');
```

INPUT:

Enter x: [10 5 18 9 14 0 4 6 15 21]

Enter h: [12 27 31 10 21 5 16 7 9 1]

OUTPUT:



RESULT:

Without using in-built function for convolution, linear convolution for two input sequences is performed.

(b).CIRCULAR CONVOLUTION

AIM:

To perform circular convolution of two sequences without using in-built function for circularconvolution.

APPARATUS:

PC with MATLAB

THEORY:

Given two sequences $x_1(n)$ and $x_2(n)$, then the circular convolution of these 2 sequences is given by $x_3(n)= x_1(n)$ N $x_2(n)$ which is given by the following equation,

$x_3(n) = \sum_{m=0}^{N-1}$ x1(m) $x_2((n-m))_N$

It can be found by 2 methods:

1. Concentric circle method
2. Matrix Multiplication method

If $x_1(n)$ has L number of samples and $x_2(n)$ has M number of samples and L>M, then circular convolution between the 2 sequences can be performed by taking N=max(L,M) by adding L-M number of zero samples to the sequence $x_2(n)$so that both the sequences are periodic with N.

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

```
% To verify Circular convolutions.

clc;
close all; clear all;
x=input('Enter x(n):\n');
h=input('Enter h(n):\n');
m=length(x);%length of sequence x(n)
n=length(h);%length of sequence h(n)
N=max(m,n);%length of output sequence y(n)
%For equating both sequence length
x=[x,zeros(1,N-m)];
```

```
h=[h,zeros(1,N-n)];
for n=1:N
    Y(n)=0;
    for i=1:N
        j=n-i+1;
        if(j<=0)
            j=N+j;
        end
        Y(n)=[Y(n)+x(i)*h(j)];
    end
end
n=0:N-1;%Range of all Sequences
figure('Name','Anil Kumar Soni');
subplot(311)
disp('First Sequence x(n) is:')
disp(x)
stem(n,x)
xlabel('n')
ylabel('x(n)')
title('First Sequence')
grid on;
subplot(312)
disp('Second Sequence h(n) is:')
disp(h)
stem(n,h)
xlabel('n')
ylabel('h(n)')
title('Second Sequence')
grid on;
subplot(313)
disp('Convoluted Sequence Y(n) is:')
disp(Y)
stem(n,Y)
xlabel('n')
ylabel('Y(n)')
title('Circular Convoluted Sequence')
grid on;
```

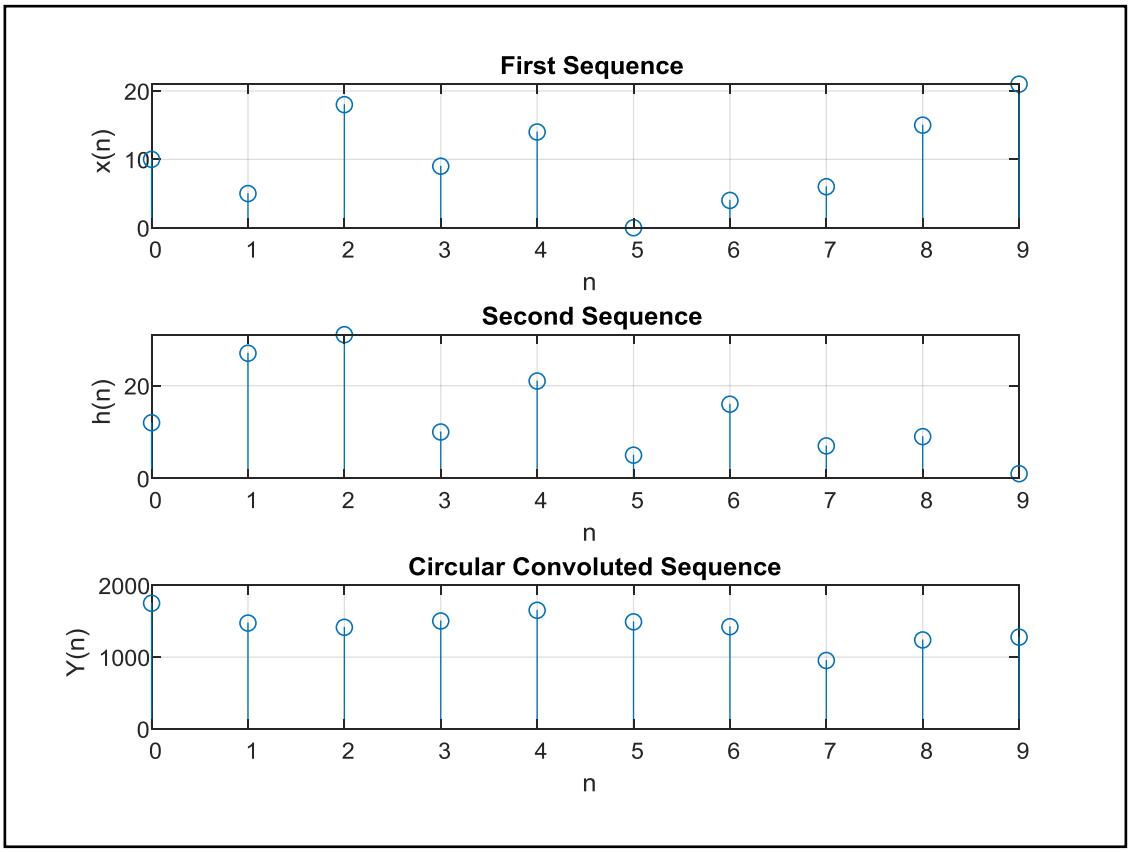
INPUT:

Enter x: [10 5 18 9 14 0 4 6 15 21]

Enter h: [12 27 31 10 21 5 16 7 9 1]

OUTPUT:

Circular convolution of two sequences is.



RESULT:

Without using built-in function for convolution, circular convolution of two input sequences is performed.

3. SPECTRAL ANALYSIS USING DFT

AIM:

To perform spectral analysis on a signal using Discrete Fourier Transform and plot the power distribution of the signal versus frequency graph

APPARATUS:

PC with MATLAB

THEORY:

The discrete Fourier transform (DFT) maps a finite number of discrete time-domain samples to the same number of discrete Fourier-domain samples. Being practical to compute, it is the primary transform applied to real-world sampled data in digital signal processing. The DFT has special relationships with the discrete-time Fourier transform and the continuous-time Fourier transform that let it be used as a practical approximation of them through truncation and windowing of an infinite-length signal. Different window functions make various tradeoffs in the spectral distortions and artifacts introduced by DFT-based spectrum analysis.

The DFT transforms *N* samples of a discrete-time signal to the same number of discrete frequency samples, and is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{i2\pi nk}{N}}$$

The DFT is invertible by the inverse discrete Fourier transform (IDFT):

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{i\frac{2\pi nk}{N}}$$

The DFT and IDFT are a self-contained, one-to-one transform pair for a length-*N*discrete-time signal. The DFT is notmerely **a**n approximation to the DTFT. However, the DFT is very often used as a practical approximation to the DTFT.

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

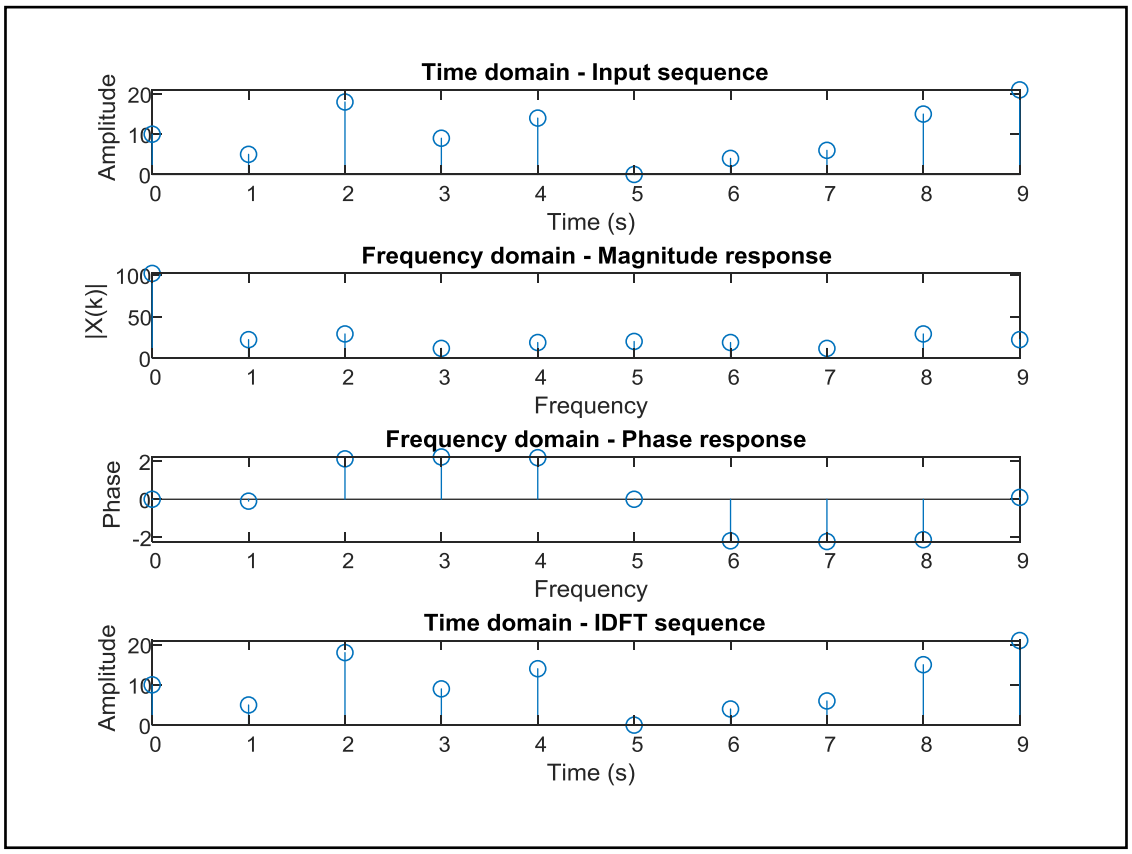
```
% To compute DFT of sequence and its Spectrum Analysis.
clc; clear all; close all;
x=input('enter input sequence') % x = [2 3 -1 4];
N = length(x);
X = zeros(N,1)
```

```
% code for DFT
for k = 0:N-1
    for n = 0:N-1
        X(k+1) = X(k+1) + x(n+1)*exp(-j*pi*2*n*k/N)
    end
end
% code for IDFT
xk = zeros(N,1)
for k = 0:N-1
    for n = 0:N-1
        xk(k+1) = xk(k+1) + X(n+1)*exp(j*pi*2*n*k/N)
    end
end
xk = xk./N;

t = 0:N-1
subplot(411)
stem(t,x);
xlabel('Time (s)');
ylabel('Amplitude');
title('Time domain - Input sequence')

subplot(412)
stem(t,abs(X))
xlabel('Frequency');
ylabel('|X(k)|');
title('Frequency domain - Magnitude response')

subplot(413)
stem(t,angle(X))
xlabel('Frequency');
ylabel('Phase');
title('Frequency domain - Phase response')

subplot(414)
stem(t,xk)
xlabel('Time (s)');
ylabel('Amplitude');
title('Time domain - IDFT sequence')
```

INPUT:

Enter x: [10 5 18 9 14 0 4 6 15 21]

OUTPUT:



RESULT:

DFT Spectral analysis on a continuous time signal was performed and the Power density spectral graph with respect to frequency was plotted

4. 8-POINT FFT ALGORITHM

AIM:

To implement 8-point FFT algorithm without using matlab inbuilt function.

APPARATUS:

PC with MATLAB

THEORY:

The DFT is the only transform that is discrete in both the time and the frequency domains, and is defined for finite-duration sequences. Although it is a computable transform, the straightforward implementation of it is very inefficient, especially when the sequence length *N* is large. In 1965 Cooley and Tukey showed a procedure to substantially reduce the amount of computations involved in the DFT. This led to the explosion of applications of the DFT, including in the digital signal processing area. Furthermore, it also led to the development of other efficient algorithms. All these efficient algorithms are collectively known as fast Fourier transform (FFT) algorithms.
Consider an *N*-point sequence *x*(*n*). Its *N*-point DFT is given by

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}, \quad 0 \le k \le N-1$$

Where W_N=exp (-j*2*pi)/N.

The FFT algorithms exploit 2 properties of twiddle factor (W_N) and reduces the number of complex multiplications to perform DFT from N^2 to (N/2)*$\log_2$N.This implies about 5000 instead of 10^6 multiplications, a reduction factor of 200.

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

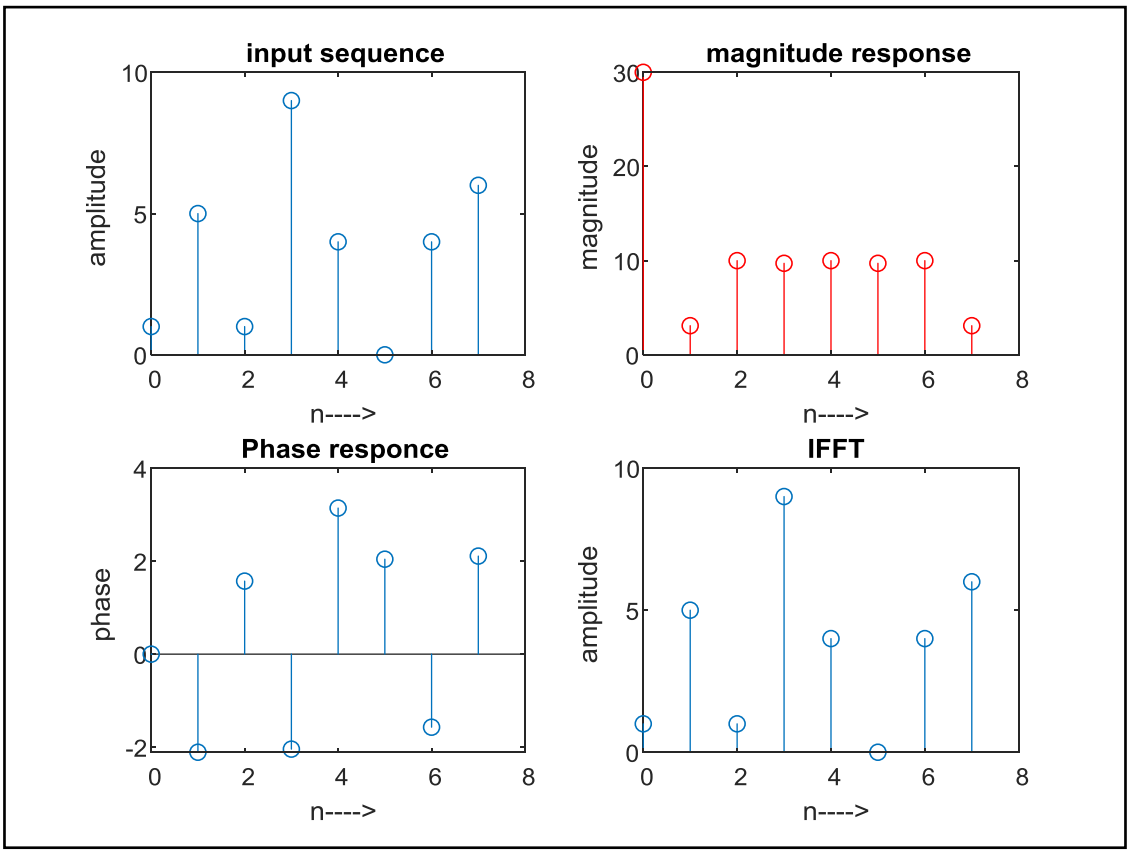
```
% To implement 8-point FFT algorithm.

clc;
clear all;
close all;
x = input('Enter the N = 8 sequence : ');
N1 = length(x);
```

```
xK1 = fft(x,N1);
xn1 = ifft(xK1);
n=0:N1-1;
subplot (2,2,1);
stem(n,x);
xlabel('n---->');
ylabel('amplitude');
title('input sequence');
subplot (2,2,2);
stem(n,abs(xK1),'r');
xlabel('n---->');
ylabel('magnitude');
title('magnitude response');
subplot (2,2,3);
stem(n,angle(xK1));
xlabel('n---->');
ylabel('phase');
title('Phase responce');
subplot (2,2,4);
stem(n,xn1);
xlabel('n---->');
ylabel('amplitude');
title('IFFT');
```

INPUT:

Enter sequence x = [1 5 1 9 4 0 4 6]

OUTPUT:



RESULT:

For the given 8 point sequence, DFT is obtained using 8 point DIT FFT and the corresponding magnitude and phase of the output are plotted.

5. FIR FILTER (LP/HP) USING RECTANGULAR WINDOW TECHNIQUE

AIM:

To design a FIR filter using Rectangular windowing technique and verify its frequency response.

APPARATUS:

PC with MATLAB

THEORY:

FIR filter is described by the difference equation

$$y[n] = \sum_{l=0}^{M} b_l x[n-l]$$

and by the transfer function

$$H(e^{j\omega}) = \frac{Y(e^{j\omega})}{X(e^{j\omega})} = \sum_{l=0}^{M} b_l e^{-j\omega l}$$

The window design method does not produce filters that are optimal but the method is easy to understand and does produce filters that are reasonably good. Of all the hand-design methods, the window method is the most popular and effective. In brief, in the window method we develop a causal linear-phase FIR filter by multipying an ideal filter that has an infinite-duration impulse response (IIR) by a finite-duration window function:

$$h[n] = h_d[n]w[n]$$

where h[n] is the practical FIR filter,$h_d[n]$ is the ideal IIR prototype filter, and w[n] is the finite-duration window function. An important consequence of this operation is that the DTFTs of $h_d[n]$ and w[n]undergo circular convolution in frequency

$$H(e^{j\omega}) = \frac{1}{2\pi}\oint_{2\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta$$

The transfer functions and corresponding impulse responses for the ideal Low pass and High pass filters are as follows:

Lowpass filters:

$$H_d(e^{j\omega}) = \begin{cases} Ge^{-j\omega n_d}, 0 \le |\omega| \le \omega_c \\ 0, \omega_c < |\omega| \le \pi \end{cases}$$

$$h_d[n] = G\frac{\sin(\omega_c(n-n_d))}{\pi(n-n_d)}$$

Highpass filters:

$$H_d(e^{j\omega}) = \begin{cases} Ge^{-j\omega n_d}, \omega_c \le |\omega| \le \pi \\ 0, |\omega| < \omega_c \end{cases}$$

$$h_d[n] = G\left(\delta[n-n_d] - \frac{\sin(\omega_c(n-n_d))}{\pi(n-n_d)}\right)$$

Rectangular window:

The rectangular window is simply obtained by segmenting a finite portion of the impulse response without any shaping in the time domain

$$w[n] = \begin{cases} 1, 0 \le n \le M \\ 0, \text{ otherwise} \end{cases}$$

and the DTFT of the window is given by

$$W(e^{j\omega}) = \frac{\sin\left(\frac{M\omega}{2}\right)}{\sin\left(\frac{\omega}{2}\right)} e^{-j\omega M/2}$$

PROCEDURE:-

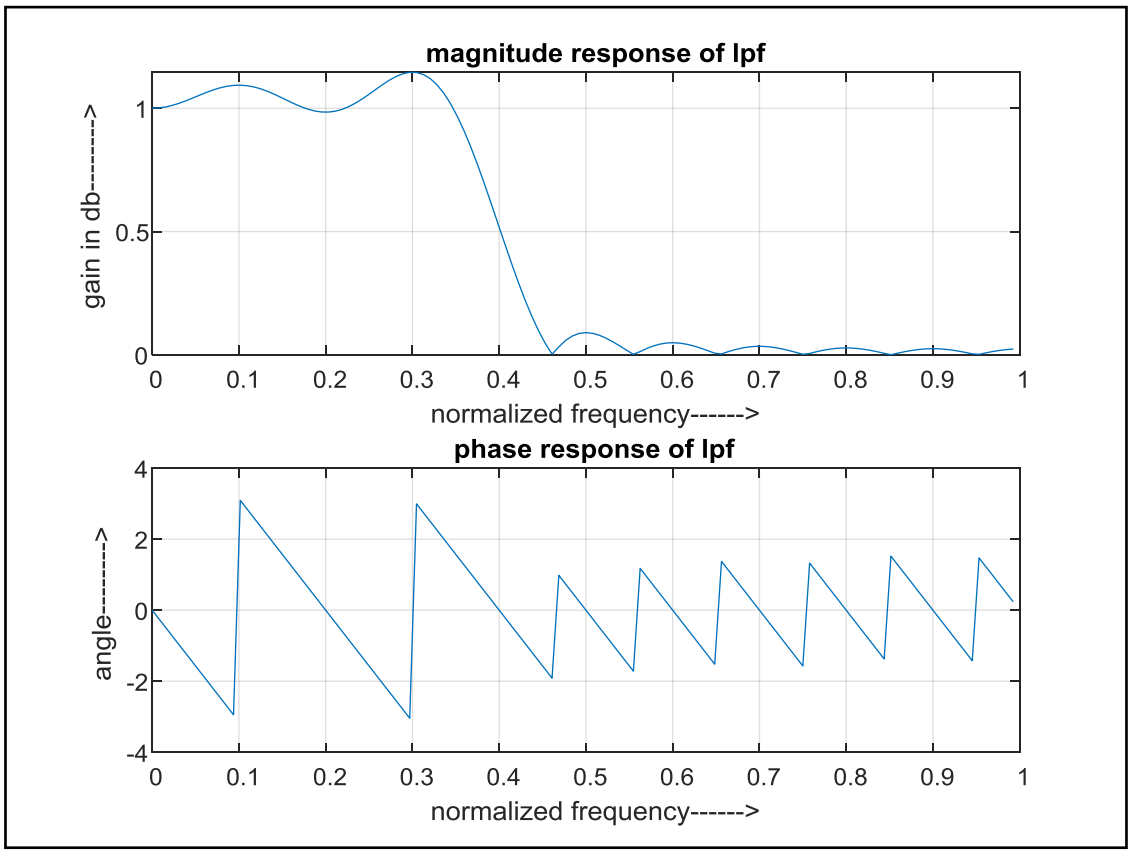
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

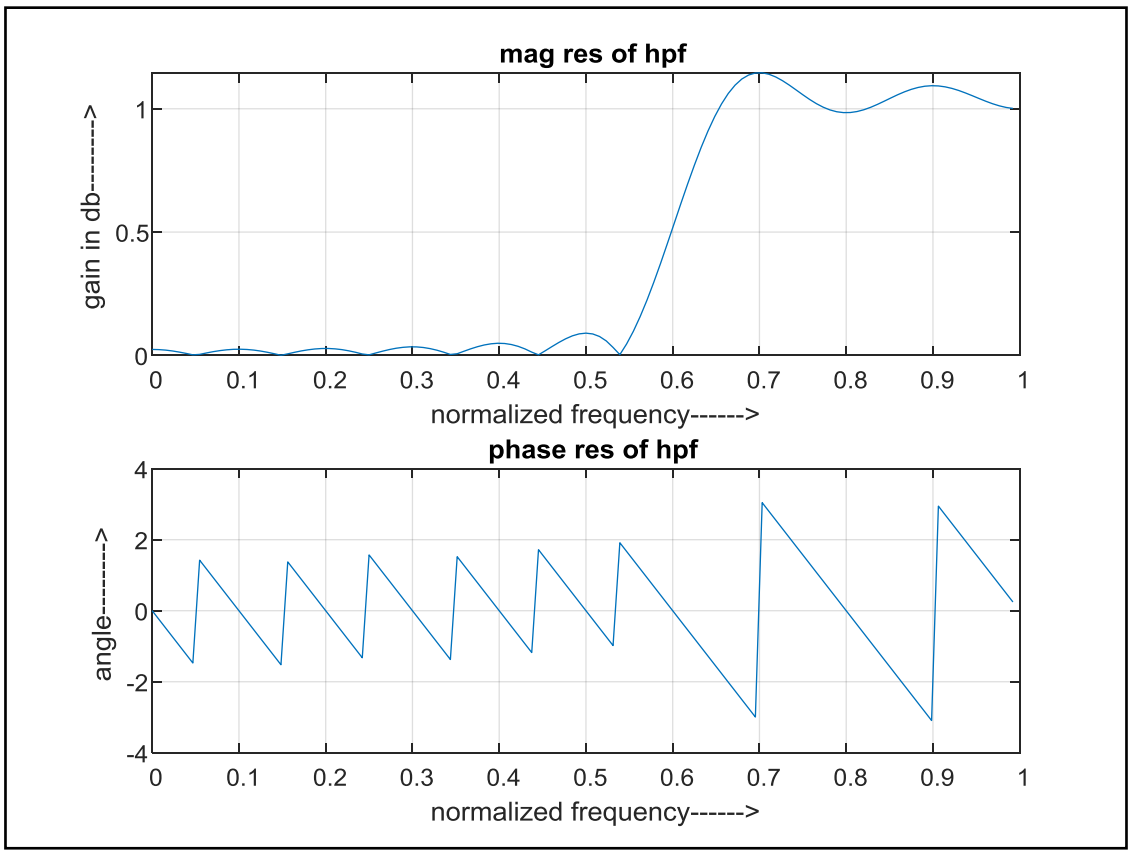
PROGRAM:

```
% To design of FIR filters using rectangular window
techniques.

clc;
clear all;
close all;

% Low Pass Filter
n=20;
fp=200;
fq=300;
fs=1000;
fn=2*fp/fs;
% window=boxcar(n+1);
window=rectwin(n+1);
b=fir1(n,fn,window);
[H W]=freqz(b,1,128);
subplot(2,1,1);
plot(W/pi,abs(H));
title('magnitude response of lpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
grid on
subplot(2,1,2);
```

```
plot(W/pi,angle(H));
title('phase response of lpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
grid on
% Highpass Filter
n=20;
fp=300;
fq=200;
fs=1000;
fn=2*fp/fs;
% window=boxcar(n+1);
window=rectwin(n+1);
b=fir1(n,fn,'high',window);
[H W]=freqz(b,1,128);
figure(2)
subplot(2,1,1);
plot(W/pi,abs(H));
title('mag res of hpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
grid on
subplot(2,1,2);
plot(W/pi,angle(H));
title('phase res of hpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
grid on
```

OUTPUT:





RESULT:

Thus, the MATLAB program for FIR LP/HP using rectangular window Techniques was executed and its frequency response is also verified.

6. FIR FILTER (LP/HP) USING TRIANGULAR WINDOW TECHNIQUE

AIM:

To design a FIR filter using Triangular windowing technique and verify its frequency response.

APPARATUS:

PC with MATLAB

THEORY:

The window method for a causal linear-phase FIR filter is obtained by multipying an ideal filter that has an infinite-duration impulse response (IIR) by a finite-duration window function:

$$h[n] = h_d[n]w[n]$$

where h[n] is the practical FIR filter,$h_d[n]$ is the ideal IIR prototype filter, and w[n] is the finite-duration window function. An important consequence of this operation is that the DTFTs of $h_d[n]$ and w[n] undergo circular convolution in frequency

$$H(e^{j\omega}) = \frac{1}{2\pi}\oint_{2\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta$$

Triangular (Bartlett) window

The Bartlett window is triangularly shaped

$$w[n] = \begin{cases} 1-|(2n/M)-1|, & 0 \le n \le M \\ 0, & \text{otherwise} \end{cases}$$

Because the Bartlett window can be thought of as having been obtained by convolving two rectangular windows of half the width.Its transform is easily obtained by squaring the transform of the rectangular windows:

$$W(e^{j\omega}) = \left(\frac{\sin\left(\frac{M\omega}{4}\right)}{\sin\left(\frac{\omega}{2}\right)}\right)^2 e^{-j\omega M/2}$$

The Bartlett window is having a wider mainlobe than the rectangular window, but more attenuated sidelobes

PROCEDURE:-

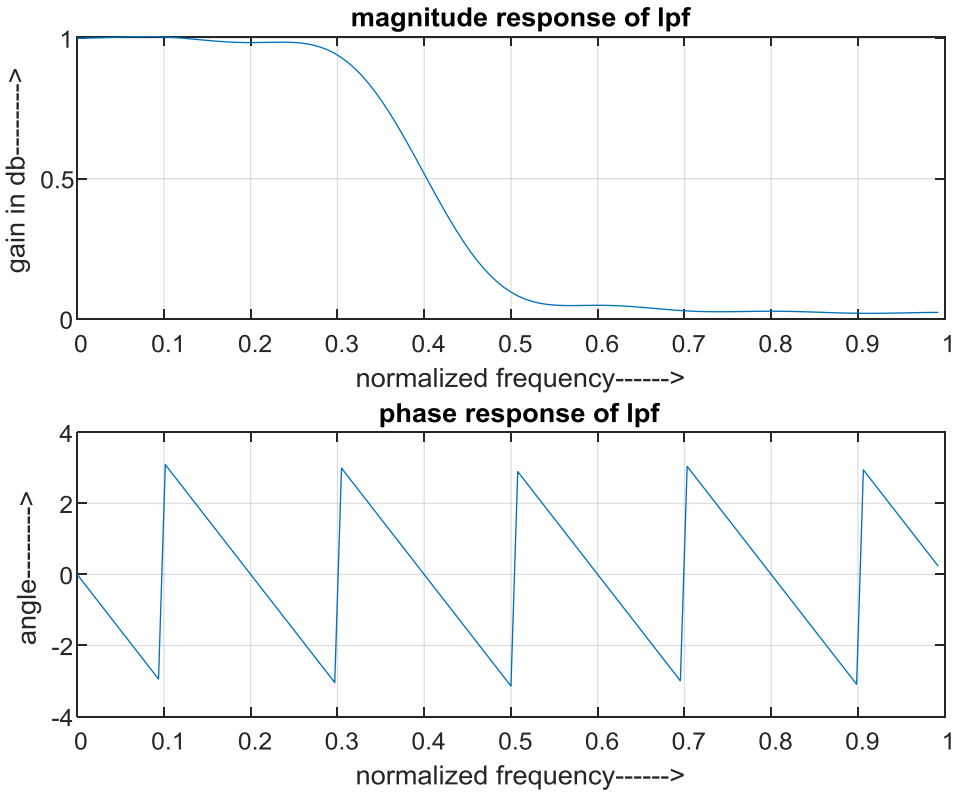
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

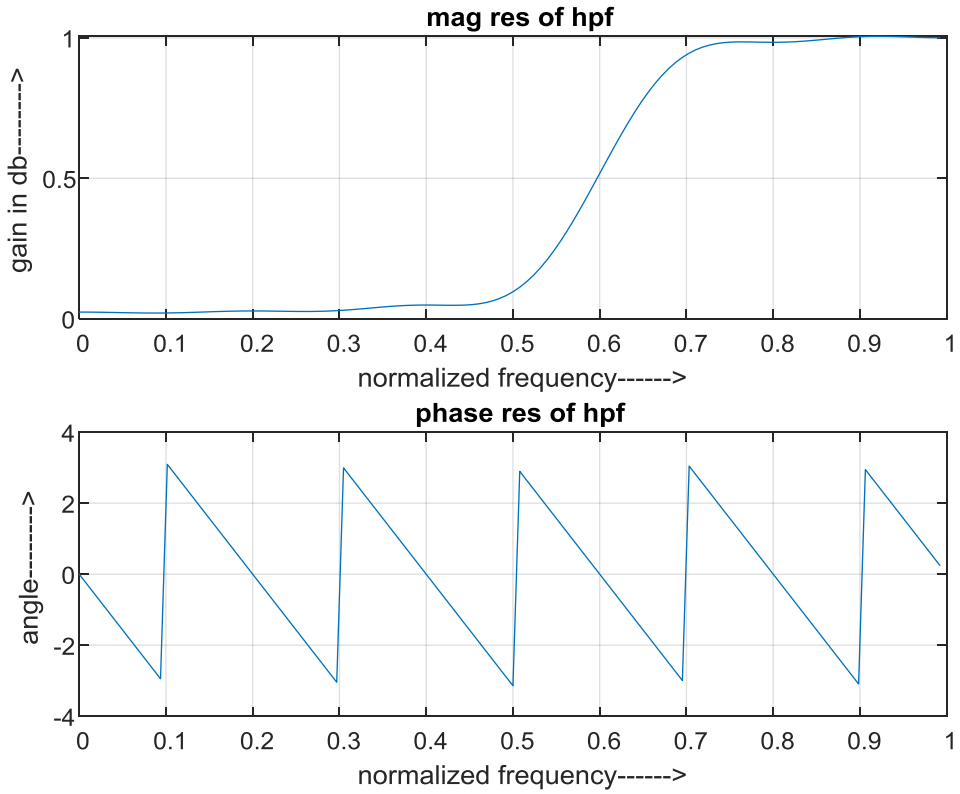
PROGRAM:

```
% To design of FIR filters using triangular window
techniques.

clc;
clear all;
close all;

% Low Pass Filter
n=20;
fp=200;
fq=300;
fs=1000;
fn=2*fp/fs;
window=triang(n+1);
b=fir1(n,fn,window);
[H W]=freqz(b,1,128);
subplot(2,1,1);
plot(W/pi,abs(H));
title('magnitude response of lpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
subplot(2,1,2);
plot(W/pi,angle(H));
title('phase response of lpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');

% Highpass Filter
n=20;
fp=300;
fq=200;
fs=1000;
fn=2*fp/fs;
```

```
window=triang(n+1);
b=fir1(n,fn,'high',window);
[H W]=freqz(b,1,128);
figure(2)
subplot(2,1,1);
plot(W/pi,abs(H));
title('mag res of hpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
subplot(2,1,2);
plot(W/pi,angle(H));
title('phase res of hpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
```

OUTPUT:





RESULT:

Thus the MATLAB program for FIR LP\HP using Triangular window Techniques was executed and its frequency response is also verified.

7. FIR FILTER (LP/HP) USING KAISER WINDOW TECHNIQUE

AIM:

To design a FIR filter using Kaiser windowing technique and verify its frequency response.

APPARATUS:

PC with MATLAB

THEORY:

The window method for a causal linear-phase FIR filter is obtained by multipying an ideal filter that has an infinite-duration impulse response (IIR) by a finite-duration window function:

$$h[n] = h_d[n]w[n]$$

where h[n] is the practical FIR filter,$h_d[n]$ is the ideal IIR prototype filter, and w[n] is the finite-duration window function. An important consequence of this operation is that the DTFTs of $h_d[n]$ and w[n] undergo circular convolution in frequency

$$H(e^{j\omega}) = \frac{1}{2\pi}\oint_{2\pi} H_d(e^{j\theta})W(e^{j(\omega-\theta)})d\theta$$

Kaiser Window:

A very flexible family of window functions has been developed by Kaiser. These windows are nearly optimum in the sense of having the largest energy in the mainlobe for a given peak sidelobe level.

They are closely related to the prolate spheroidal wavefunctions which are the optimum time-limited, continuous-time functions in a similar sense. The Kaiser windows are of the form,

$$w_K(n) = \frac{I_0[\beta\sqrt{1-(1-2n/M)^2}]}{I_0[\beta]}, \qquad n = 0, 1, \ldots, M,$$

where $I_0[\ ]$ is the modified zeroth-order Bessel function of the first kind and ß is a shape parameter determining the tradeoff between the mainlobe width and the peak sidelobe level. Typical values for ß are in the range $4 < ß < 9$. $I_0[\ ]$ is most easily computed from its power series expansion

$$I_0[x] = 1 + \sum_{m=1}^{\infty}\left[\frac{(x/2)^m}{m!}\right]^2,$$

PROCEDURE:-

- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

```
% To design of FIR filters using Kaiser window techniques.

clc;
```

```
clear all;
close all;

% Low Pass Filter
n=100;
fp=200;
fq=300;
fs=1000;
fn=2*fp/fs;
beta = input('Enter beta value ') % use 0.5 to 2.5
window=kaiser((n+1),beta);
b=fir1(n,fn,window);
[H W]=freqz(b,1,128);
subplot(2,1,1);
plot(W/pi,abs(H));
title('magnitude response of lpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
grid on
subplot(2,1,2);
plot(W/pi,angle(H));
title('phase response of lpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
grid on

% Highpass Filter
n=100;
fp=300;
fq=200;
fs=1000;
fn=2*fp/fs;
window=kaiser((n+1),beta);
b=fir1(n,fn,'high',window);
[H W]=freqz(b,1,128);
figure(2)
subplot(2,1,1);
plot(W/pi,abs(H));
title('mag res of hpf');
ylabel('gain in db-------->');
xlabel('normalized frequency------>');
grid on
subplot(2,1,2);
```

```
plot(W/pi,angle(H));
title('phase res of hpf');
ylabel('angle-------->');
xlabel('normalized frequency------>');
grid on
```

INPUT::

enter the passband ripple: 0.02

enter the stopband ripple: 0.01

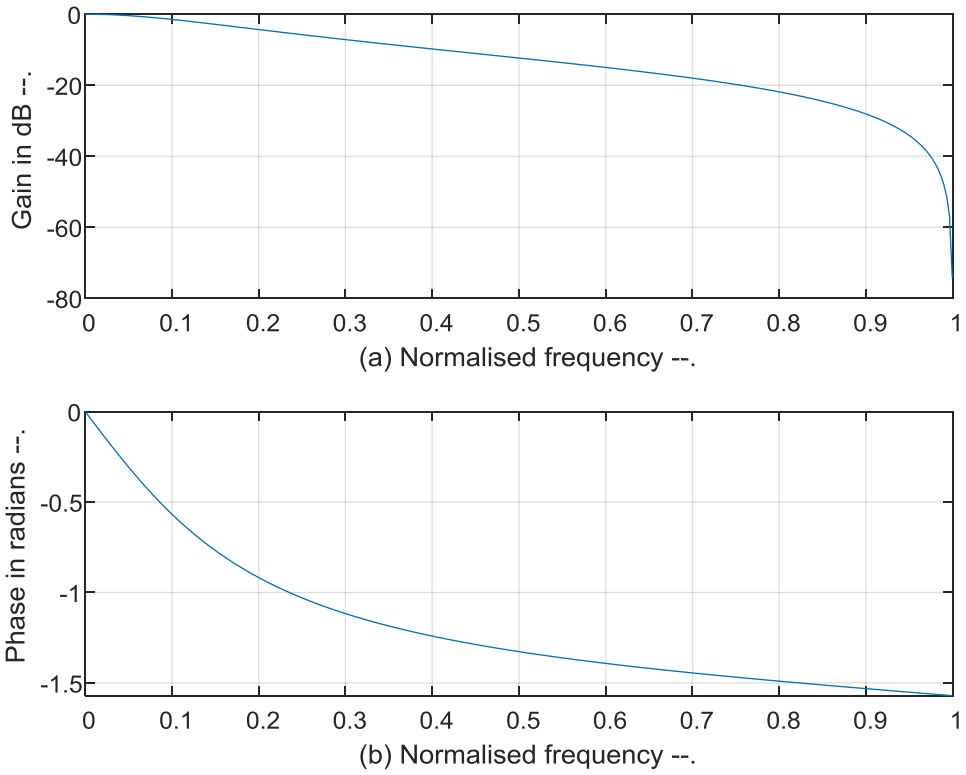
enter the passband frequency: 1000

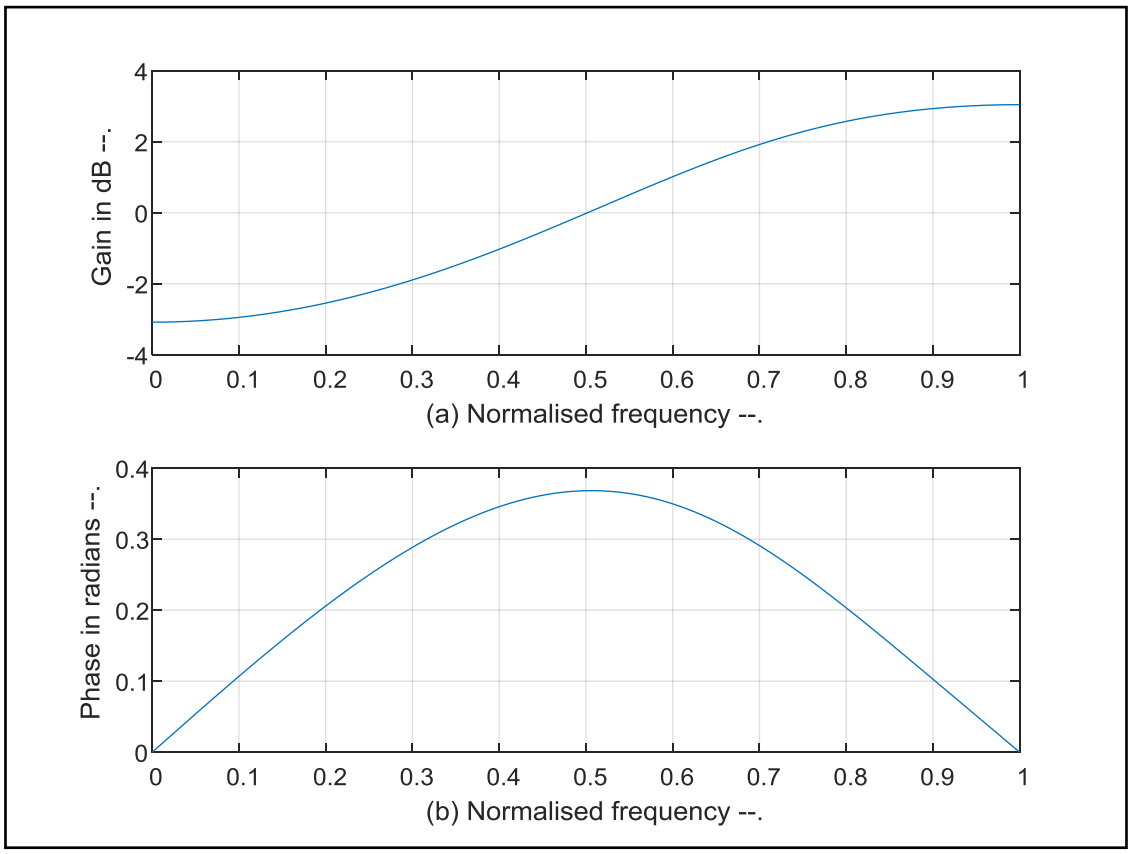
enter the stopband frequency: 1500

enter the sampling frequency: 10000

enter the beta value: 1.2

OUTPUT:





RESULT:

Thus the MATLAB program for FIR LP\HP using Triangular window Techniques was executed and its frequency response is also verified.

8. IIR BUTTERWORTH FILTER IMPLEMENTATION

AIM: Program for Design of Butterworth Analog Low-pass/High-pass Filter.

SOFTWARE: MATLAB

PROGRAM:

```
% To implement LP IIR filter for a given sequence
clc;
close all;
clear all;
format long
rp=input('enter the passband ripple');
rs=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
fs=input('enter the sampling freq');
w1=2*wp/fs;
w2=2*ws/fs;
% LPF
[n,wn]= buttord(w1,w2,rp,rs);
[b,a]= butter(n,wn);
w = 0:.01:pi;
[h,om]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(1)
subplot(2,1,1);plot(om/pi,m);
ylabel('Gain in dB --.');
xlabel('(a) Normalised frequency --.');
grid on
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b) Normalised frequency --.');
ylabel('Phase in radians --.');
grid on

% HPF
[n,wn]=buttord(w1,w2,rp,rs,'s');
[b,a]=butter(n,wn,'high','s');
w=0:.01:pi;
[h,om]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(2)
subplot(2,1,1);
plot(om/pi,m);
ylabel('Gain in dB --.');
```

```
xlabel('(a) Normalised frequency --.');
grid on
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b) Normalised frequency --.');
ylabel('Phase in radians --.');
grid on
```

INPUT::
enter the passband ripple: 10

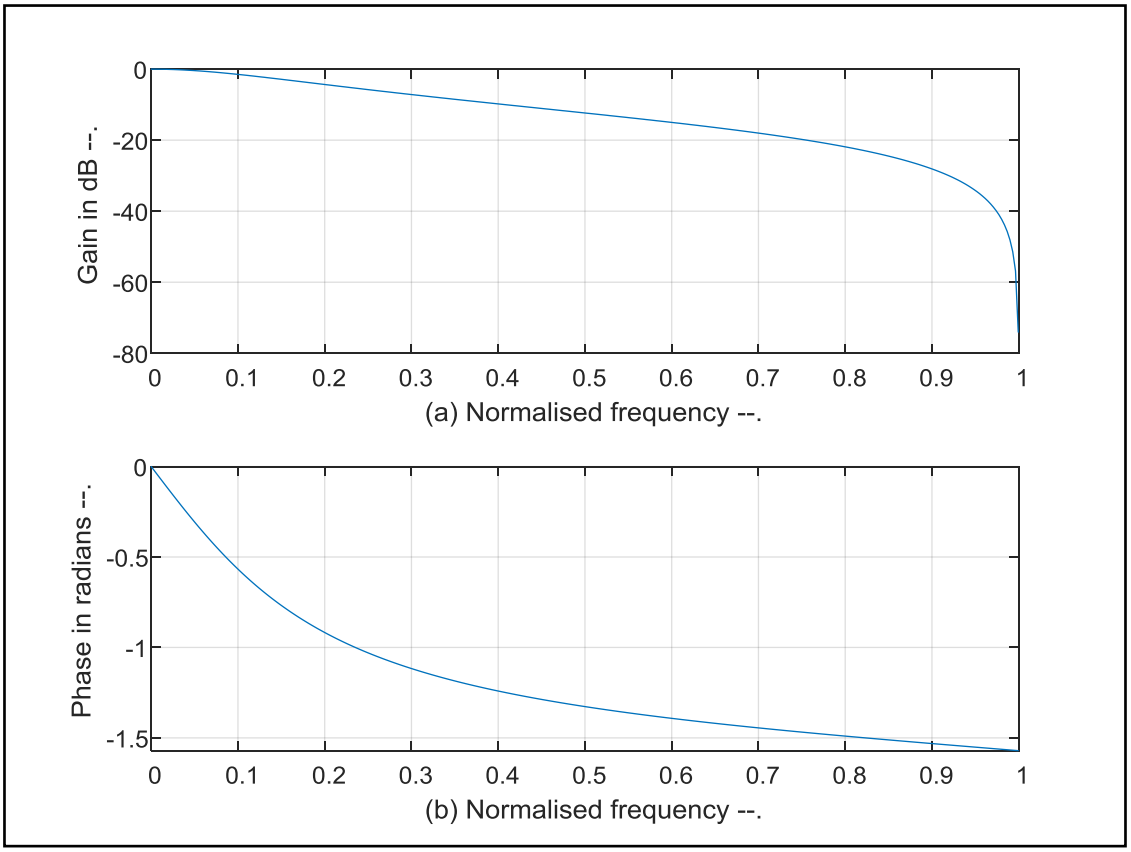
enter the stopband ripple: 15

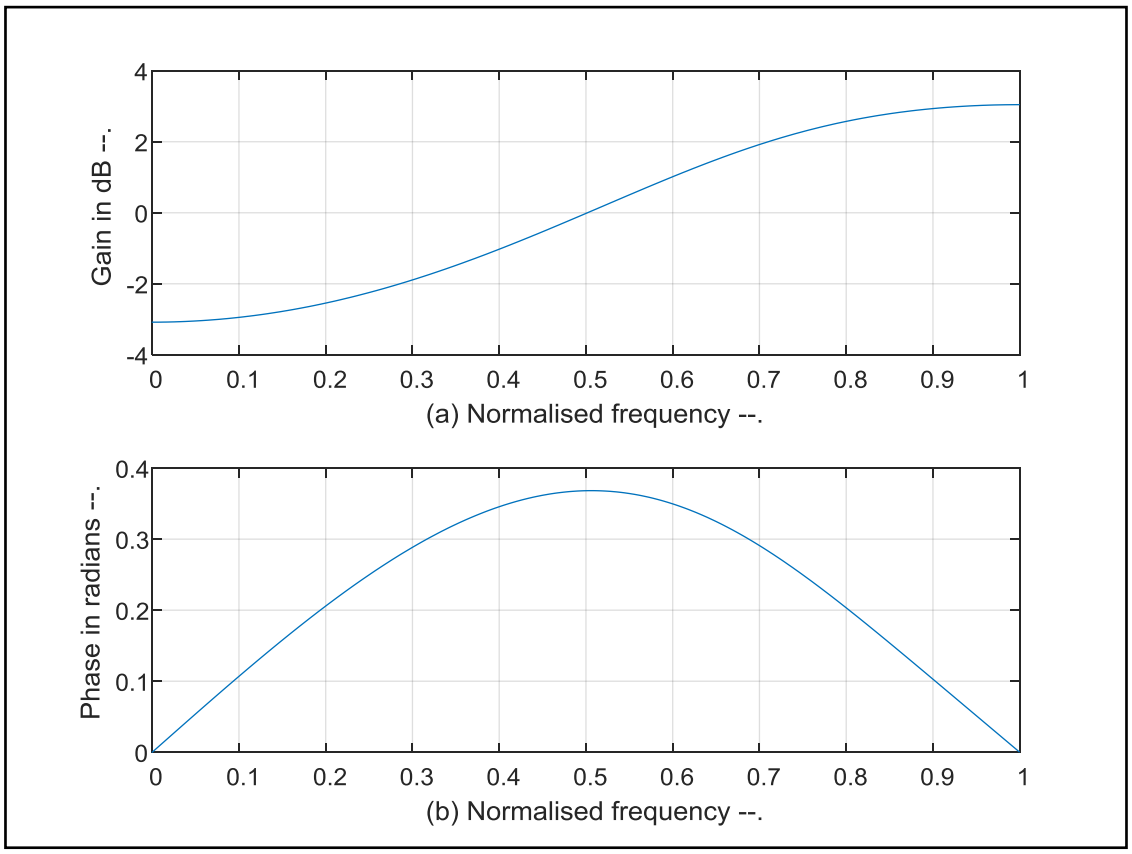
enter the passband frequency: 1000

enter the stopband frequency: 1500

enter the sampling frequency: 5000

OUTPUT:





RESULT:

Thus the MATLAB program for IIR LP\HP using butterworth filter and its frequency response is also verified.

9. IIR CHEBYSHEV FILTER IMPLEMENTATION

AIM: Program for Design of Chebyshev Analog Low-pass/High-pass Filter.

SOFTWARE: MATLAB

PROGRAM:

```
% To implement LP IIR filter for a given sequence
clc;
close all;
clear all;
format long
rp=input('enter the passband ripple');
rs=input('enter the stopband ripple');
wp=input('enter the passband freq');
ws=input('enter the stopband freq');
fs=input('enter the sampling freq');
w1=2*wp/fs;
w2=2*ws/fs;
% LPF
[n,wn]= cheb1ord (w1,w2,rp,rs);
[b,a]= cheby1(n,wn);
w = 0:.01:pi;
[h,om]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(1)
subplot(2,1,1);plot(om/pi,m);
ylabel('Gain in dB --.');
xlabel('(a) Normalised frequency --.');
grid on
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b) Normalised frequency --.');
ylabel('Phase in radians --.');
grid on

% HPF
[n,wn]=cheb1ord(w1,w2,rp,rs,'s');
[b,a]=cheby1(n,wn,'high','s');
w=0:.01:pi;
[h,om]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(2)
subplot(2,1,1);
plot(om/pi,m);
ylabel('Gain in dB --.');
```

```
xlabel('(a) Normalised frequency --.');
grid on
subplot(2,1,2);
plot(om/pi,an);
xlabel('(b) Normalised frequency --.');
ylabel('Phase in radians --.');
grid on
```

INPUT::
enter the passband ripple: 10

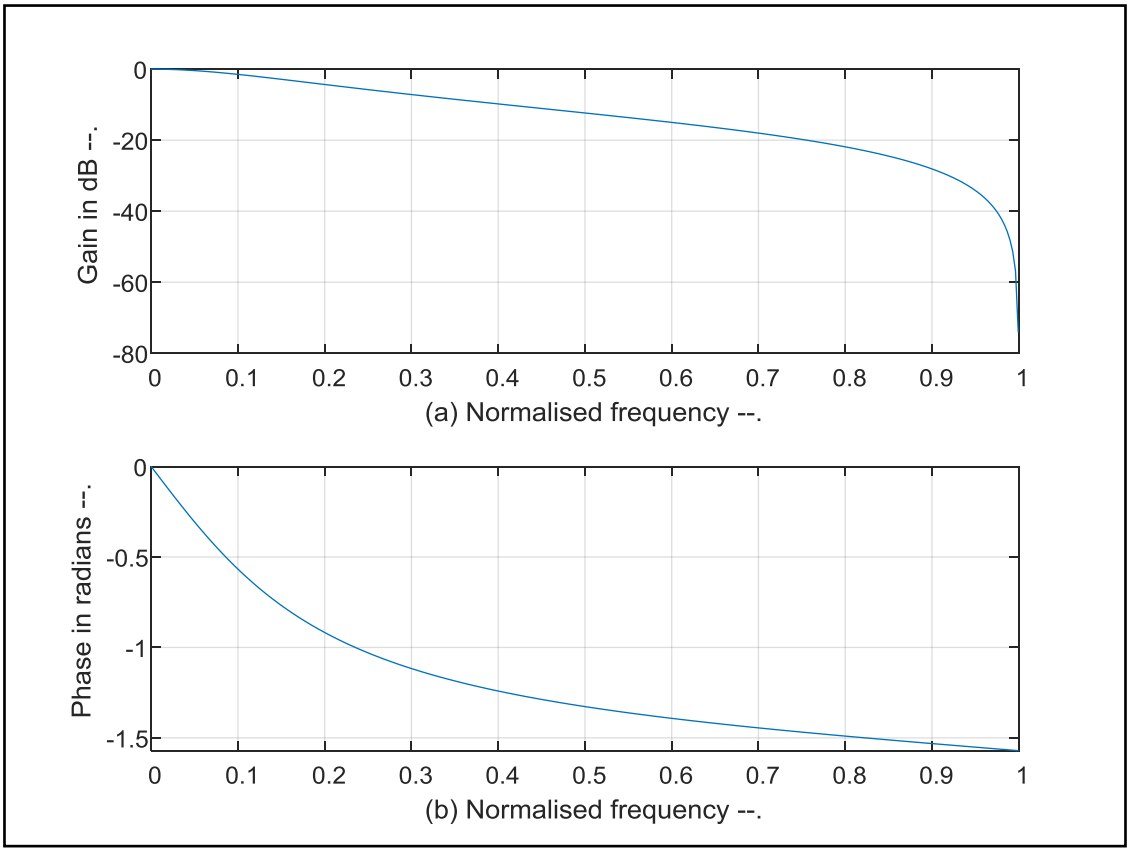
enter the stopband ripple: 15

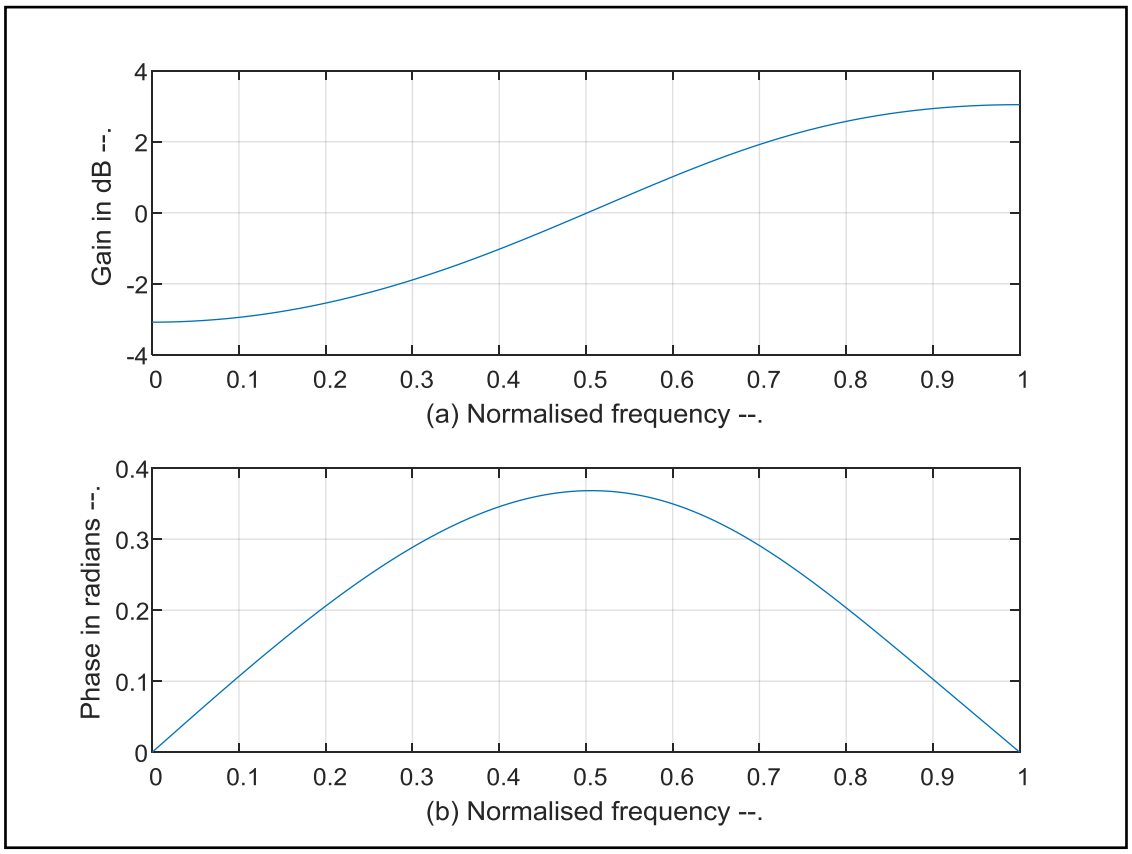
enter the passband frequency: 1000

enter the stopband frequency: 1500

enter the sampling frequency: 5000

OUTPUT:





RESULT:

Thus the MATLAB program for IIR LP\HP using butterworth filter and its frequency response is also verified.

10. UPSAMPLING A SINUSOIDAL SIGNAL

AIM:

To generate the up sample (interpolation) by an integer factor

APPARATUS:

PC with MATLAB

PROGRAM:

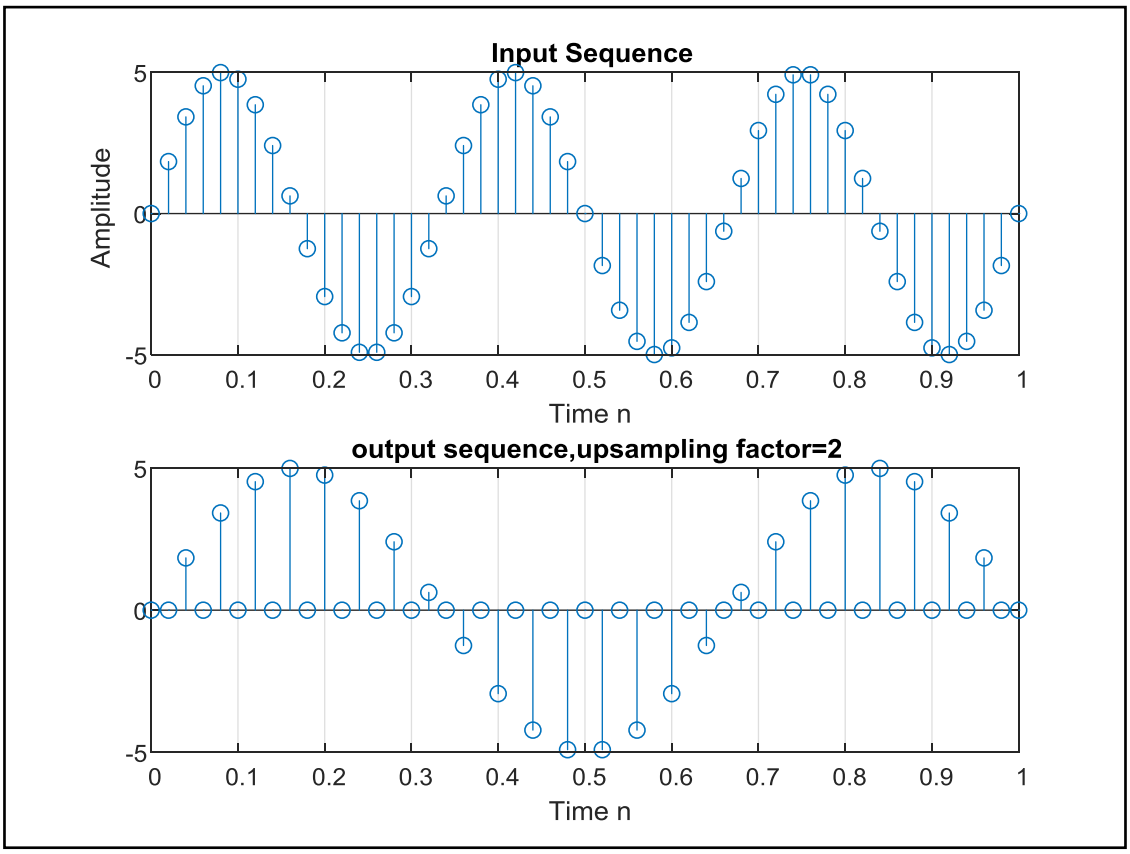
```
% Program for upsampling a sinusoidal signal by factor L
N=input('Input length of the sinusoidal sequence=');
L=input('Up Samping factor=');%take min 1000
fi=input('Input signal frequency=');
intv = 1/N;
% Generate the sinusoidal sequence for the specified length N
n=0:intv:1; % range of time
x=5*sin(2*pi*fi*n);
% Generate the upsampled signal
y=zeros (1,L*length(x));
y([1:L:length(y)])=x;
%Plot the input sequence
subplot (2,1,1);
stem (n,x);
title('Input Sequence');
xlabel('Time n');
ylabel('Amplitude');
%Plot the output sequence
subplot (2,1,2);
stem (n,y(1:length(x)));
title(['output sequence,upsampling factor=',num2str(L)]);
xlabel('Time n');
ylabel('Amplitude');
```

INPUT::
Input length of the sinusoidal sequence=50

Up Samping factor=2

Input signal frequency=3

OUTPUT

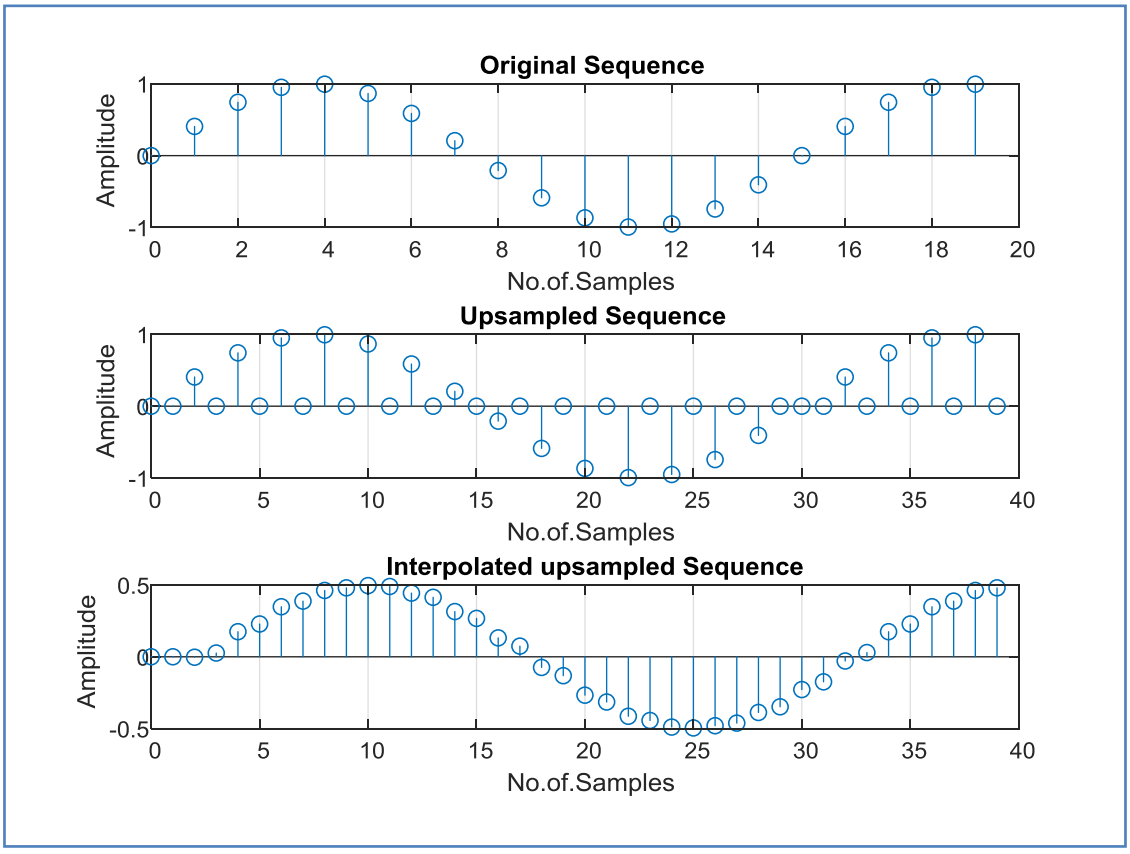


PROGRAM:

```
% Interpolation of the Signal
clc;
clear all;
close all;
N=20;
n=0:1:N-1;
x=sin(2*pi*n/15);
L=2;
figure(1)
subplot(3,1,1)
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(1)
subplot(3,1,2)
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(1)
subplot(3,1,3)
stem(n1-1,y);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Interpolated upsampled Sequence');
```

OUTPUT



11. DOWNSAMPLING A SINUSOIDAL SIGNAL

AIM:

To generate the down sample (decimation) by an integer factor

APPARATUS:

PC with MATLAB

PROGRAM:

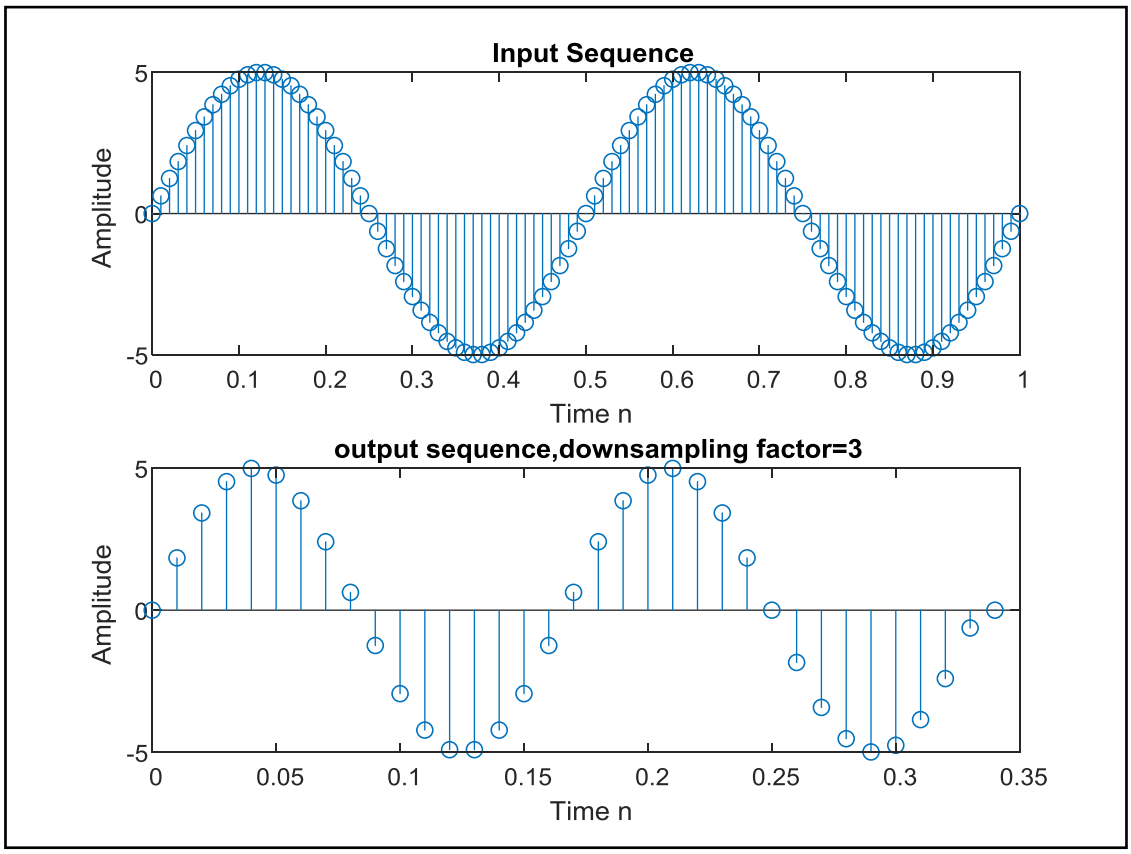
```
% Program for downsampling a sinusoidal signal by factor M
clc; clear all; close all;
N=input('Input length of the sinusoidal sequence=');
M=input('Down Samping factor=');%take min 1000
fi=input('Input signal frequency=');
intv = 1/N;
% Generate the sinusoidal sequence for the specified length N
m=0:intv:1; % range of time
x=5*sin(2*pi*fi*m);
%Plot the input sequence
subplot (2,1,1);
stem (m,x);
title('Input Sequence');
xlabel('Time n');
ylabel('Amplitude');
% Generate the upsampled signal
myrem = rem(length(x),M);
x = [x zeros(1,myrem)];
y = x(1:M:length(x));
%Plot the output sequence
subplot (2,1,2);
stem (m(1:length(y)),y);
title(['output sequence,downsampling factor=',num2str(M)]);
xlabel('Time n');
ylabel('Amplitude');
```

INPUT::
Input length of the sinusoidal sequence=100

Down Samping factor=3

Input signal frequency=2

OUTPUT

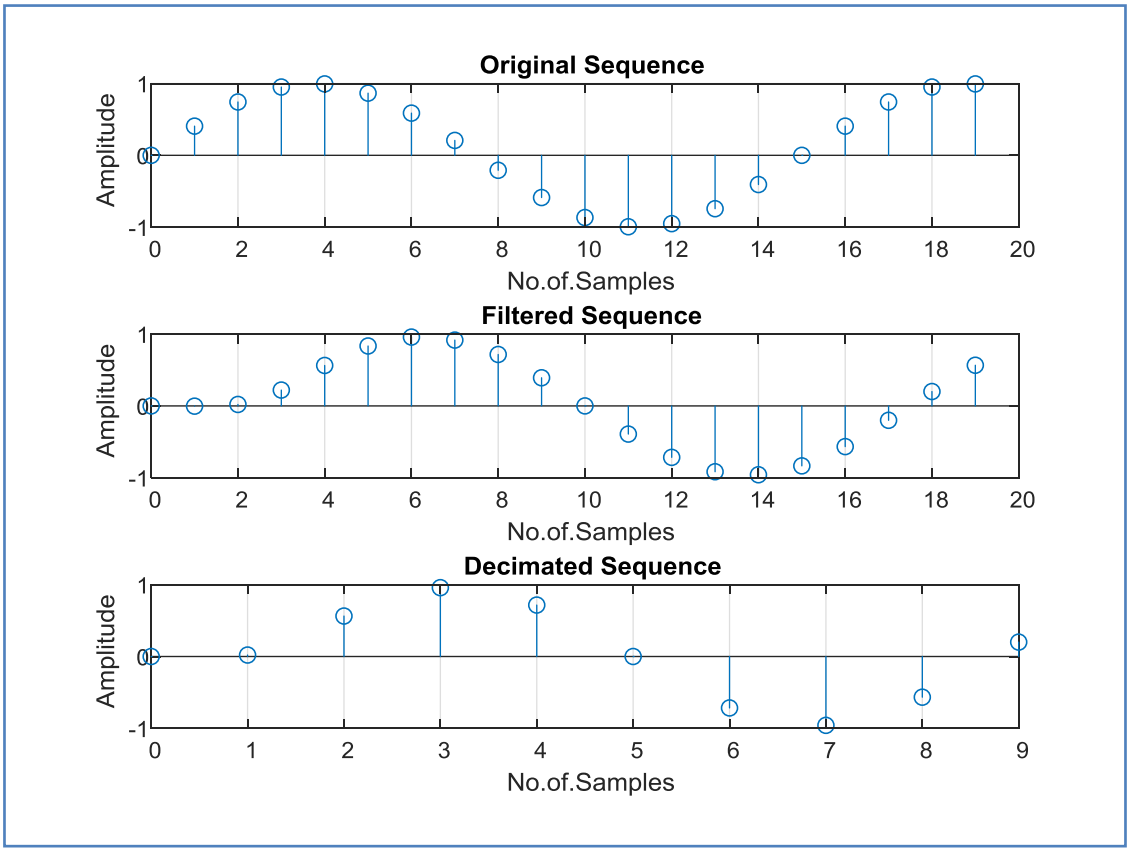


PROGRAM:

```
%  To generate the down sample (decimation) by an Integer
factor
clc;
clear all;
close all;
N=20 ;
n=0:1:N-1;
x=sin(2*pi*n/15);
M=2;
figure(1)
subplot(3,1,1)
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(1)
subplot(3,1,2)
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(1)
subplot(3,1,3)
stem(n(1:length(x1)),x1);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Decimated Sequence');
```

OUTPUT



ADDITIONAL EXPERIMENTS

1. POWER SPECTRAL DENSITY ESTIMATION

AIM:

To calculate the power spectral density of a signal and plot the power distribution of the signal versus frequency graph

APPARATUS:

PC with MATLAB

THEORY:

The discrete Fourier transform (DFT) maps a finite number of discrete time-domain samples to the same number of discrete Fourier-domain samples. Being practical to compute, it is the primary transform applied to real-world sampled data in digital signal processing. The DFT has special relationships with the discrete-time Fourier transform and the continuous-time Fourier transform that let it be used as a practical approximation of them through truncation and windowing of an infinite-length signal. Different window functions make various tradeoffs in the spectral distortions and artifacts introduced by DFT-based spectrum analysis.

The DFT transforms *N* samples of a discrete-time signal to the same number of discrete frequency samples, and is defined as

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{i2\pi nk}{N}}$$

The DFT is invertible by the inverse discrete Fourier transform (IDFT):

$$x(n) = \frac{1}{\mathrm{N}} \sum_{k=0}^{N-1} X(k) e^{i\frac{2\pi nk}{N}}$$

The DFT and IDFT are a self-contained, one-to-one transform pair for a length-*N* discrete-time signal. The DFT is notmerely **a**n approximation to the DTFT. However, the DFT is very often used as a practical approximation to the DTFT.

PROCEDURE:-

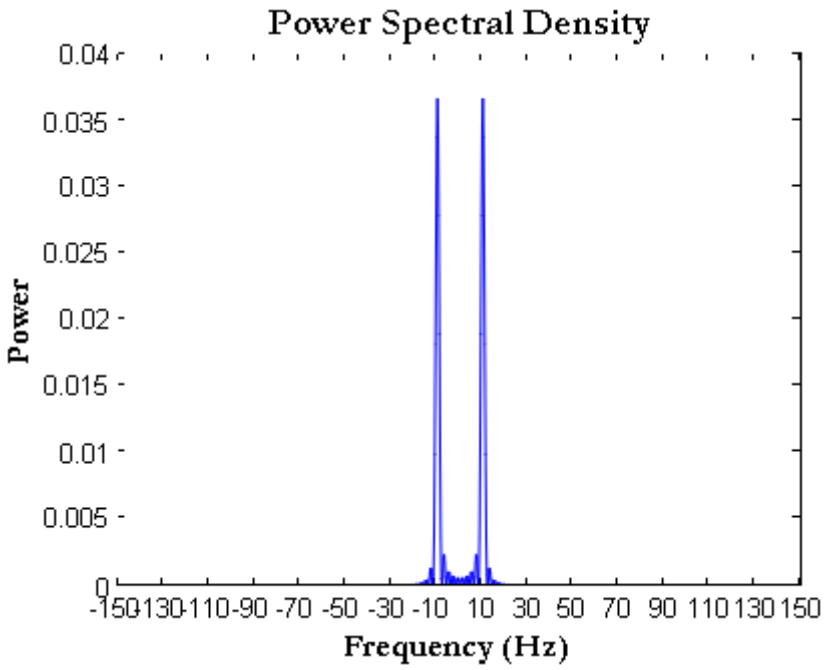
- Open MATLAB
- Open new M-file
- Type the program
- Save in current directory
- Compile and Run the program
- For the output see command window\ Figure window

PROGRAM:

```
clc;
```

```
closeall;
clearall;
Fs = 1000;
t = 0:1/Fs:1-1/Fs;
x = cos(2*pi*100*t) + randn(size(t));
N = length(x);
xdft = fft(x);
xdft = xdft(1:N/2+1);
psdx = (1/(Fs*N)) * abs(xdft).^2;
psdx(2:end-1) = 2*psdx(2:end-1);
freq = 0:Fs/length(x):Fs/2;
plot(freq,10*log10(psdx))
gridon
title('Power Spectral Density')
xlabel('Frequency (Hz)')
ylabel('Power (dB)')
```

OUTPUT:



RESULT:

DFT Spectral analysis on a continuous time signal was performed and the Power density spectral graph with respect to frequency was plotted