Lab manual for

Digital Signal Processing Lab

III B. Tech II Semester

Prepared by

J. Sunil Kumar, P.Saritha
&
Ch.Sateesh kumar reddy

Department of Electronics & Communication Engineering

Turbomachinery Institute of Technology & Sciences

(Approved by AICTE & Affiliated to JNTUH)

Indresam(v), Patancheru(M), Medak(Dist). Pin: 502 319

Turbomachinery Institute of Technology & Sciences

<u>*Certificate*</u>

This is to certify that Mr. / Ms. ………………………………….. RollNo……………..… of I/II/III/IV B.Tech I / II Semester of ………….……………………..…………branch has completed the laboratory work satisfactorily in ……………….……….……..... Lab for the academic year 20 … to 20 …as prescribed in the curriculum.

Place: ……………….….

Date: ……………..….

Lab In charge **Head of the Department** **Principal**

.

LIST OF EXPERIMENTS:

1. Generation of Sinusoidal Waveform / Signal
2. To find the DFT / IDFT of given DT Signal
3. To find the frequency response of a given system in transfer function
4. Implementation of FFT of given sequence
5. Implementation of LP FIR filter for given sequence
6. Implementation of HP FIR filter for given sequence
7. Implementation of LP IIR filter for given sequence
8. Implementation of HP IIR filter for given sequence
9. Determination of power spectrum of a given sequence
10. Generation of DTMF Signals
11. Implementation of Decimation Process
12. Implementation of Interpolation Process

1. INTRODUCTION TO MATLAB.

GENERATION OF BASIC SIGNALS USING MATLAB

AIM: Generation of Sine Wave & Illustration of the Sampling Process in the Time Domain

EQUIPMENT REQUIRED:	P – IV Computer
	Windows Xp SP2
	MATLAB 7.0

THEORY:

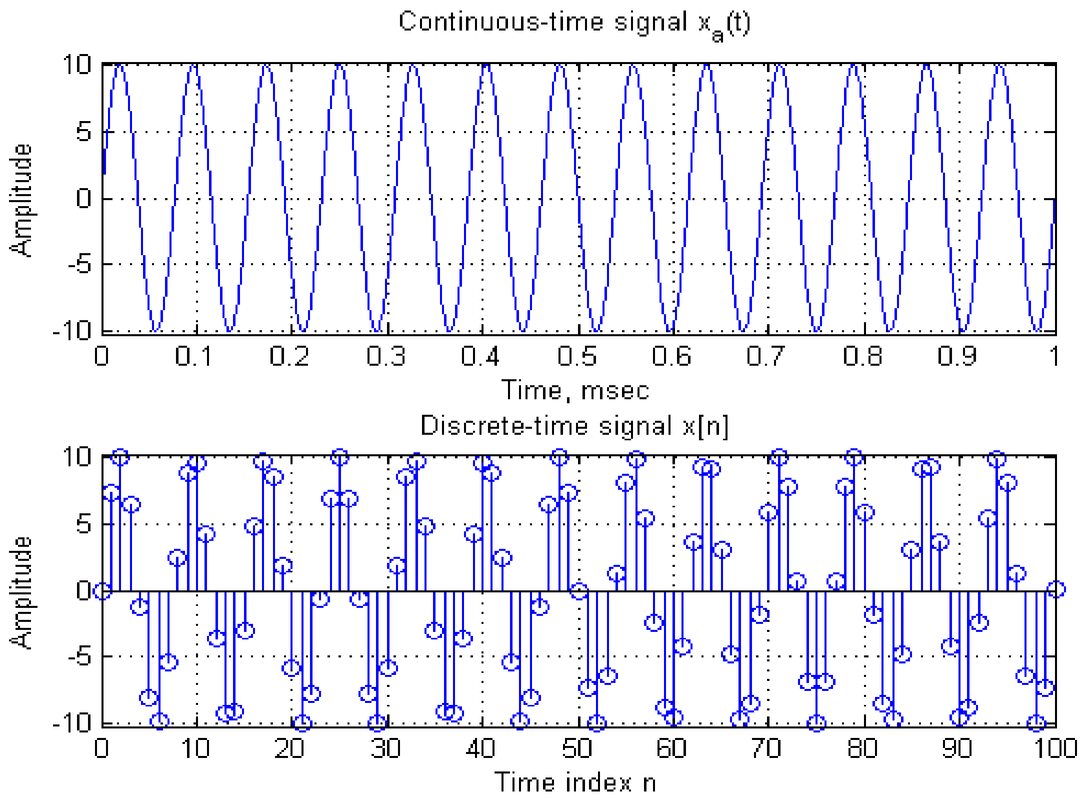
PROGRAM :

% 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:



RESULT_

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*∏*f*t)+exp(-8*∏*f*)

   Y=((exp(-1.56∏f)*Sin(2∏f)+cos(2∏f)

2. DFT/IDFT of a sequence without using the inbuilt functions

AIM: To find the DFT/IDFT of a sequence without using the inbuilt functions

EQUIPMENT REQUIRED:	P – IV Computer
	Windows Xp SP2
	MATLAB 7.0

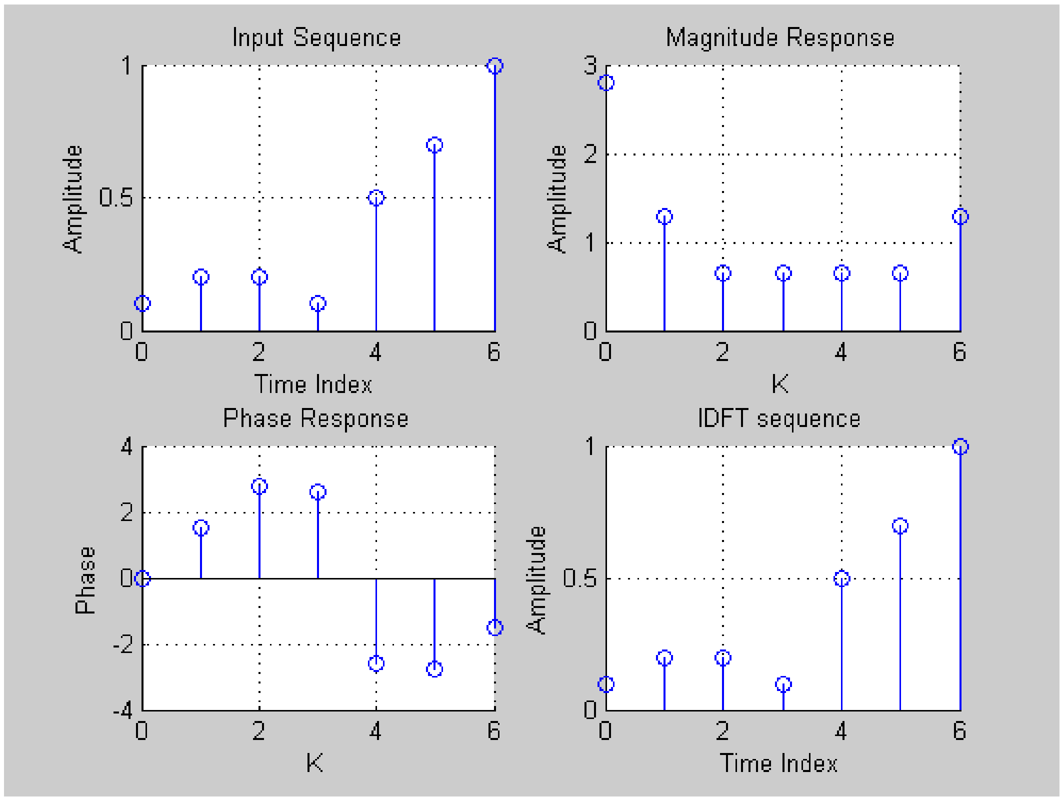
THEORY:

PROGRAM:

```
%program to find the DFT/IDFT of a sequence without using the inbuilt functions
clc
close all;
clear all;
xn=input('Enter the sequence x(n)');  %Get the sequence from user
ln=length(xn);            %find the length of the sequence
xk=zeros(1,ln);           %initilise an array of same size as that of input sequence
ixk=zeros(1,ln);           %initilise an array of same size as that of input sequence
%code block to find the DFT of the sequence
%-----------------------------------------------------------
for k=0:ln-1
   for n=0:ln-1
       xk(k+1)=xk(k+1)+(xn(n+1)*exp((-i)*2*pi*k*n/ln));
   end
end
%------------------------------------------------------------
%code block to plot the input sequence
%------------------------------------------------------------
t=0:ln-1;
subplot(221);
stem(t,xn);
grid
ylabel ('Amplitude');
xlabel ('Time Index');
title('Input Sequence');
%---------------------------------------------------------------
magnitude=abs(xk); % Find the magnitudes of individual DFT points
%code block to plot the magnitude response
%------------------------------------------------------------
t=0:ln-1;
subplot(222);
stem(t,magnitude);
grid
ylabel ('Amplitude');
xlabel ('K');
title ('Magnitude Response');
%------------------------------------------------------------
phase=angle(xk); % Find the phases of individual DFT points
%code block to plot the magnitude sequence
%------------------------------------------------------------
t=0:ln-1;
subplot(223);
stem(t,phase);
grid
ylabel ('Phase');
xlabel ('K');
title('Phase Response');
%------------------------------------------------------------
% Code block to find the IDFT of the sequence
%------------------------------------------------------------
for n=0:ln-1
   for k=0:ln-1
```

```
        ixk(n+1)=ixk(n+1)+(xk(k+1)*exp(i*2*pi*k*n/ln));
    end
end
ixk=ixk./ln;
%-----------------------------------------------------------
%code block to plot the input sequence
%-----------------------------------------------------------
t=0:ln-1;
subplot(224);
stem(t,xn);
grid;
ylabel ('Amplitude');
xlabel ('Time Index');
title ('IDFT sequence');
```

Expected Output Waveform:



RESULT:

VIVA QUESTIONS:

1. How many additions and multiplications are needed in DFT?
2. What is the relation between Fourier transform and Z-transform?.

3. Study Frequency Response Of Second Order System

AIM: To study frequency response of second order system using MATLAB & Code Composer Studio3.1.

EQUIPMENT REQUIRED: P – IV Computer

Windows Xp SP2

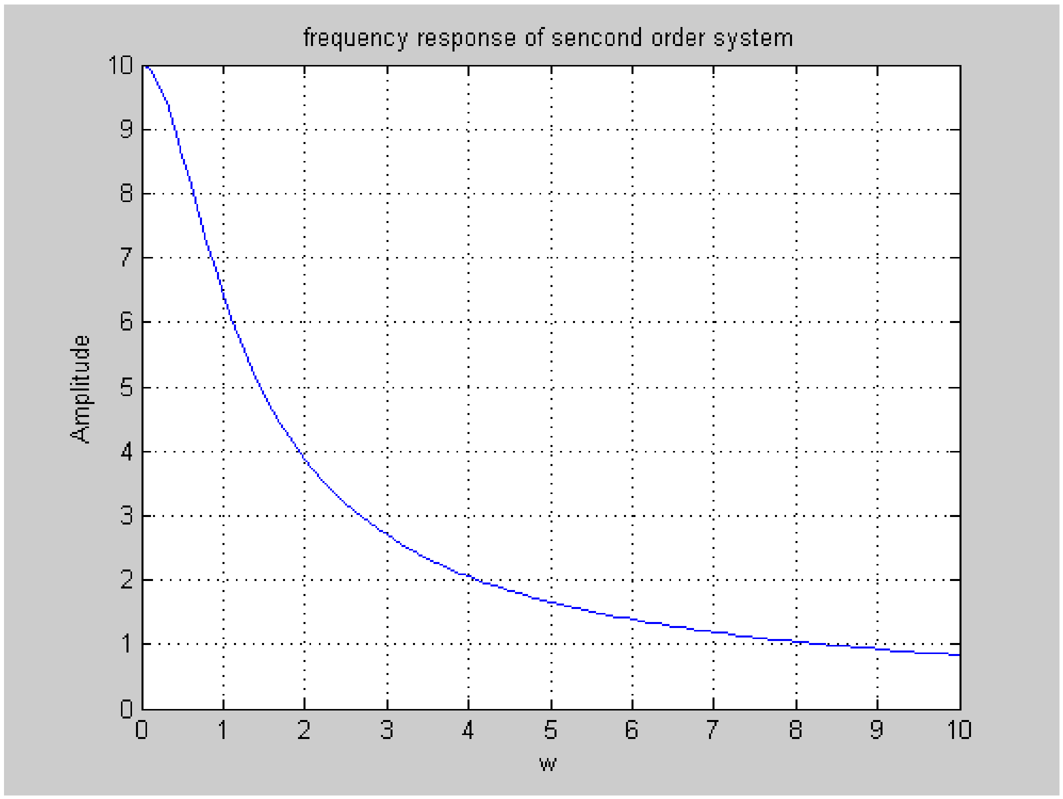
MATLAB 7.0

THEORY:

PROGRAM:

```
clc
close all
x=1
num=[10];
dem=[0.01 1.2 1.0];
c=tf(num,dem);     %Specify transfer functions
w=0:0.1:10;
H=freqresp(c,w);
for i=1:1:101
    p(i)=abs(H(:,:,i))
end
plot(w,p)
grid;
title('frequency response of sencond order system')
xlabel('w')
ylabel('Amplitude')
```

OUTPUT WAVE FORM:



RESULT:

4. IMPLEMENTATION OF FFT OF GIVEN SEQUENCE

AIM: Implementation of FFT of given sequence

EQUIPMENT REQUIRED:	P – IV Computer
	Windows Xp SP2
	MATLAB 7.0

THEORY:

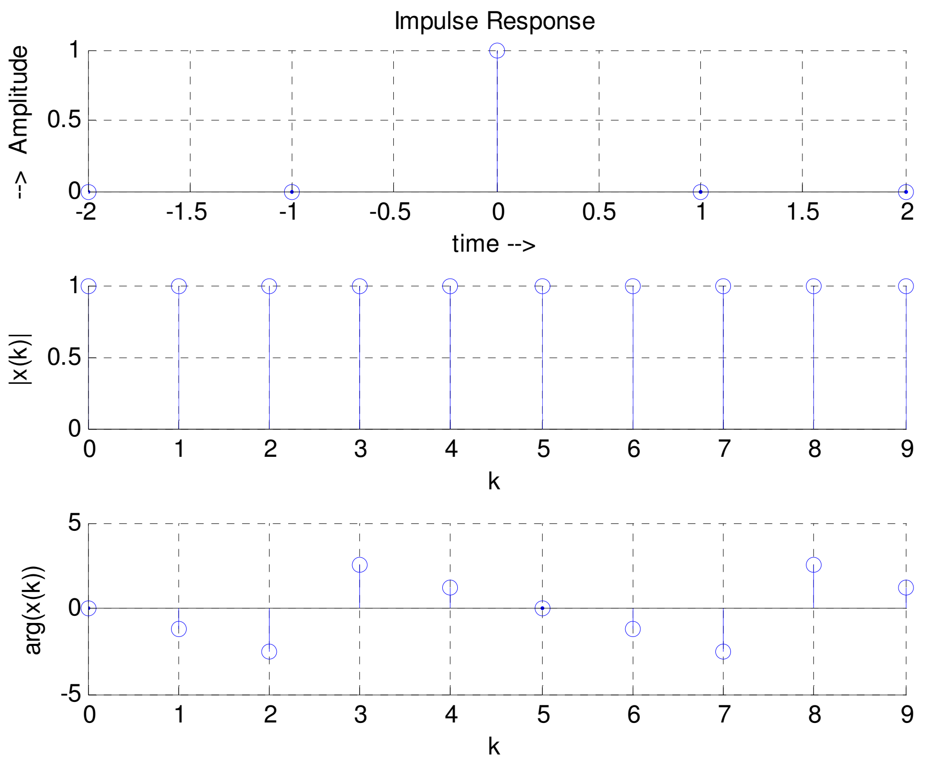
PROGRAM

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

clc;
clear all;
close all;
  %impulse sequence
  t=-2:1:2;
  y=[zeros(1,2) 1 zeros(1,2)];
  subplot (3,1,1);
  stem(t,y);
  grid;
  input('y=');
  disp(y);
  title ('Impulse Response');
  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:

```
y=
   0    0    1    0    0


enter the length of the FFT sequence: 10

 1.0000          0.3090 - 0.9511i      -0.8090 - 0.5878i       -0.8090 + 0.5878i   0.3090 + 0.9511i

 1.0000          0.3090 - 0.9511i       -0.8090 - 0.5878i       -0.8090 + 0.5878i   0.3090 + 0.9511i
```

EXPECTED WAVEFORMS



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

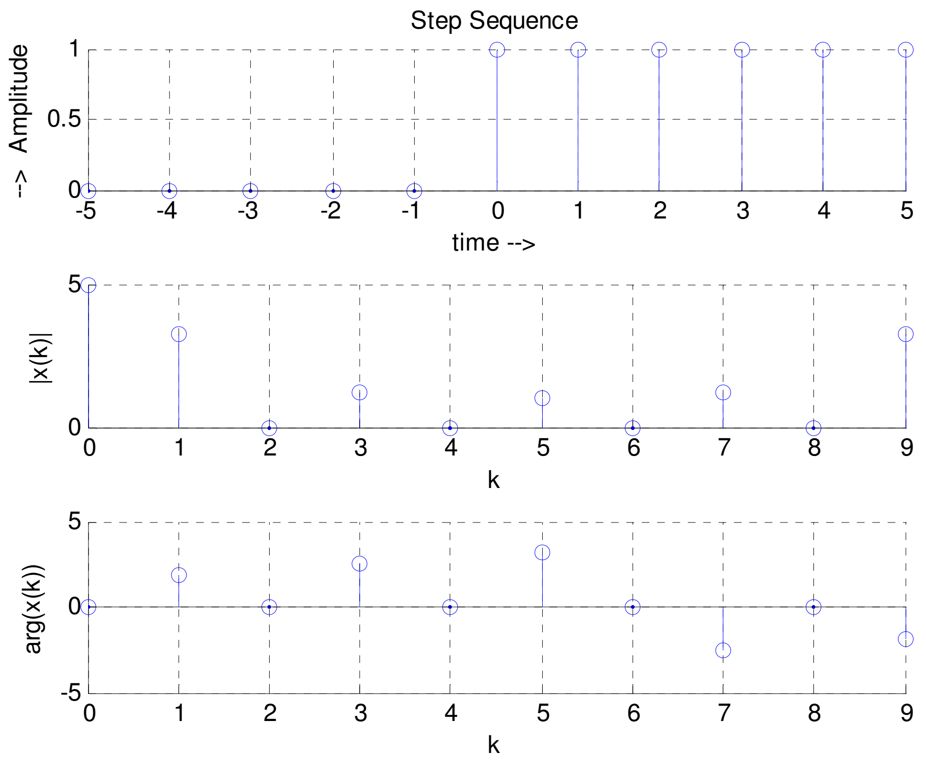
```
clc;
clear all;
close all;
   %Step Sequence
   s=input ('enter the length of step sequence');
   t=-s:1:s;
   y=[zeros(1,s) ones(1,1) ones(1,s)];
   subplot(3,1,1);
   stem(t,y);
   grid
   input('y=');
   disp(y);
   title ('Step Sequence');
   xlabel ('time -->');
   ylabel ('-->  Amplitude');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

EXPECTED WAVEFORMS



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

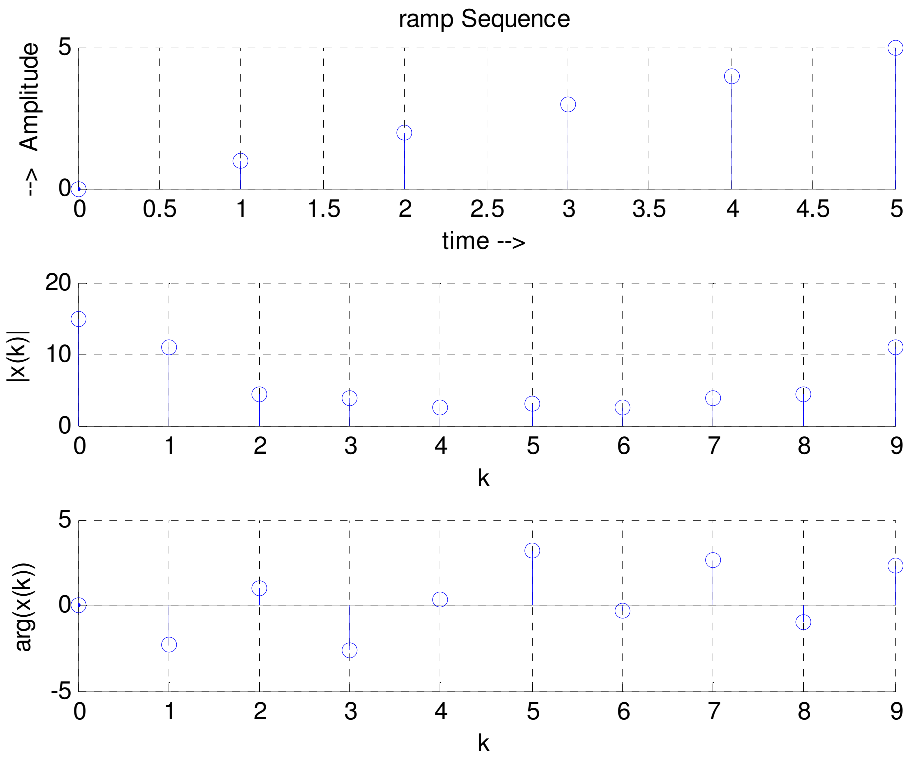
```
clc;
clear all;
close all;
    %Ramp Sequence
    s=input ('enter the length of Ramp sequence: ');
    t=0:s;
    y=t
    subplot(3,1,1);
    stem(t,y);
```

```
grid
input('y=');
disp(y);
title ('ramp Sequence');
xlabel ('time -->');
ylabel ('--> Amplitude');
xn=y;
N=input('enter the legth 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))');
```

OUTPUT: enter the length of Ramp sequence: 5

```
y =     0     1     2     3     4     5

enter the length of the FFT sequence: 10
 15.0000          -7.7361 - 7.6942i      2.5000 + 3.4410i       -3.2639 - 1.8164i       2.5000 + 0.8123i
-3.0000           2.5000 - 0.8123i       -3.2639 + 1.8164i     2.5000 - 3.4410i        -7.7361 + 7.6942i
```

EXPECTED WAVEFORMS



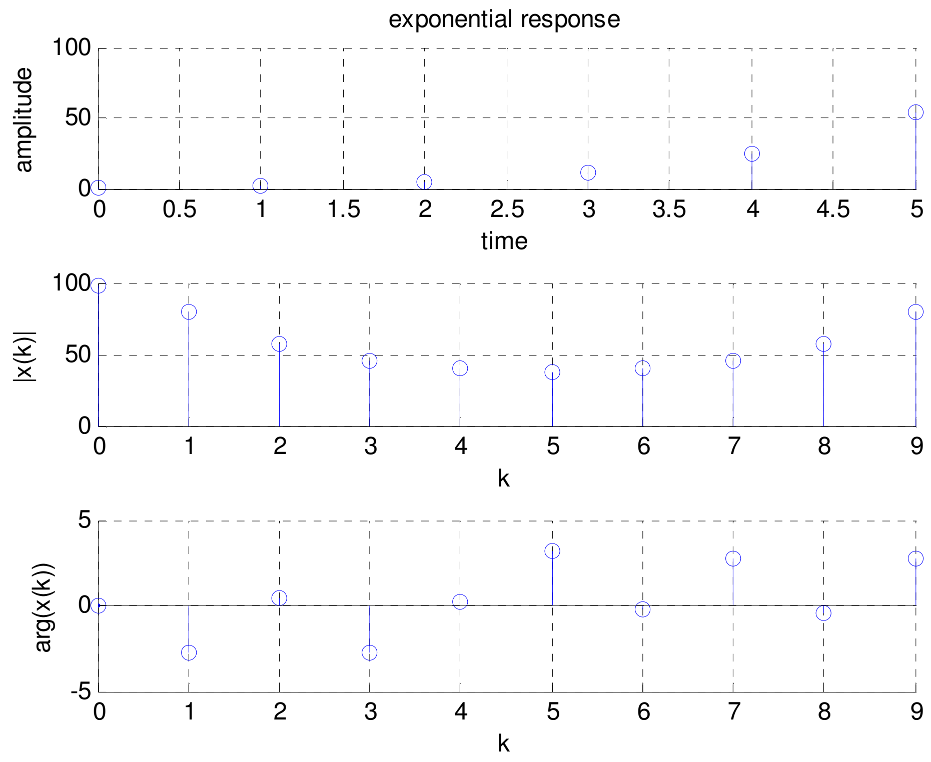
%To compute the FFT of the Exponential sequence and plot magnitude and phase response

```
clc;
clear all;
close all;
%exponential sequence
n=input('enter the length of exponential sequence: ');
t=0:1:n;
a=input('enter "a" value: ');
y=exp(a*t);
input('y=')
disp(y);
subplot(3,1,1);
stem(t,y);
grid;
title('exponential response');
xlabel('time');
ylabel('amplitude');
   disp(y);
   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);
  grid;
   disp(xk);
    xlabel('k');
   ylabel('arg(x(k))');
```

OUTPUTS:

```
enter the length of exponential sequence: 5

enter "a" value: 0.8

y=
    1.0000    2.2255    4.9530   11.0232   24.5325   54.5982

enter the length of the FFT sequence: 10

 98.3324       -73.5207 -30.9223i    50.9418 +24.7831i     -41.7941 -16.0579i     38.8873 + 7.3387i
  -37.3613     38.8873 - 7.3387i     -41.7941 +16.0579i     50.9418 -24.7831i     -73.5207 +30.9223i
```

EXPECTED WAVEFORMS



RESULT:

5. IMPLEMENTATION OF LP FIR FILTERS

AIM: Implementation of Low Pass FIR filter for given sequence.

EQUIPMENT REQUIRED: P – IV Computer
Windows Xp SP2
MATLAB 7.0

THEORY:

A Finite Impulse Response (FIR) filter is a discrete linear time-invariant system whose output is based on the weighted summation of a finite number of past inputs. An FIR transversal filter structure can be obtained directly from the equation for discrete-time convolution.

$$y(n) = \sum_{k=0}^{N-1} x(k)h(n-k) \quad 0 < n < N-1 \qquad (1)$$

In this equation, x(k) and y(n) represent the input to and output from the filter at time n. h(n-k) is the transversal filter coefficients at time n. These coefficients are generated by using FDS (Filter Design Software or Digital filter design package).

FIR – filter is a finite impulse response filter. Order of the filter should be specified. Infinite response is truncated to get finite impulse response. placing a window of finite length does this. Types of windows available are Rectangular, Barlett, Hamming, Hanning, Blackmann window etc. This FIR filter is an all zero filter.

PROCEDURE*:*

1. Enter the passband ripple (rp) and stopband ripple (rs).
2. Enter the passband frequency (fp) and stopband frequency (fs).
3. Enter the sampling frequency (f).
4. Calculate the analog passband edge frequency (wp) and stop band edge frequency (ws)
   wp=2*fp/f ws=2*fs/f
5. Calculate the order of the filter using the following formula,

   $$n = \frac{(-20\log_{10}\ (rp.rs) - 13)}{(14.6\ (fs-fp)/f).}$$

   [Use 'ceil( )' for rounding off the value of 'n' to the nearest integer] if 'n' is an odd number, then reduce its value by '1'.

6. Generate (n+1)th point window coefficients.For example boxcar(n+1) generates a rectangular window. y=boxcar(n+1)
7. Design an nth order FIR filter using the previously generated (n+1) length window function. b=fir1(n,wp,y)
8. Find the frequency response of the filter by using 'freqz( )' function. [h,o]=freqz(b,a,k) This function returns k-point complex frequency response vector 'h' and k-point frequency vector 'o' in radians/samples of the filter.

   $$H(e^{iw}) = \frac{B(e^{jw})}{A(e^{jw})} = \frac{b(1)+b(2)e^{-jw}+\ldots\ldots\ldots\ldots b(m+1)e^{-jmw}}{a(1)+a(2)e^{-jw}+\ldots\ldots\ldots\ldots a(n+1)e^{-jnw}}$$

   Where a, b are vectors containing the denominator and numerator coefficients. Here a=1.

9. Calculate the magnitude of the frequency response in decibels (dB). m= 20*log$_{10}$(abs(h))
10. Plot the magnitude response [magnitude in dB Vs normalized frequency (o/pi)]
11. Give relevant names to x- and y- axes and give an appropriate title for the plot.

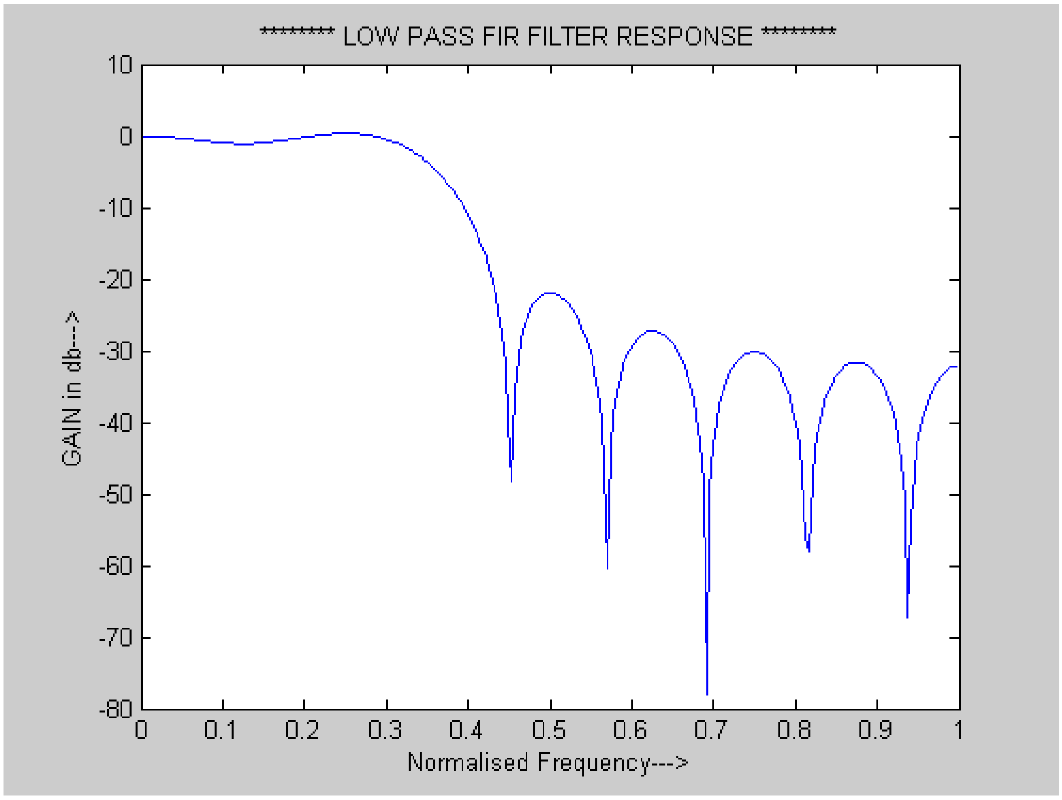
PROGRAM

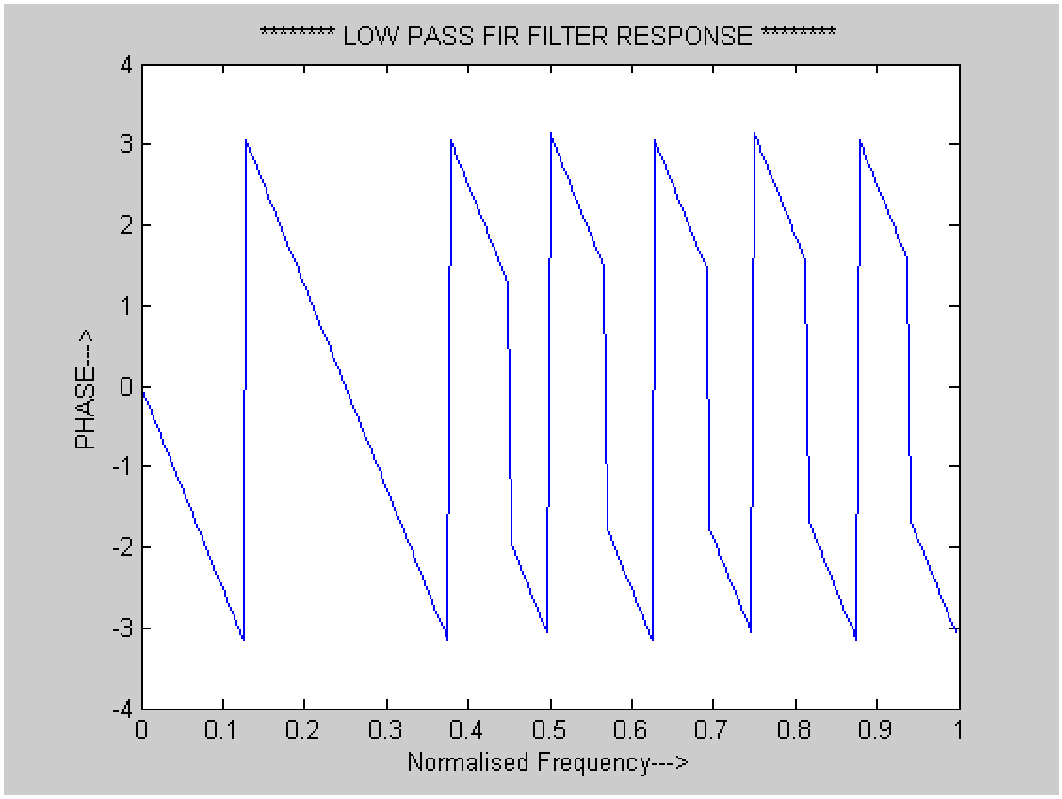
```
clc;
close all;
clear all;
rp=0.05%input('enter the passband ripple');
rs=0.04%input('enter the stopband ripple');
fp=1500%input('enter the passband frequency');
fs=2000%input('enter the stopband frequency');
f=8000%input('enter the sampling freq');
wp=2*fp/f;
ws=2*fs/f;
num=-20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
n1=n+1;
if(rem(n,2)~=0)
n1=n;
n=n-1;
end
y=boxcar(n1);
b=fir1(n,wp,y);
[h,o]=freqz(b,1,256);
m=20*log10(abs(h));
an=angle(h);
figure(1)
plot(o/pi,m);
title('******** LOW PASS FIR FILTER RESPONSE ********');
ylabel('GAIN in db--->');
```

```
xlabel('Normalised Frequency--->');

figure(2)

plot(o/pi,an);

title('******** LOW PASS FIR FILTER RESPONSE ********');

ylabel('PHASE--->');

xlabel('Normalised Frequency--->');
```

Input:

```
rp =   0.05

rs =   0.04

fp =  1500

fs =   2000

f =     8000
```





RESULT: The implimentation of Butterworth Lowpass FIR Filters Completed

6. IMPLEMENTATION OF HP FIR FILTERS

AIM: Implementation of High Pass FIR filter for given sequence.

EQUIPMENT REQUIRED: P – IV Computer

Windows Xp SP2

MATLAB 7.0

THEORY:

A Finite Impulse Response (FIR) filter is a discrete linear time-invariant system whose output is based on the weighted summation of a finite number of past inputs. An FIR transversal filter structure can be obtained directly from the equation for discrete-time convolution.

$$y(n) = \sum_{k=0}^{N-1} x(k)h(n-k) \quad 0 < n < N-1 \qquad (1)$$

In this equation, x(k) and y(n) represent the input to and output from the filter at time n. h(n-k) is the transversal filter coefficients at time n. These coefficients are generated by using FDS (Filter Design Software or Digital filter design package).

FIR – filter is a finite impulse response filter. Order of the filter should be specified. Infinite response is truncated to get finite impulse response. placing a window of finite length does this. Types of windows available are Rectangular, Barlett, Hamming, Hanning, Blackmann window etc. This FIR filter is an all zero filter.

PROCEDURE*:*

1. Enter the passband ripple (rp) and stopband ripple (rs).
2. Enter the passband frequency (fp) and stopband frequency (fs).
3. Enter the sampling frequency (f).
4. Calculate the analog passband edge frequency (wp) and stop band edge frequency (ws)
   wp=2*fp/f ws=2*fs/f
5. Calculate the order of the filter using the following formula,

   $$n = \frac{(-20\log_{10}(rp.rs) - 13)}{(14.6\,(fs-fp)/f)}.$$

   [Use 'ceil( )' for rounding off the value of 'n' to the nearest integer] if 'n' is an odd number, then reduce its value by '1'.
6. Generate (n+1)th point window coefficients.For example boxcar(n+1) generates a rectangular window. y=boxcar(n+1)
7. Design an nth order FIR filter using the previously generated (n+1) length window function. b=fir1(n,wp,'high',y)
8. Find the frequency response of the filter by using 'freqz( )' function. [h,o]=freqz(b,a,k) This function returns k-point complex frequency response vector 'h' and k-point frequency vector 'o' in radians/samples of the filter.

$$H(e^{iw}) = \frac{B(e^{jw})}{A(e^{jw})} = \frac{b(1)+b(2)e^{-jw}+\ldots\ldots\ldots\ldots b(m+1)e^{-jmw}}{a(1)+a(2)e^{-jw}+\ldots\ldots\ldots\ldots a(n+1)e^{-jnw}}$$

Where a, b are vectors containing the denominator and numerator coefficients.

Here a=1.

9. Calculate the magnitude of the frequency response in decibels (dB). m= $20*\log_{10}(abs(h))$
10. Plot the magnitude response [magnitude in dB Vs normalized frequency (o/pi)]
11. Give relevant names to x- and y- axes and give an appropriate title for the plot.

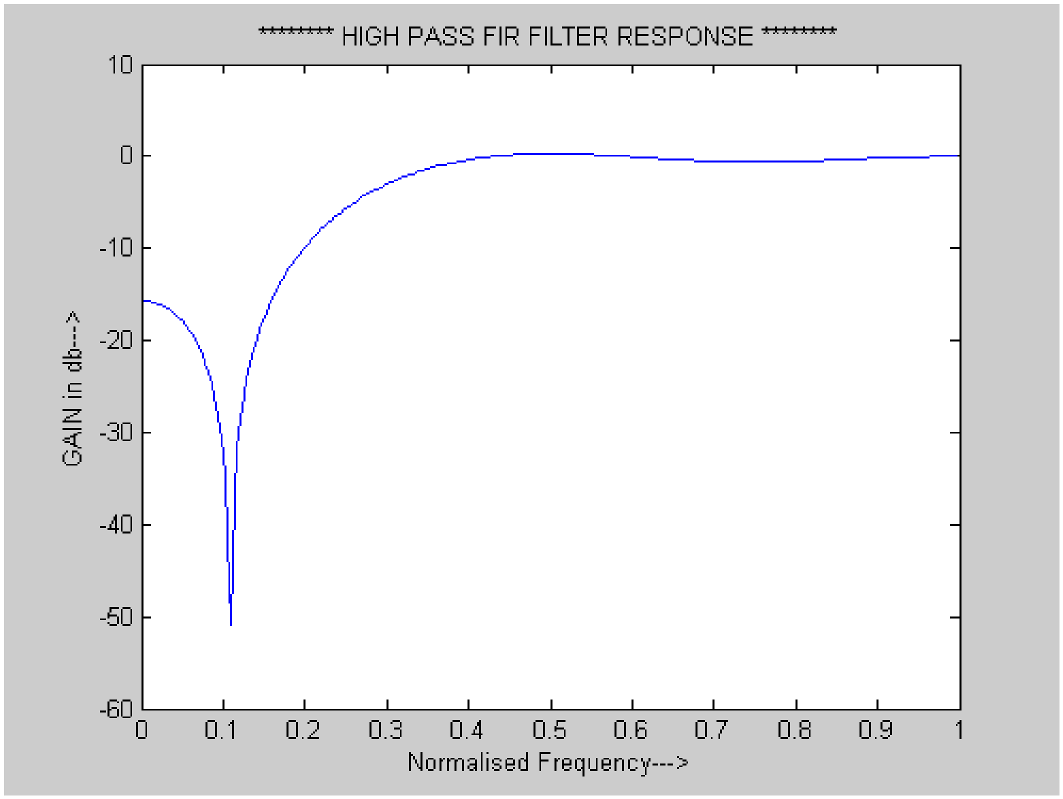
PROGRAM

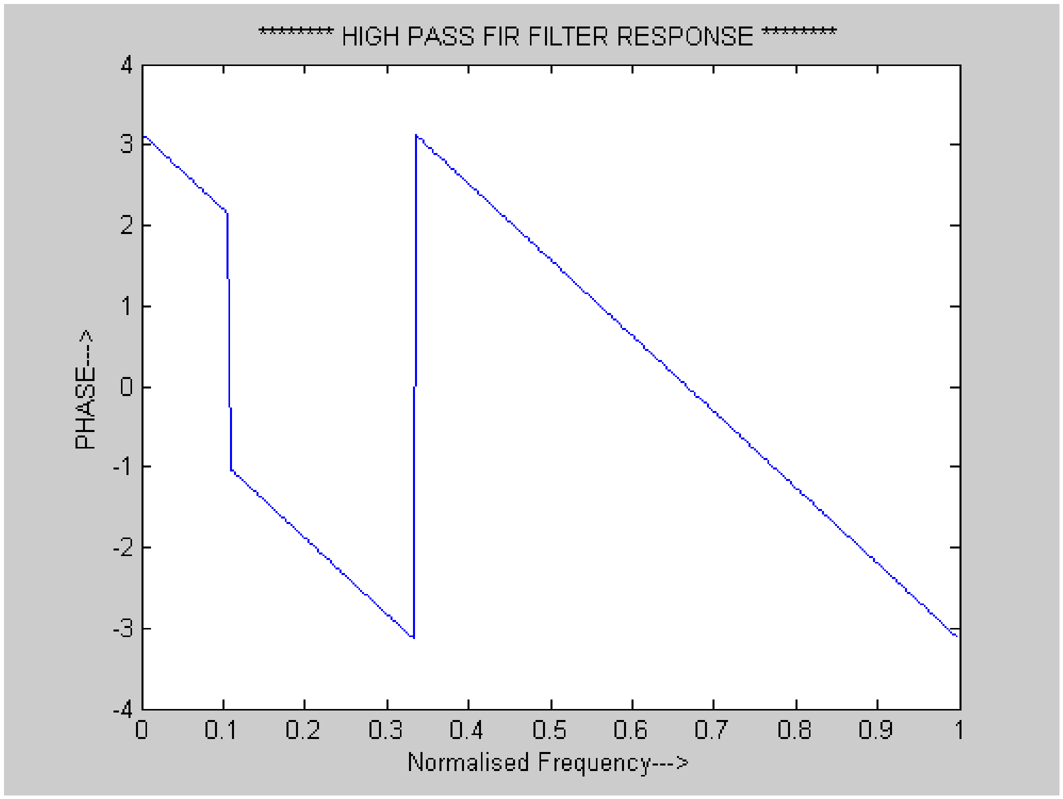
```
clc;

close all;

clear all;

rp=0.05%input('enter the passband ripple');

rs=0.06%input('enter the stopband ripple');

fp=1000%input('enter the passband frequency');

fs=2000%input('enter the stopband frequency');

f=8000%input('enter the sampling freq');

wp=2*fp/f;

ws=2*fs/f;

num=-20*log10(sqrt(rp*rs))-13;

dem=14.6*(fs-fp)/f;

n=ceil(num/dem);

n1=n+1;

if(rem(n,2)~=0)

n1=n;

n=n-1;

end

y=boxcar(n1);

b=fir1(n,wp,'high',y);

[h,o]=freqz(b,1,256);

m=20*log10(abs(h));

an=angle(h);
```

```
figure(1)
plot(o/pi,m);
title('******** HIGH PASS FIR FILTER RESPONSE ********');
ylabel('GAIN in db--->');
xlabel('Normalised Frequency--->');
figure(2)
plot(o/pi,an);
title('******** HIGH PASS FIR FILTER RESPONSE ********');
ylabel('PHASE--->');
xlabel('Normalised Frequency--->');
```

Input:

```
rp  =   0.05
rs  =   0.06
fp  =       1000
fs  =       2000
f   =       8000
```





RESULT: The implimentation of Butterworth Highpass FIR Filters Completed

7. IMPLEMENTATION OF LP IIR FILTERS

AIM: Implementation of Low Pass IIR filter for given sequence.

EQUIPMENT REQUIRED: P – IV Computer

Windows Xp SP2

MATLAB 7.0

THEORY:

PROCEDURE:

1. Enter the pass band ripple (rp) and stop band ripple (rs).
2. Enter the pass band frequency (fp) and stop band frequency (fs).
3. Get the sampling frequency (f).
4. Calculate the analog pass band edge frequencies, w1 and w2.
   w1 = 2*fp/f w2 = 2*fs/f
5. Calculate the order and 3dB cutoff frequency of the analog filter. [Make use of the following function] [n,wn]=buttord(w1,w2,rp,rs,'s')
6. Design an nth order analog high pass Butter worth filter using the following statement.
   [b,a]=butter(n,wn,'s')
7. Find the complex frequency response of the filter by using 'freqs( )'function
   [h,om]=freqs(b,a,w)

   where, w = 0:0.01:pi

   This function returns complex frequency response vector 'h' and frequency vector ‘om’ in radians/samples of the filter.

$$H(s)=\frac{b(s)}{a(s)}=\frac{b(1)S^{nb-1}+b(2)S^{nb-2}+\ldots\ldots\ldots\ldots b(nb)}{a(1)S^{na-1}+a(2)S^{na-2}+\ldots\ldots\ldots\ldots a(na)}$$

Where a,b are the vectors containing the denominator and numerator coefficients.

8. Calculate the magnitude of the frequency response in decibels (dB)
   m=20*$\log_{10}$(abs(h))
9. Plot the magnitude response [magnitude in dB Vs normalized frequency (om/pi)]
10. Give relevant names to x and y axes and give an appropriate title for the plot.
11. Plot all the responses in a single figure window.[Make use of subplot( )].

PROGRAM:

```
clc;
close all;
clear all;
format long
rp=input('enter the passband ripple');
rs=input('enter stopband ripple');
wp=input('enter passband freq');
ws=input('enter stopband freq');
fs=input('enter sampling freq');
w1=2*wp/fs;
w2=2*ws/fs;

%Analog LPF
[n,wn]= buttord(w1,w2,rp,rs);
[b,a]=butter(n,wn,'s');
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
figure(3)
plot(om/pi,m);
```

```
title('**** Analog Output Magnitude *****');
ylabel('gain in db...>');
xlabel('normalised freq..>');
figure(2)
plot(om/pi,an);
title('**** Analog Output Phase ****');
xlabel('normalised freq..>');
ylabel('phase in radians...>');

n
wn

%Digital 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)
plot(om/pi,m);
title('**** Digital Output Magnitude *****');
ylabel('gain in db...>');
xlabel('normalised freq..>');
figure(4)
plot(om/pi,an);
title('**** Digital Output Phase ****');
xlabel('normalised freq..>');
ylabel('phase in radians...>');
n
wn
```

INPUT:

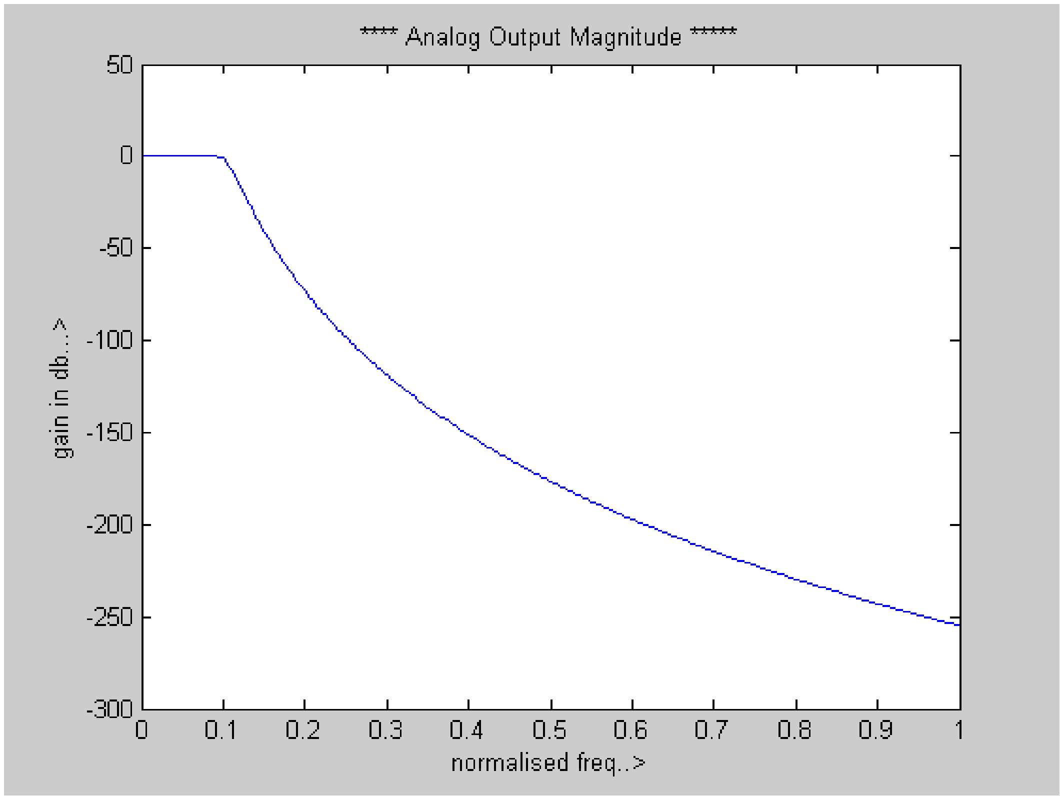
rp = 0.500 rs = 100 wp = 1500 fs = 10000

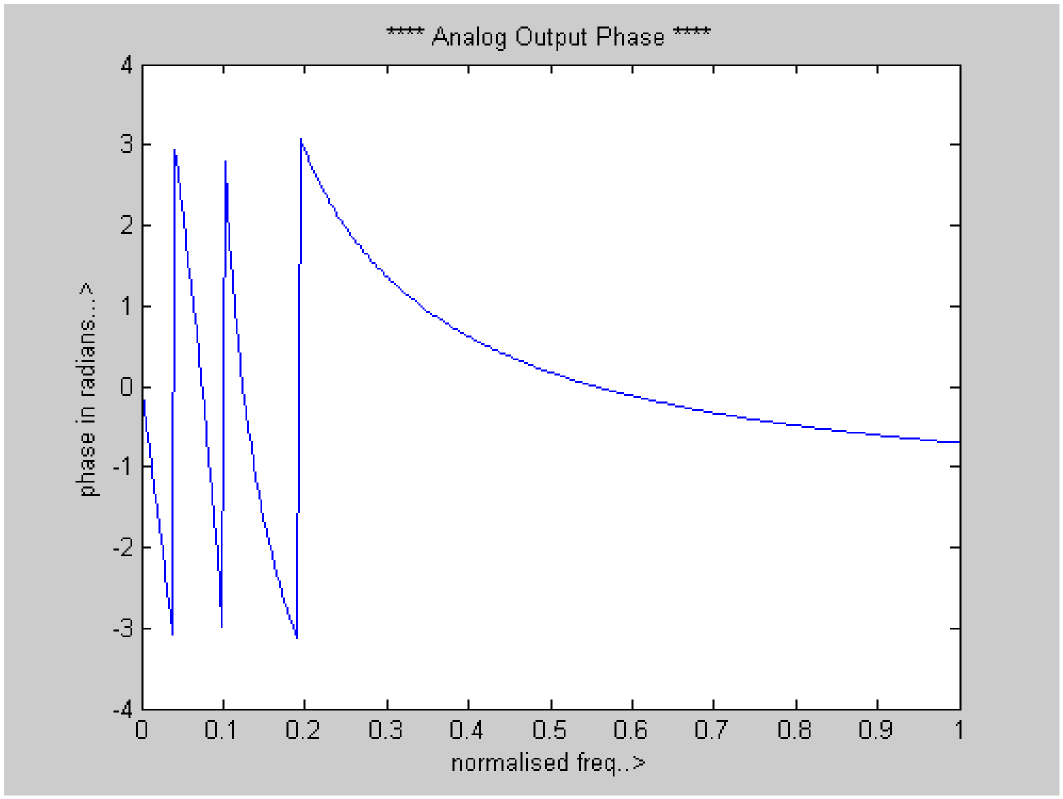
ws = 3000

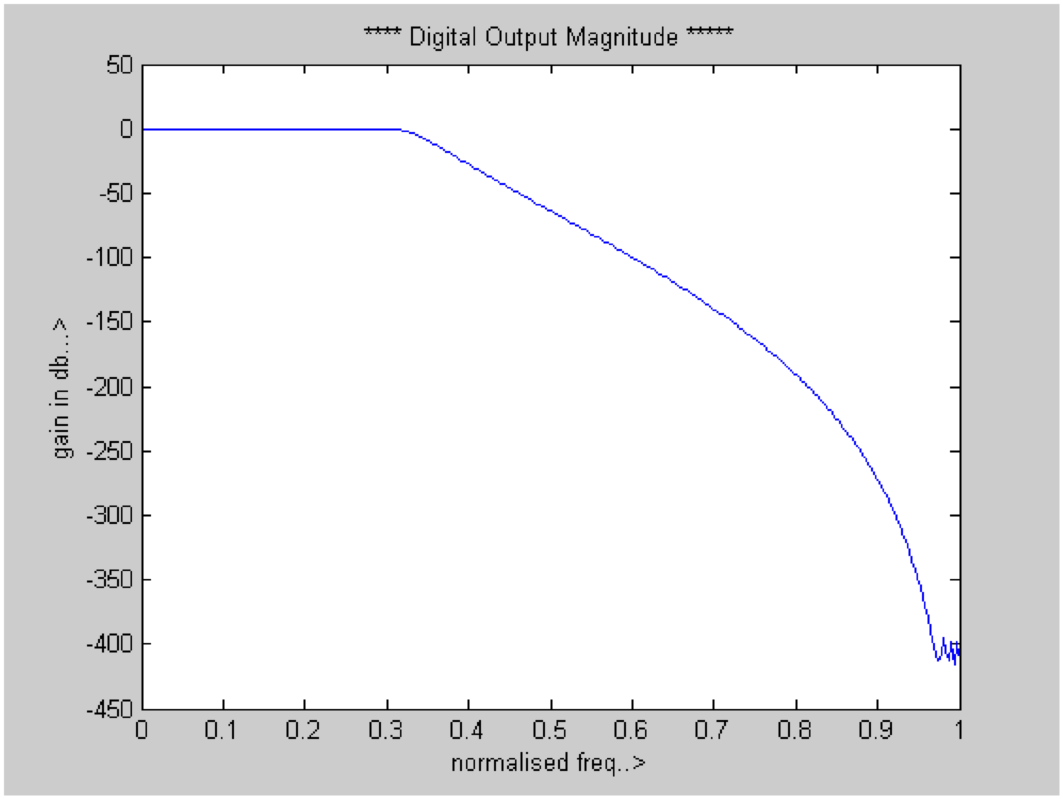
Output:

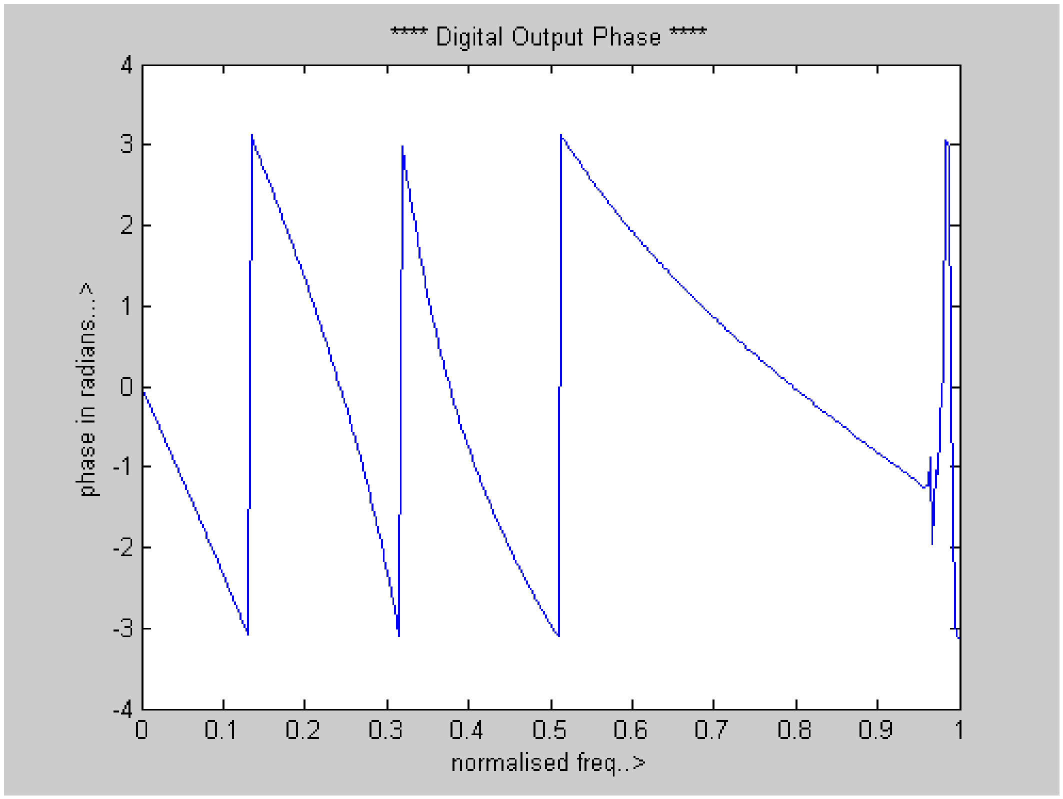
n = 13 n = 13

wn = 0.32870936151976 wn = 0.32870936151976




**** Analog Output Phase ****
phase in radians...>
normalised freq..>
4
3
2
1
0
-1
-2
-3
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


**** Digital Output Magnitude *****
gain in db...>
normalised freq..>
50
0
-50
-100
-150
-200
-250
-300
-350
-400
-450
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Result: Butter worth Digital and analog low pass IIR filters are implemented using MATLAB.

8. IMPLEMENTATION OF HP IIR FILTERS

AIM: To implement the analog & digital High Pass IIR filter.

EQUIPMENT REQUIRED: P – IV Computer

Windows Xp SP2

MATLAB 7.0

THEORY:

PROCEDURE:

1. Enter the pass band ripple (rp) and stop band ripple (rs).
2. Enter the pass band frequency (fp) and stop band frequency (fs).
3. Get the sampling frequency (f).
4. Calculate the analog pass band edge frequencies, w1 and w2.
   w1 = 2*fp/f w2 = 2*fs/f
5. Calculate the order and 3dB cutoff frequency of the analog filter. [Make use of the following function] [n,wn]=buttord(w1,w2,rp,rs,'s')
6. Design an nth order analog high pass Butter worth filter using the following statement.
   [b,a]=butter(n,wn,'s')
7. Find the complex frequency response of the filter by using 'freqs( )'function
   [h,om]=freqs(b,a,w)

where, w = 0:0.01:pi

This function returns complex frequency response vector 'h' and frequency vector 'om' in radians/samples of the filter.

$$H(s) = \frac{b(s)}{a(s)} = \frac{b(1)S^{nb-1}+b(2)S^{nb-2}+\ldots\ldots\ldots\ldots\ldots b(nb)}{a(1)S^{na-1}+a(2)S^{na-2}+\ldots\ldots\ldots\ldots\ldots a(na)}$$

Where a,b are the vectors containing the denominator and numerator coefficients.

8. Calculate the magnitude of the frequency response in decibels (dB)
   $m=20*\log_{10}(abs(h))$
9. Plot the magnitude response [magnitude in dB Vs normalized frequency (om/pi)]
10. Give relevant names to x and y axes and give an appropriate title for the plot.
11. Plot all the responses in a single figure window.[Make use of subplot( )].

PROGRAM:

```
clc;
close all;
clear all;
format long
rp=input('enter the passband ripple');
rs=input('enter stopband ripple');
wp=input('enter passband freq');
ws=input('enter stopband freq');
fs=input('enter sampling freq');
w1=2*wp/fs;
w2=2*ws/fs;

%Analog HPF
[n,wn]= buttord(w1,w2,rp,rs);
[b,a]=butter(n,wn,'high','s');
w=0:.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
```

```
figure(1)
plot(om/pi,m);
title('**** Analog Output Magnitude *****');
ylabel('gain in db...>');
xlabel('normalised freq..>');
figure(2)
plot(om/pi,an);
title('**** Analog Output Phase ****');
xlabel('normalised freq..>');
ylabel('phase in radians...>');
n
wn

%Digital HPF
[n,wn]= buttord(w1,w2,rp,rs);
[b,a]=butter(n,wn,'high');
w=0:.01:pi;
[h,om]=freqz(b,a,w); m=20*log10(abs(h));
an=angle(h);
figure(3)
plot(om/pi,m);
title('**** Digital Output Magnitude *****');
ylabel('gain in db...>');
xlabel('normalised freq..>');
figure(4)
plot(om/pi,an);
title('**** Digital Output Phase ****');
xlabel('normalised freq..>');
ylabel('phase in radians...>');
n
wn
```

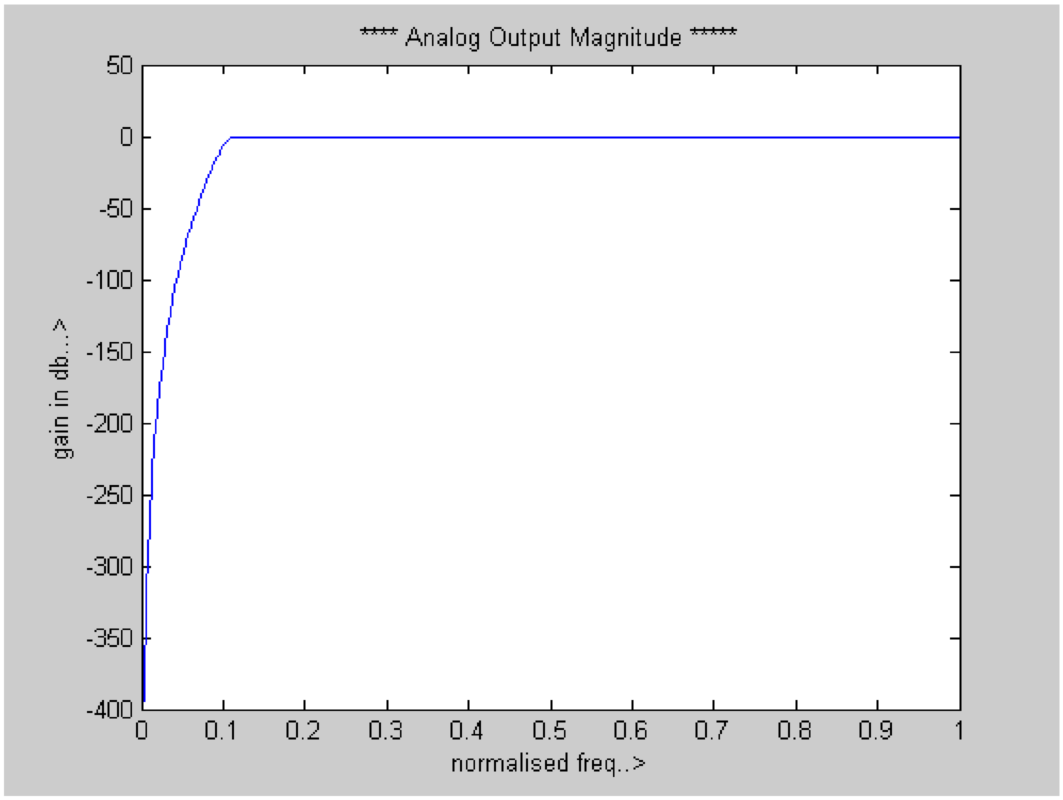
Input:

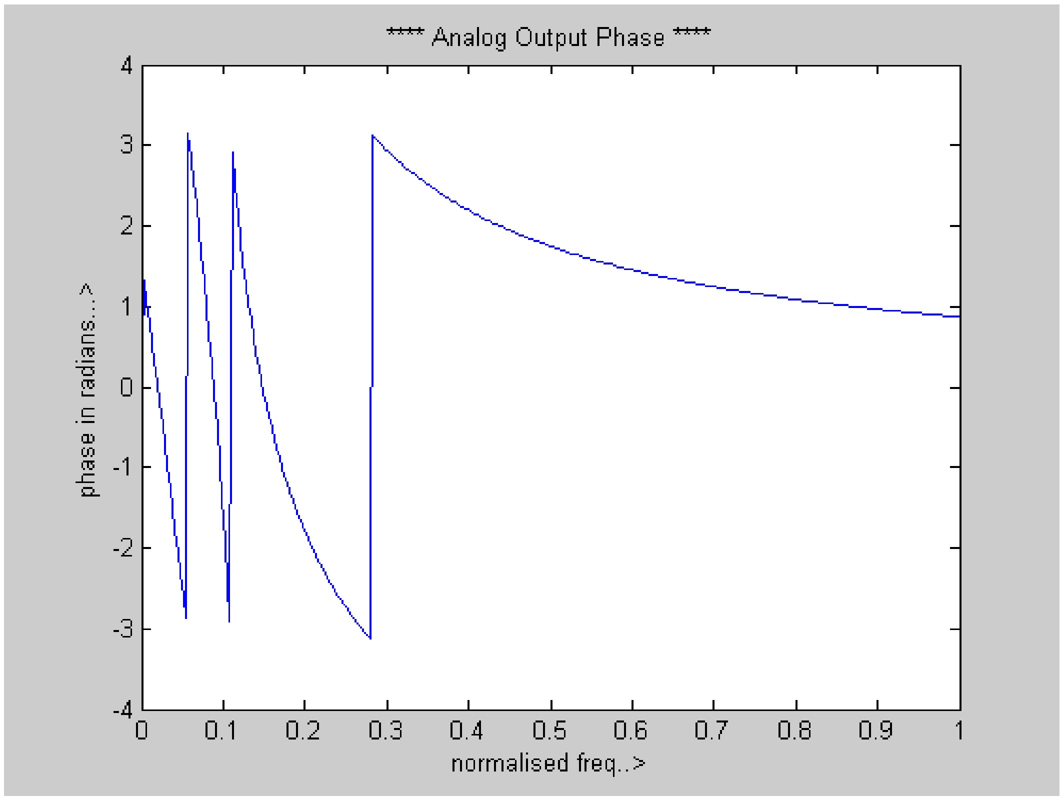
r_p = 0.5000 r_s = 100

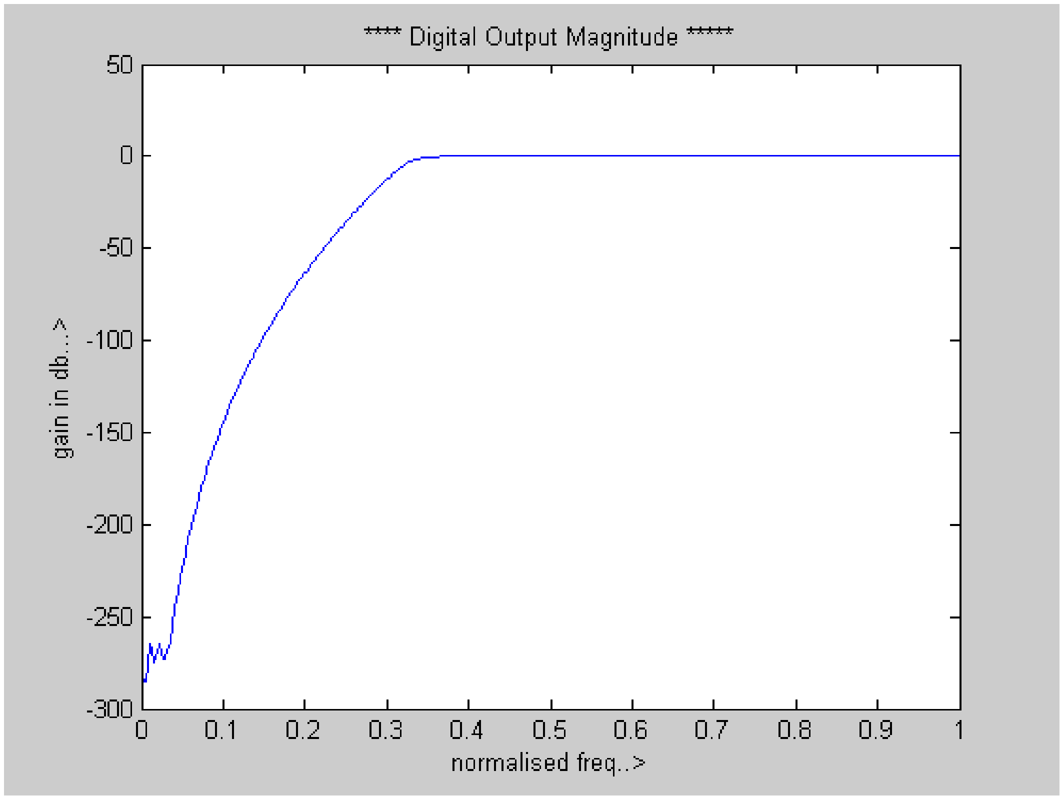
w_p = 1200 w_s =   2400 f_s =   8000

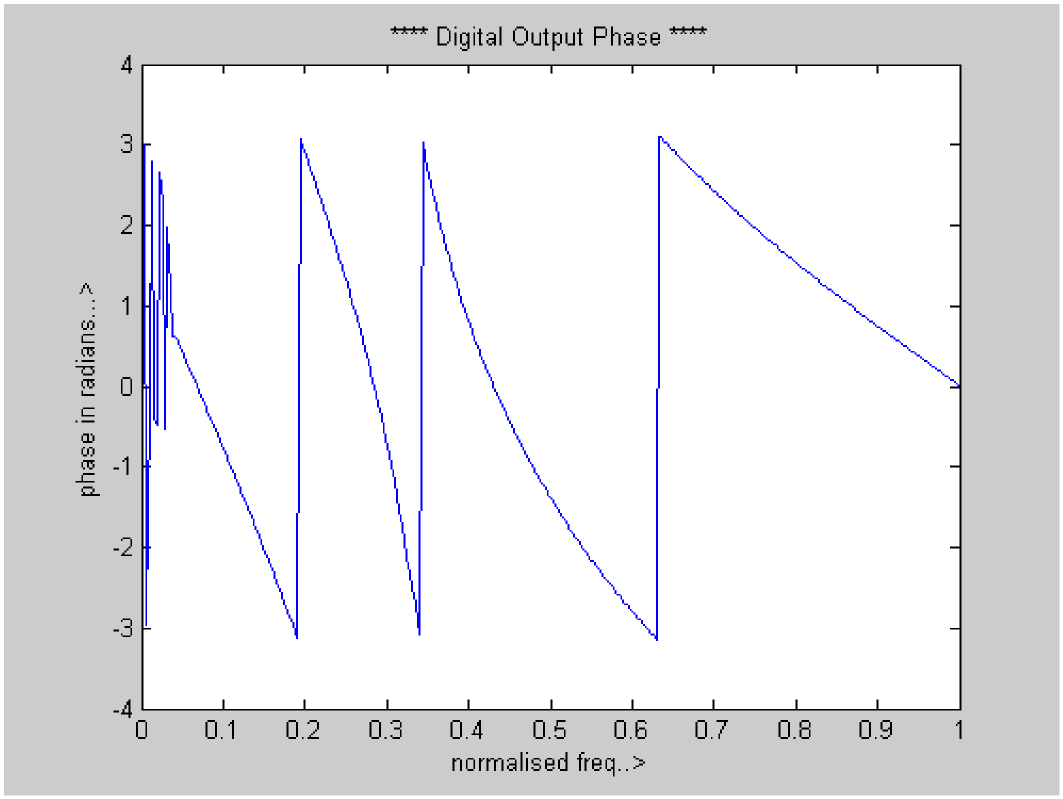
Output:

```
n =     13                                   n =     13

wn =   0.32870936151976                      wn =   0.32870936151976
```






**** Digital Output Magnitude *****
gain in db...>
normalised freq..>
50
0
-50
-100
-150
-200
-250
-300
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


**** Digital Output Phase ****
phase in radians...>
normalised freq..>
4
3
2
1
0
-1
-2
-3
-4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

9. DTMF SIGNAL GENERATION

AIM:

The objective of this program is To Generate Dual Tone Multiple Frequency (DTMF) Signals.

EQUIPMENT REQUIRED: P – IV Computer
Windows Xp SP2
MATLAB 7.0

THEORY:

Dual Tone Multiple Frequency (DTMF) Signals.

Program:

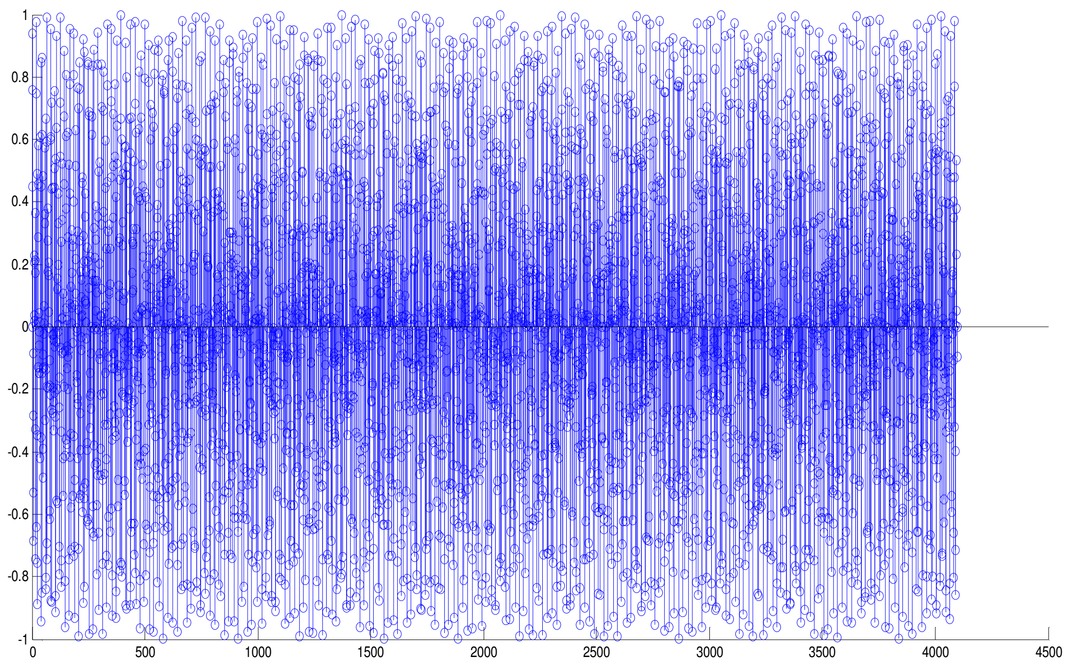
```
clc;
clear all;
close all;
number=input('enter a phone number with no spaces','s');
%number=1;
fs=8192;    % fs is the sampling Frequency
T=0.5;      % T stores how for how long a tone will be played
x= 2*pi*[697 770 852 941];
y= 2*pi*[1209 1336 1477 1633];
t=[0:1/fs:T]'
tx=[sin(x(1)*t),sin(x(2)*t),sin(x(3)*t),sin(x(4)*t)]/2;
ty=[sin(y(1)*t),sin(y(2)*t),sin(y(3)*t),sin(y(4)*t)]/2;
for k=1:length(number)
   switch number(k)
       case '1'
          tone = tx(:,1)+ty(:,1);
          sound(tone);
          stem(tone);

       case '2'
          tone = tx(:,1)+ty(:,2);
          sound(tone);
```

```
  stem(tone);

case '3'
  tone = tx(:,1)+ty(:,3);
  sound(tone);
  stem(tone);

case '4'
  tone = tx(:,2)+ty(:,1);
  sound(tone);
  stem(tone);

case '5'
  tone = tx(:,2)+ty(:,2);
  sound(tone);
  stem(tone);

case '6'
  tone = tx(:,2)+ty(:,3);
  sound(tone);
  stem(tone);

case '7'
  tone = tx(:,3)+ty(:,1);
  sound(tone);
  stem(tone);

case '8'
  tone = tx(:,3)+ty(:,2);
  sound(tone);
  stem(tone);
```

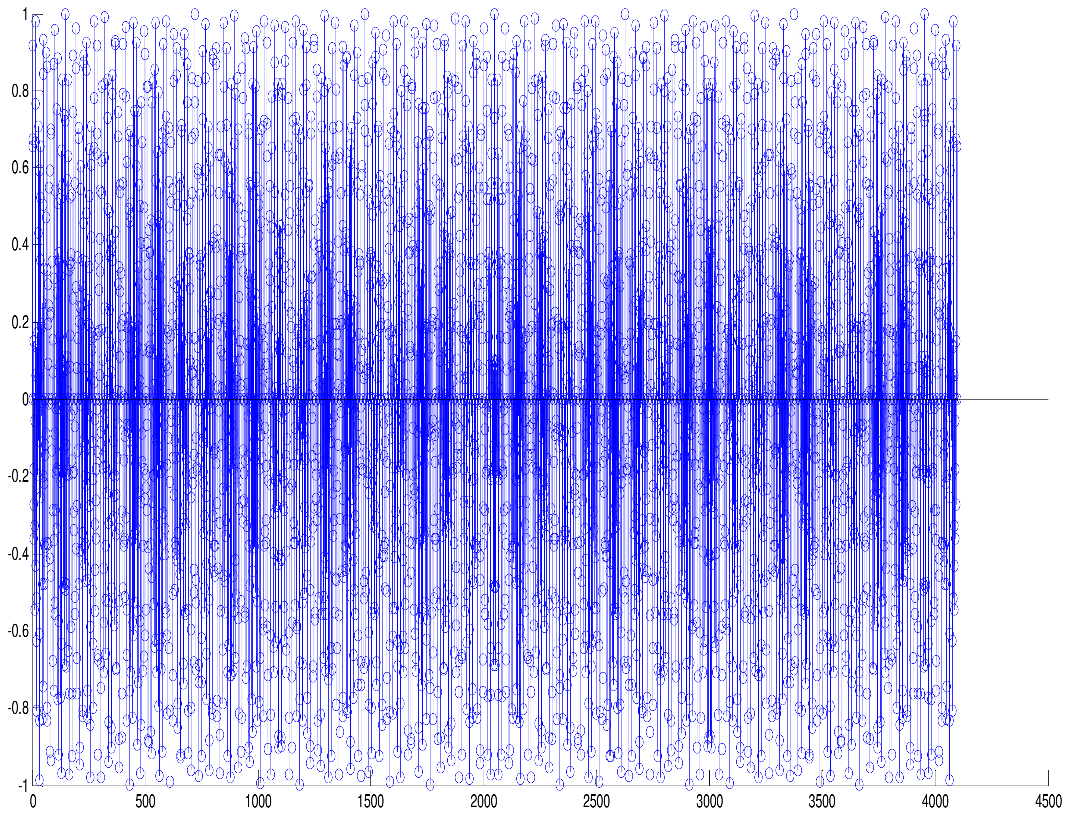
```
            case '9'
               tone = tx(:,3)+ty(:,3);
               sound(tone);
               stem(tone);

            case '*'
               tone = tx(:,4)+ty(:,1);
               sound(tone);
               stem(tone);

            case '0'
               tone = tx(:,4)+ty(:,2);
               sound(tone);
               stem(tone);

            case '#'
               tone = tx(:,4)+ty(:,3);
               sound(tone);
               stem(tone);

           otherwise
               disp('invalid number');

        end
        pause(2.70)
    end
```

Input: 0123456789*#

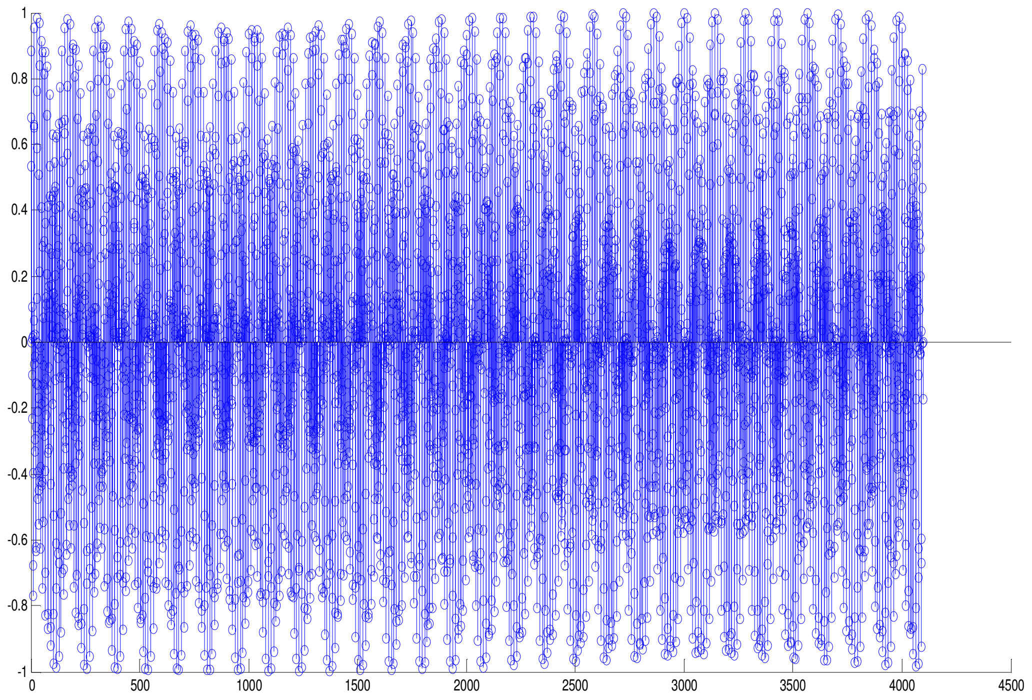
% Graph:



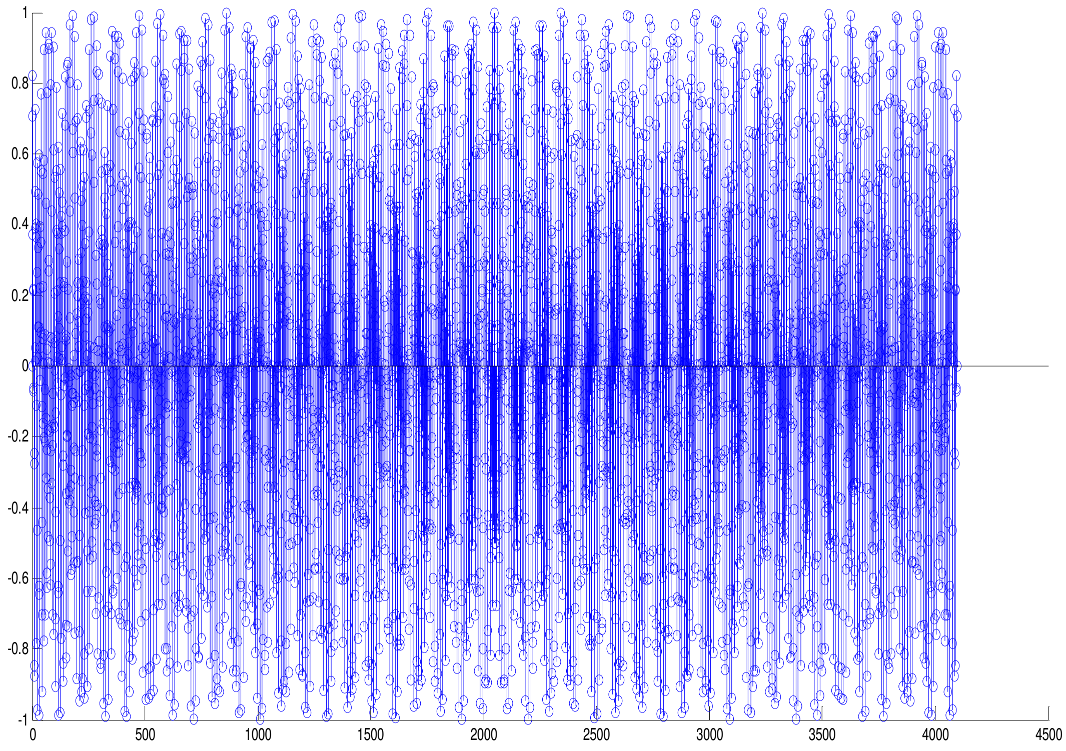
0



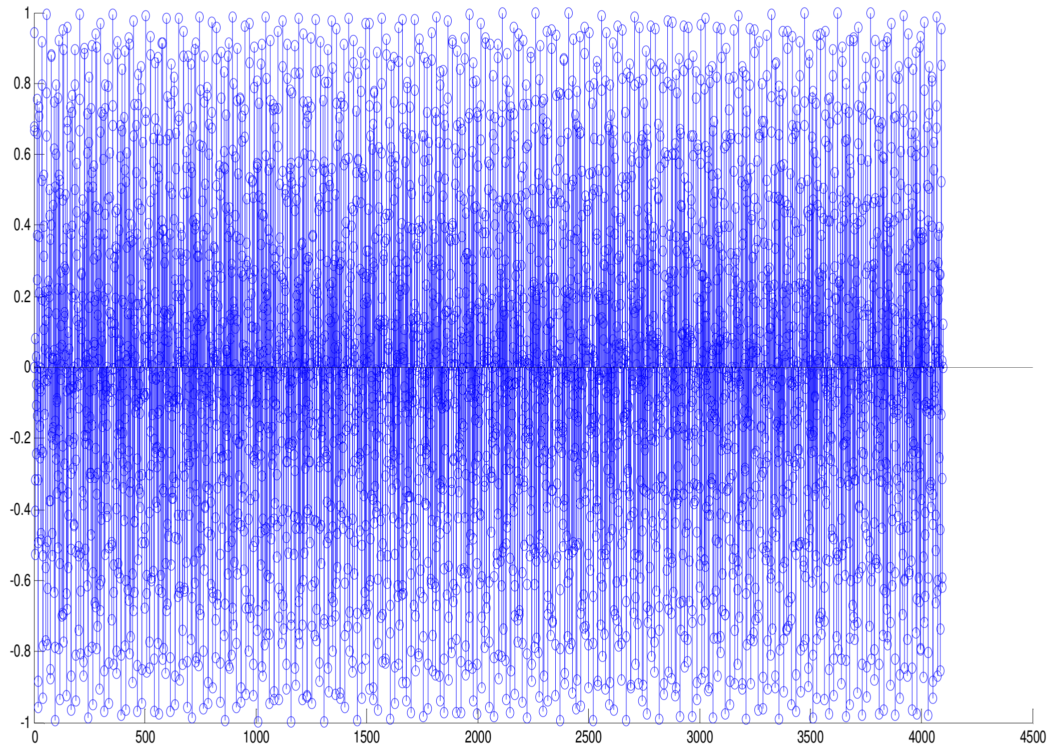
1



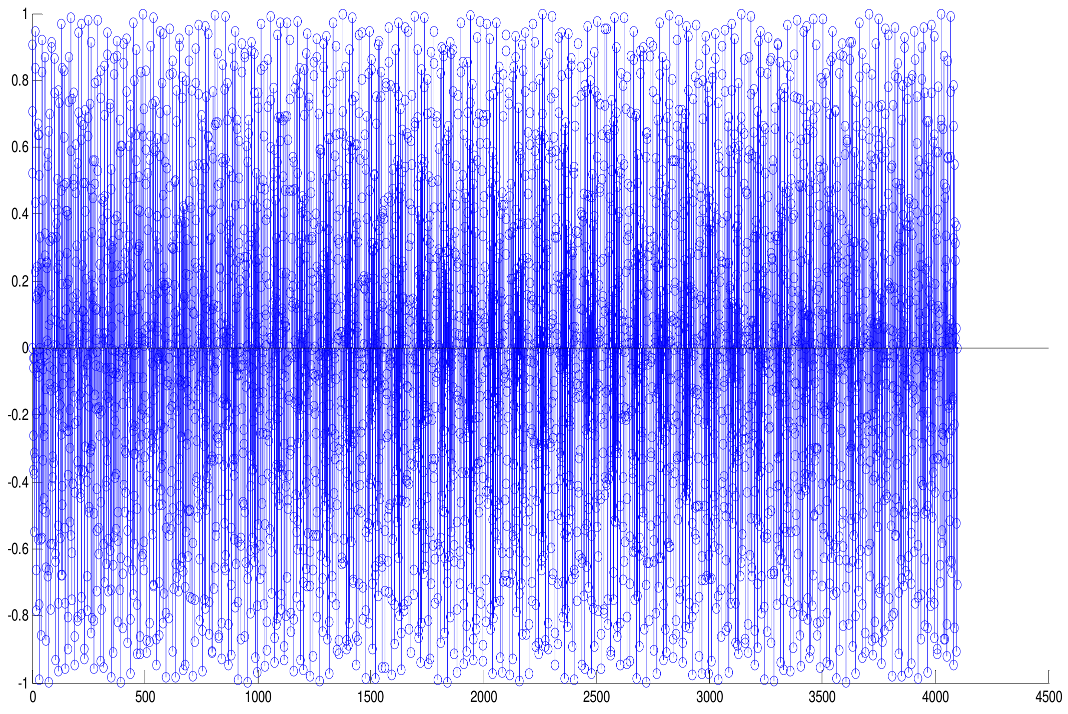
2



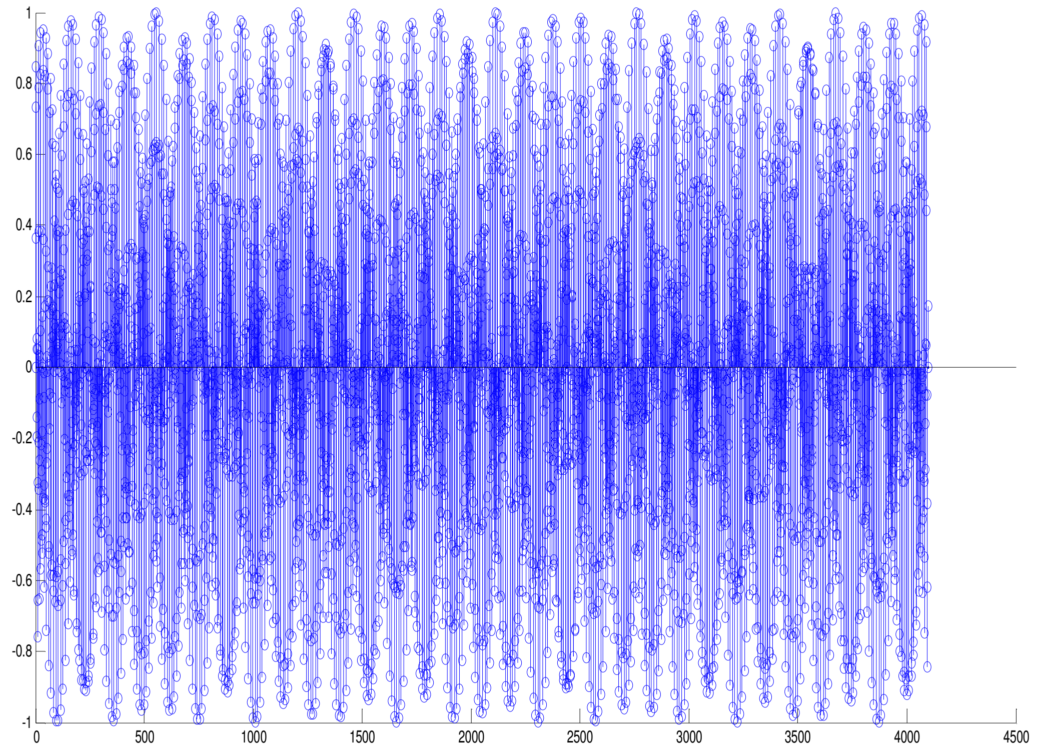
3


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

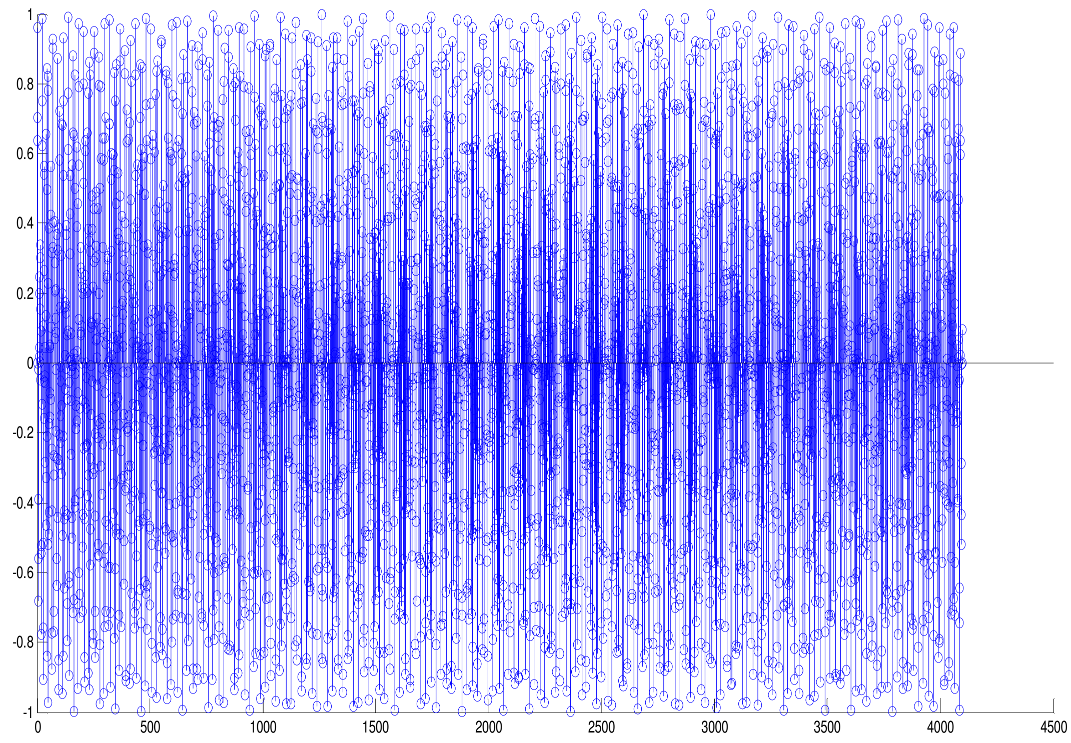
4


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

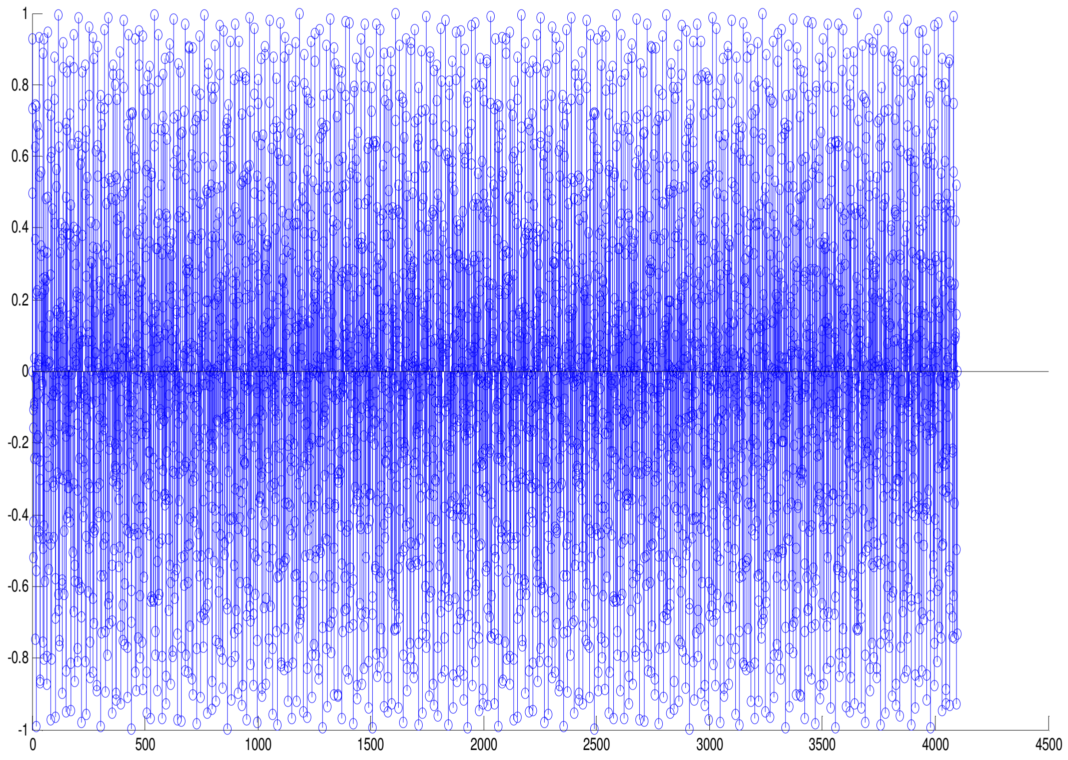
5


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

6

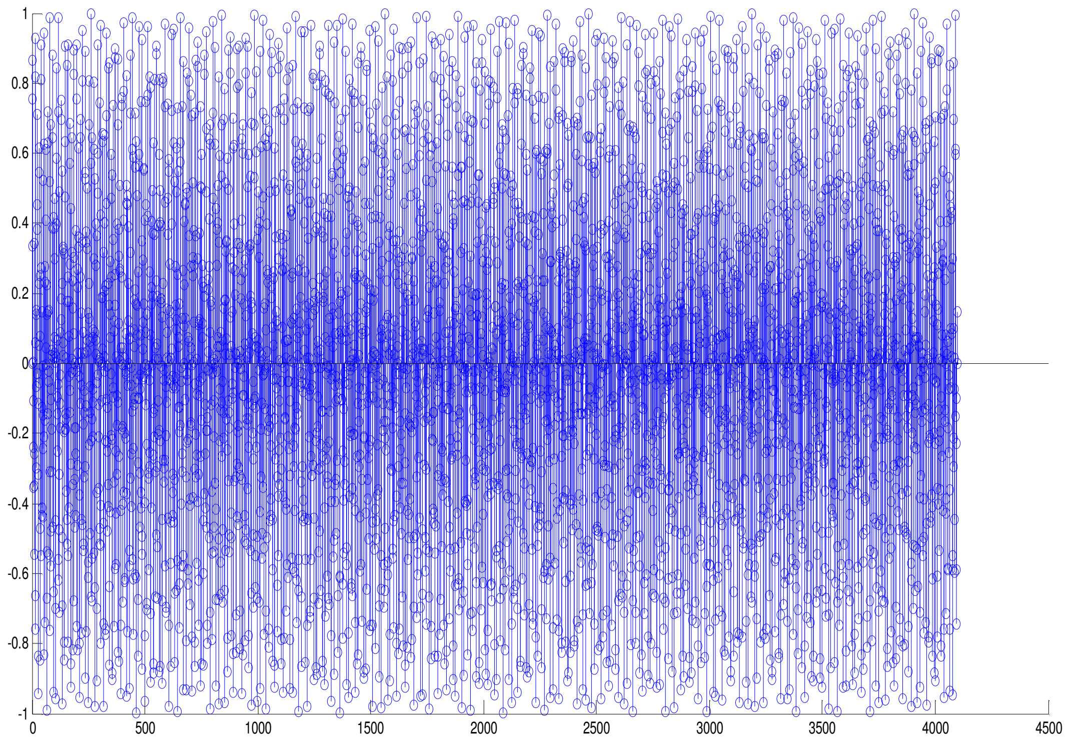

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

7



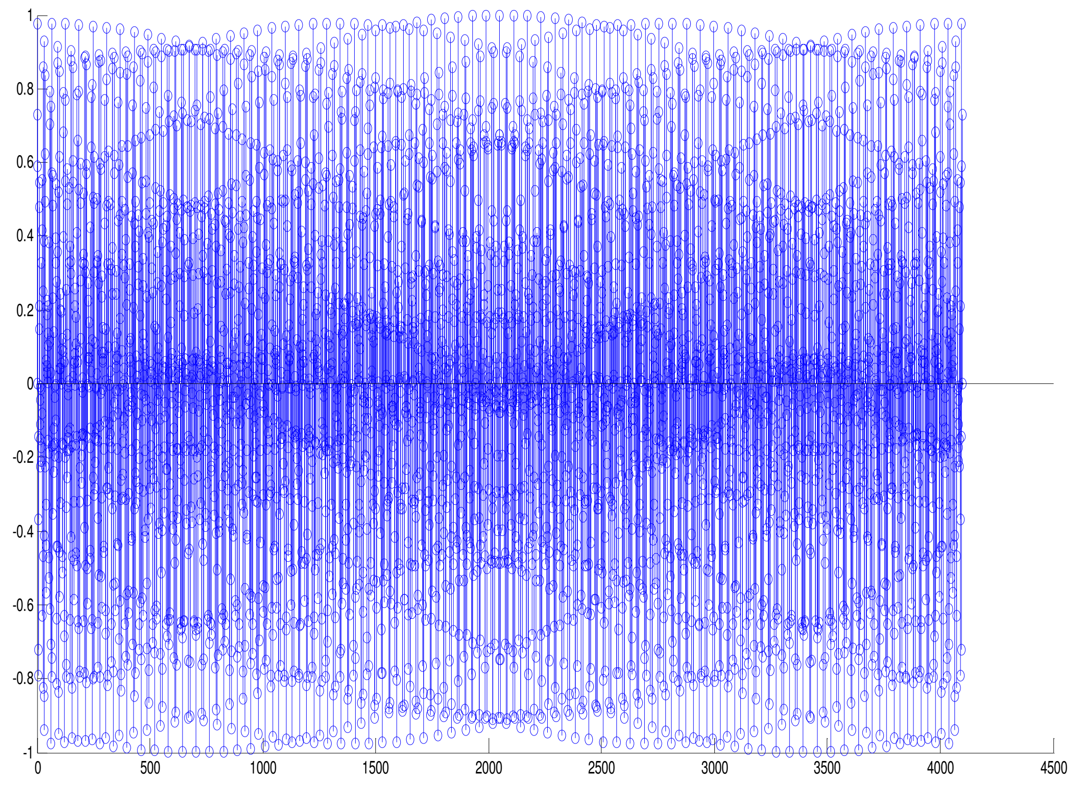
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

8

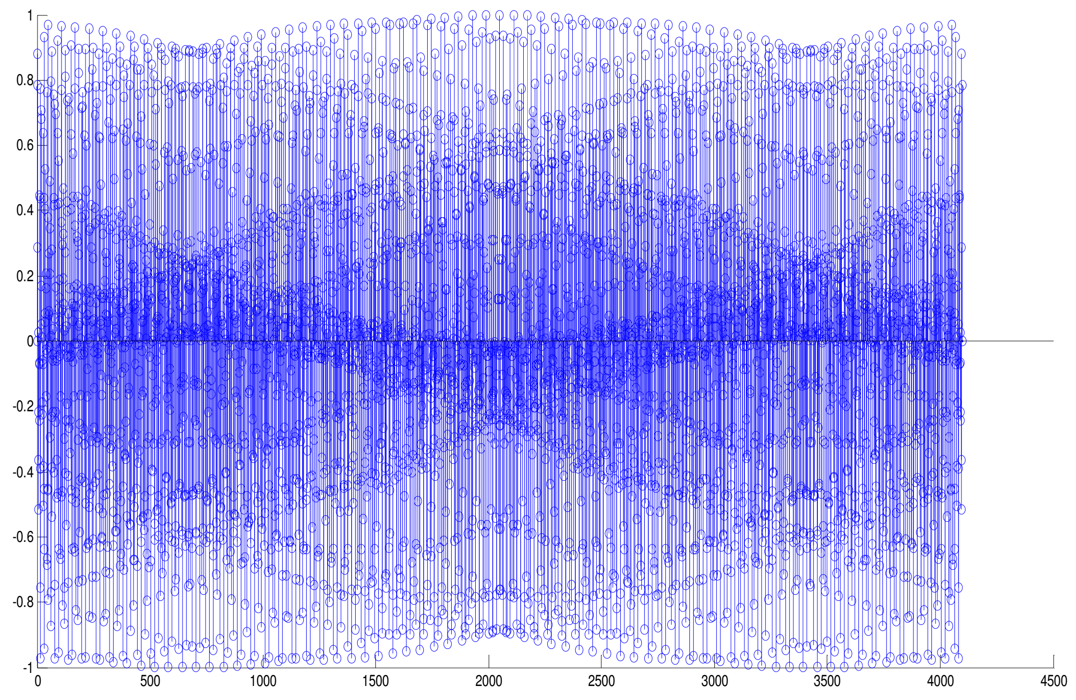


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

9


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

*


1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500

#

Comment:

Dual Tone Multiple Frequency (DTMF) is obtained for different frequencies.

Result:

This MATLAB program has been written to Dual Tone Multiple Frequency (DTMF) Signals.

10. INTERPOLATION

AIM:

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

EQUIPMENT REQUIRED: P – IV Computer
Windows Xp SP2
MATLAB 7.0

THEORY:

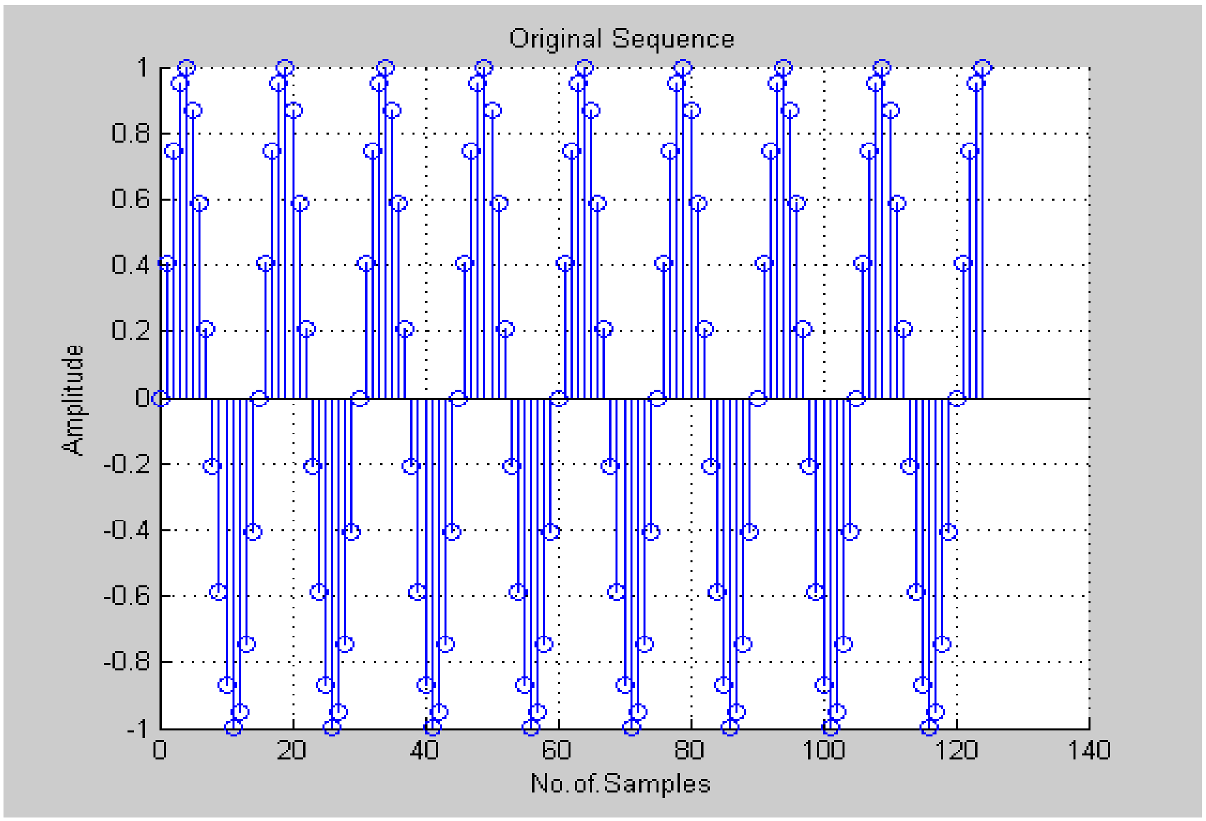
Up sampling on the Given Input Sequence and Interpolating the sequence.

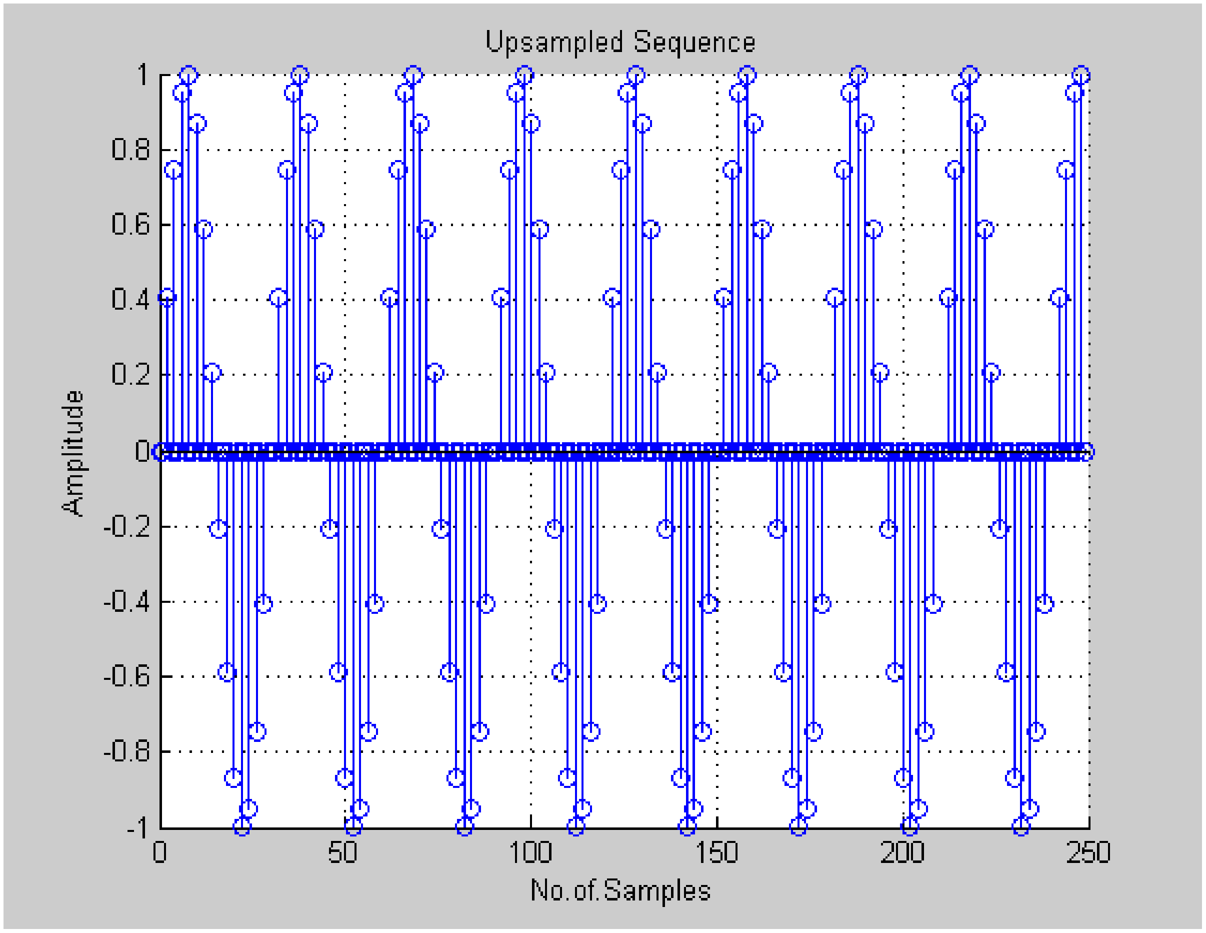
PROGRAM:

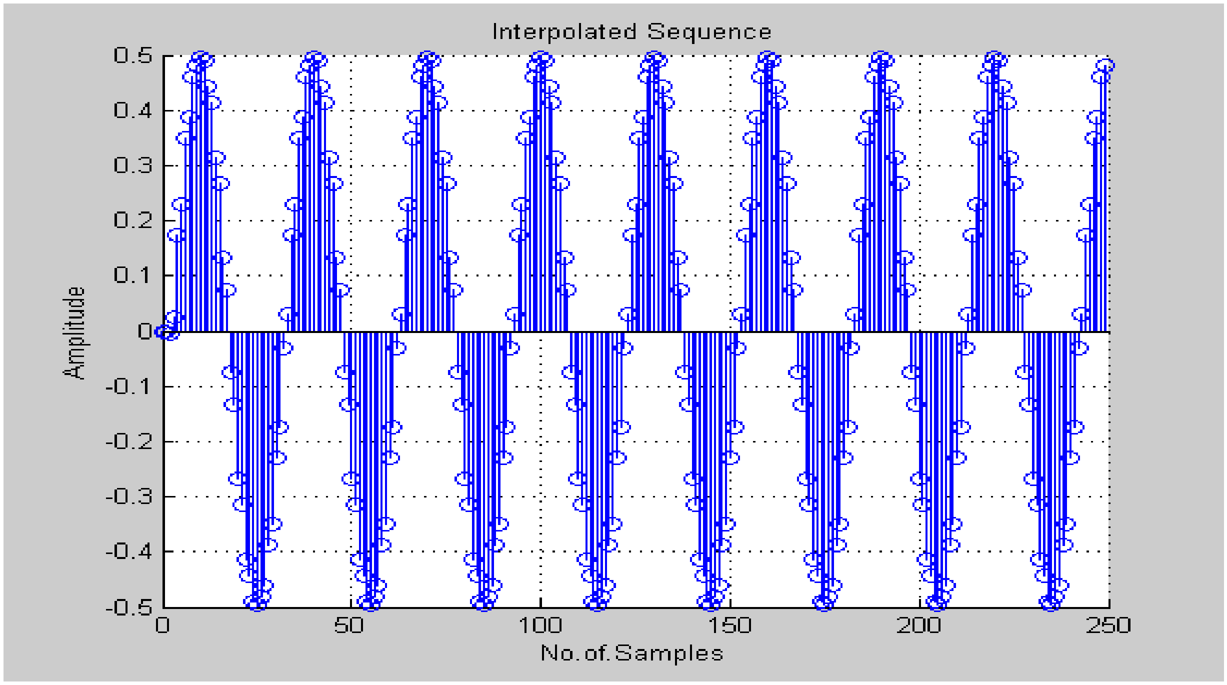
```
clc;
clear all;
close all;
N=125;
n=0:1:N-1;
x=sin(2*pi*n/15);
L=2;
figure(1)
stem(n,x);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Original Sequence');
x1=[zeros(1,L*N)];
n1=1:1:L*N;
j =1:L:L*N;
x1(j)=x;
figure(2)
stem(n1-1,x1);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');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

```
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:







Comment:

Interpolated sequence is observed from graph.

Result:

This MATLAB program has been written to perform interpolation on the Given Input Sequence.

11. DECIMATION

AIM:

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

EQUIPMENT REQUIRED: P – IV Computer

Windows Xp SP2

MATLAB 7.0

THEORY:

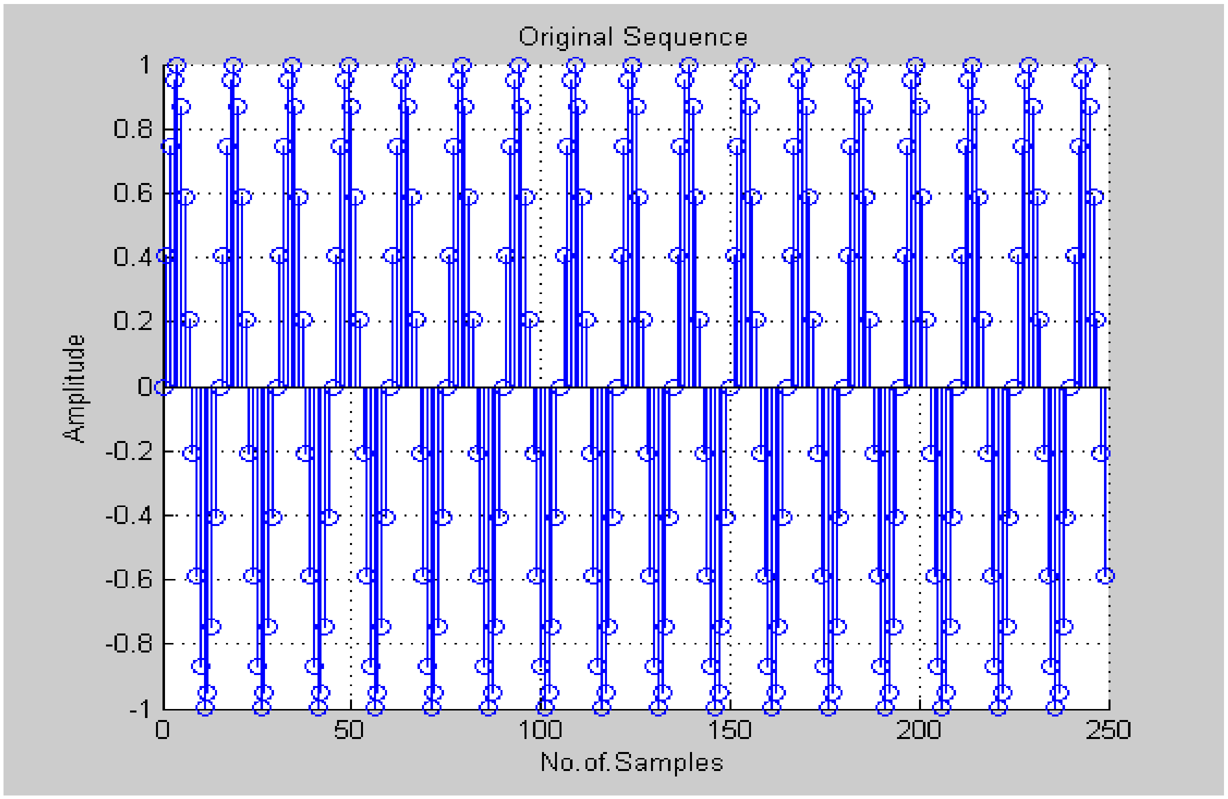
Decimation on the Given Input Sequence by using filter with filter-coefficients a and b.

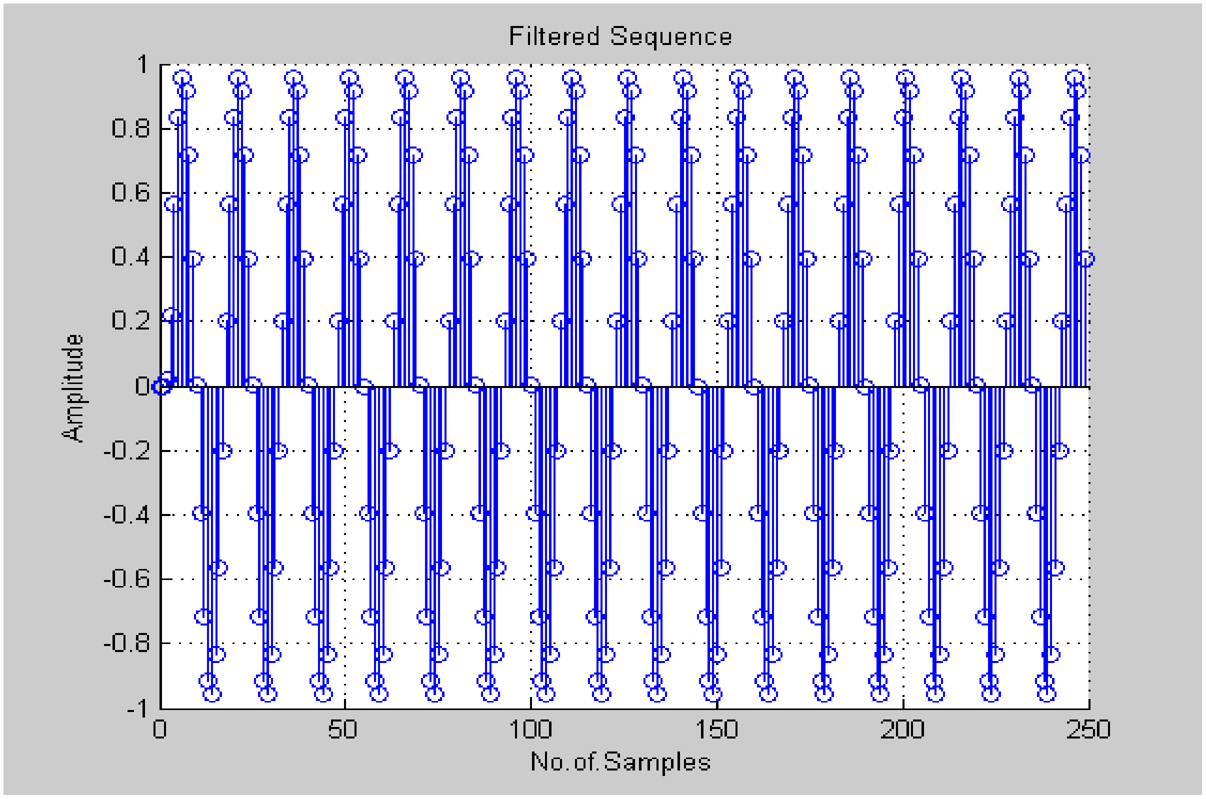
PROGRAM:

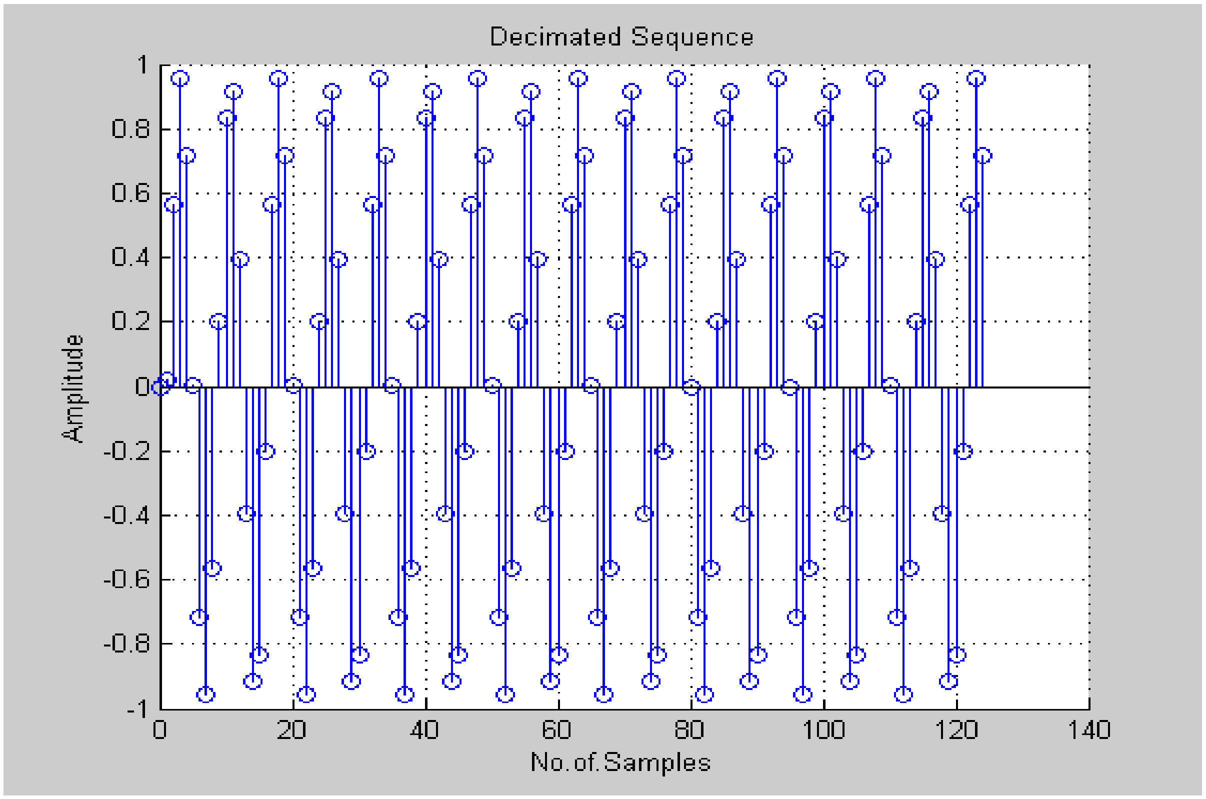
```
clc;
clear all;
close all;
N=250 ;
n=0:1:N-1;
x=sin(2*pi*n/15);
M=2;
figure(1)
stem(n,x);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Original Sequence');
a=1;
b=fir1(5,0.5,'Low');
y=filter(b,a,x);
figure(2)
stem(n,y);
grid on;
xlabel('No.of.Samples');
ylabel('Amplitude');
title('Filtered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




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

Comment:

Decimated sequence is observed from graph.

Result:

This MATLAB program has been written to perform Decimation on the Given Input Sequence.

12. POWER SPECTRAL DENSITY

AIM: Identification of Power Spectral Density using FFT..

EQUIPMENTS:

Operating System	– Windows XP
Constructor	**-** Simulator
Software	- CCStudio 3 & MATLAB 7.5

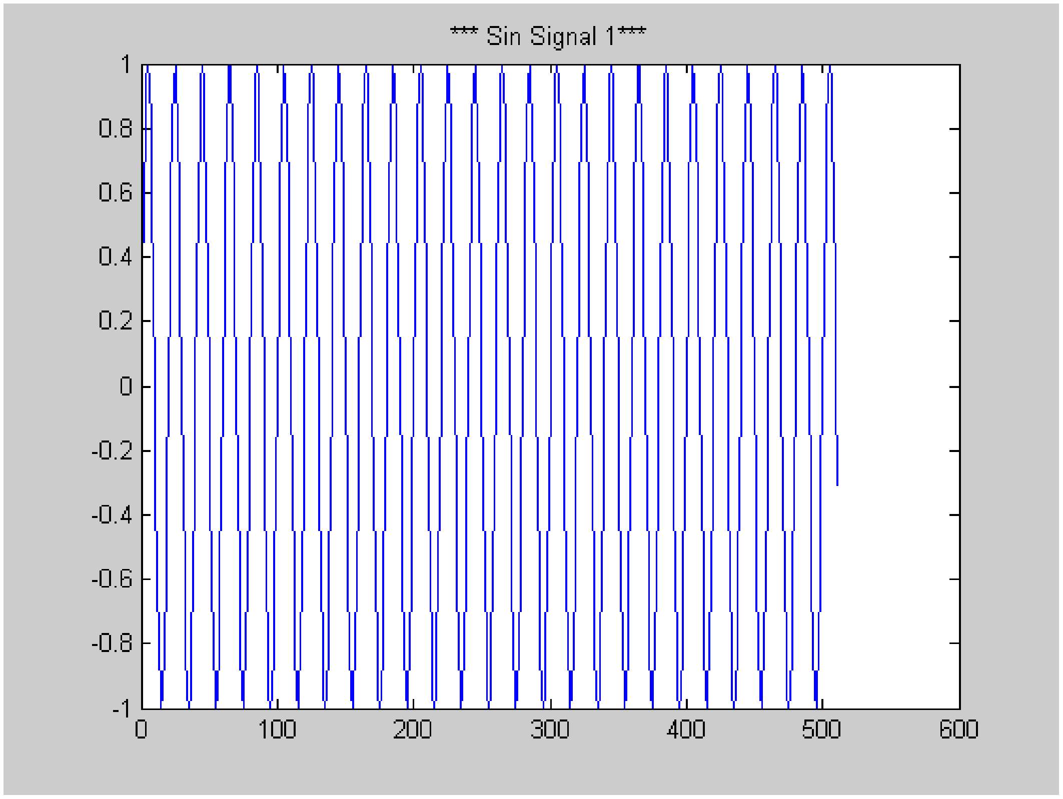
THEORY:

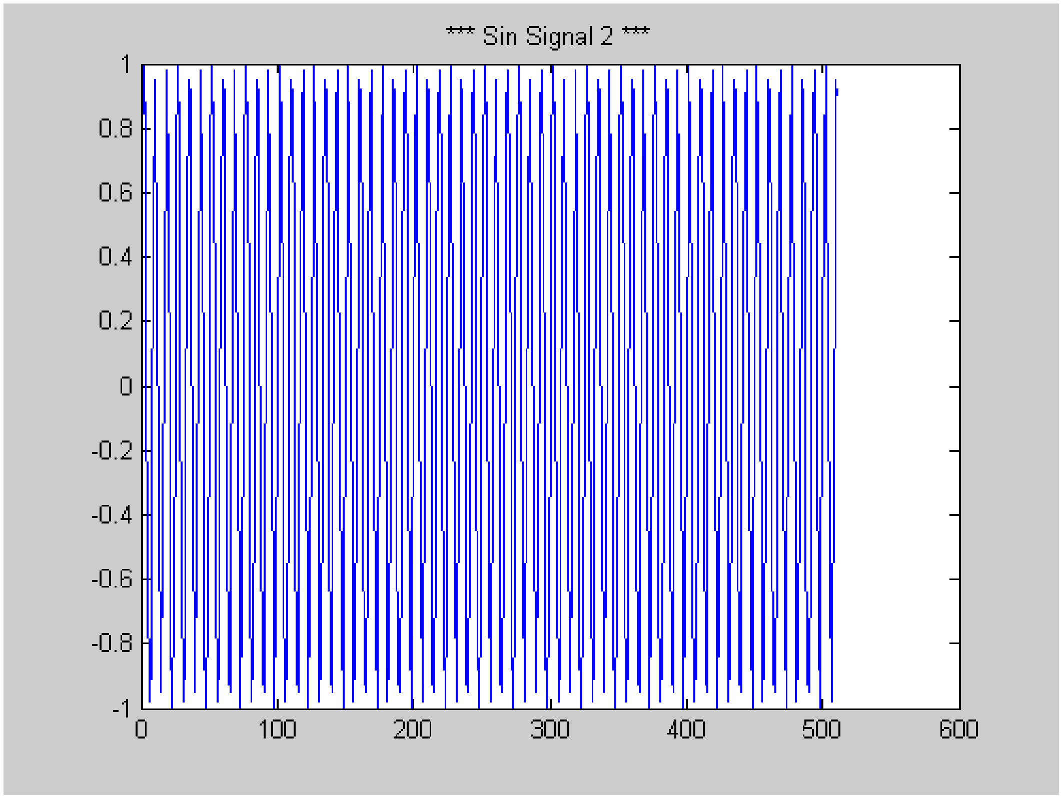
PROCEDURE:

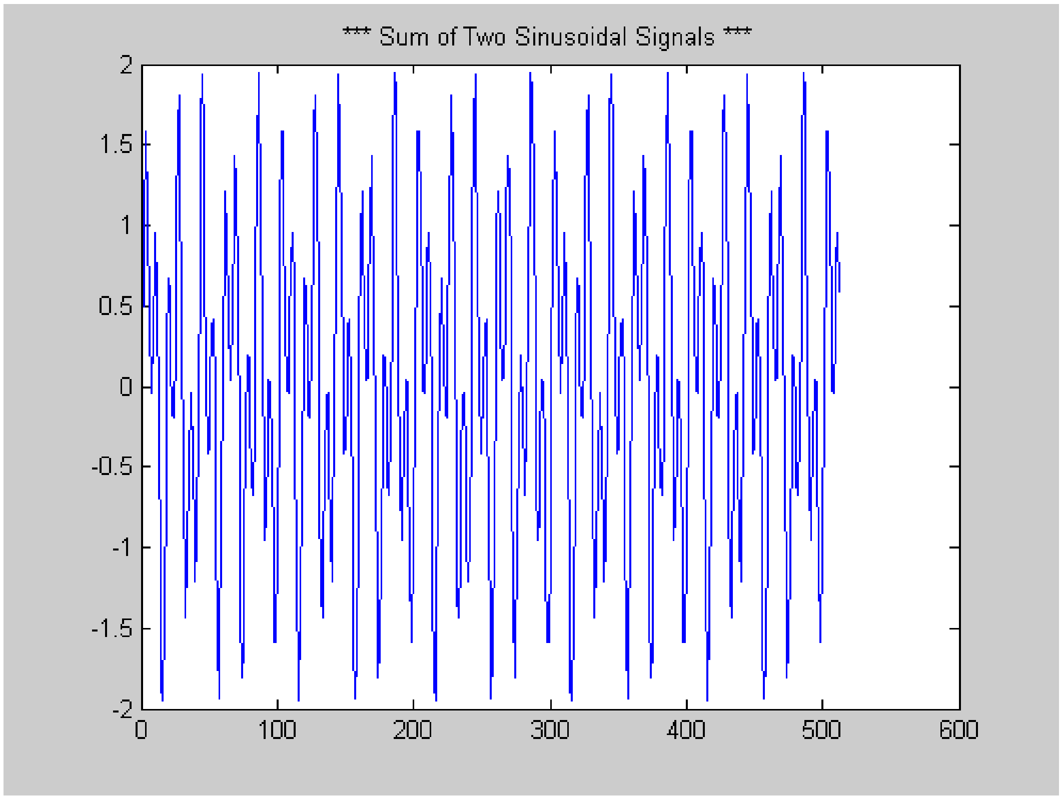
PROGRAM:

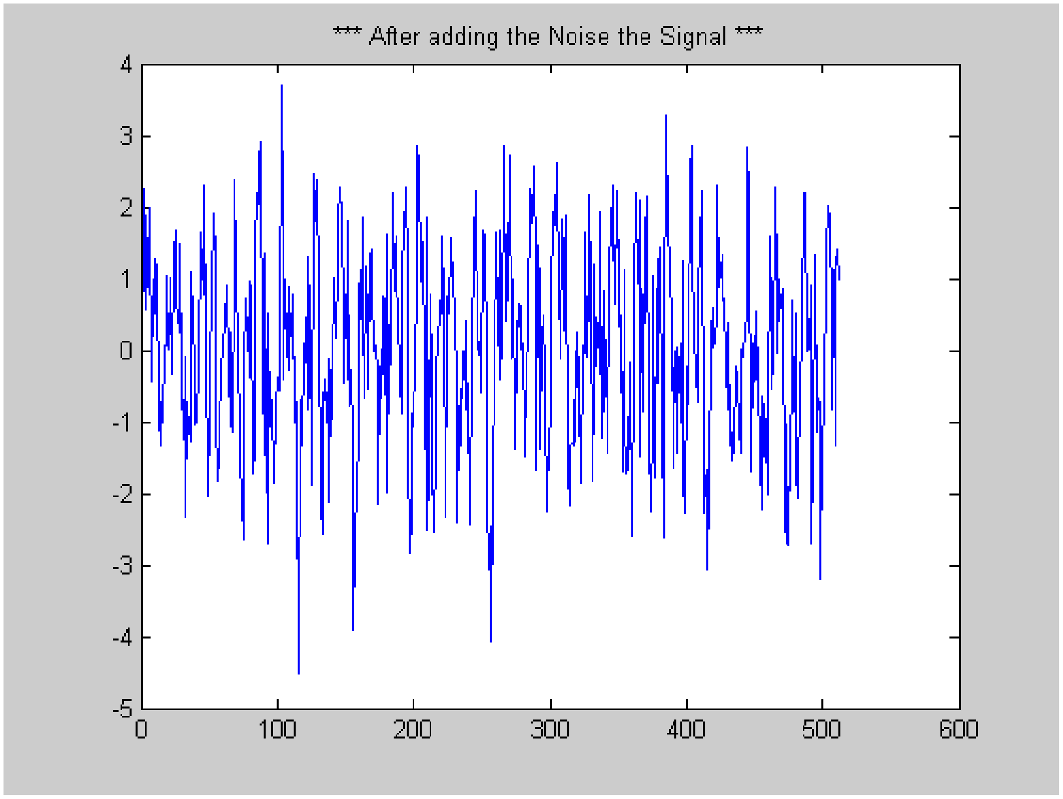
```
clc;

close all;

clear all;

t=0:511;

x1=sin(2*pi*50/1000*t);

figure(1);

%subplot(321);

plot(t,x1);

title('*** Sin Signal 1***');

x2=sin(2*pi*120/1000*t);

figure(2);

%subplot(322);

plot(t,x2);

title('*** Sin Signal 2 ***');

x=x1+x2;

figure(3);
```

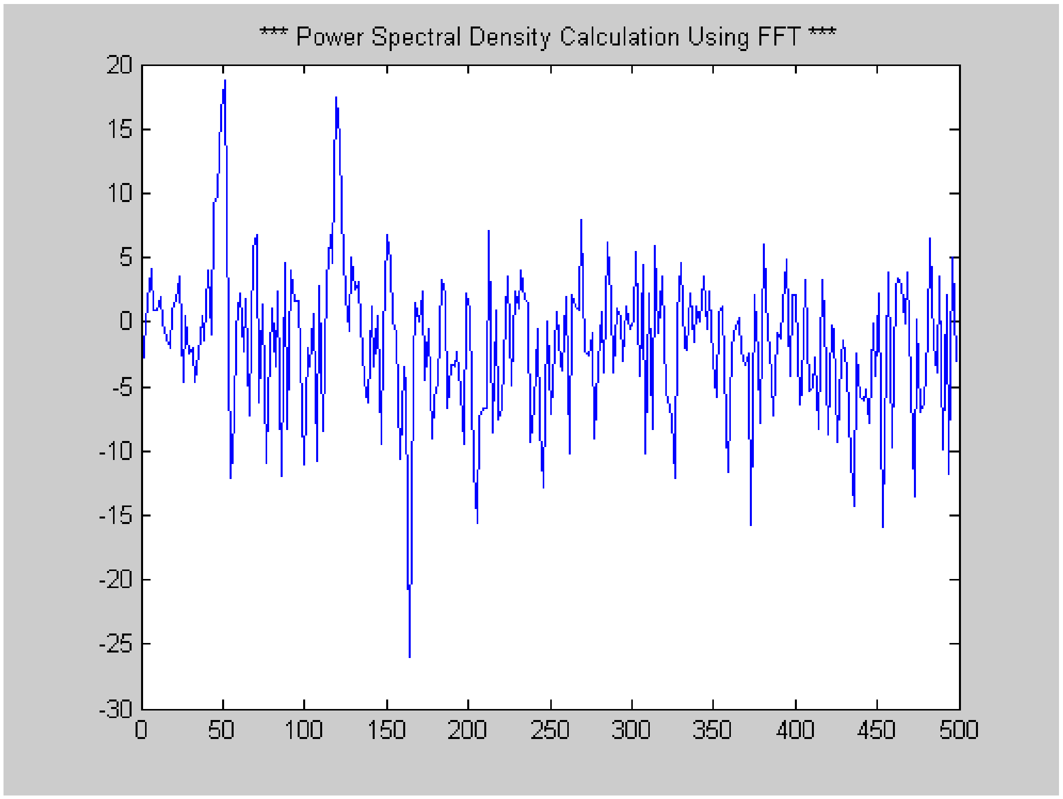
```
%subplot(323);

plot(x);

title('*** Sum of Two Sinusoidal Signals ***');

y=x+randn(size(t));

N=length(y);

figure(4);

%subplot(324);

plot(y);

title('*** After adding the Noise the Signal ***');

f1=1000*(0:N/2-1)/N;

p=(abs(fft(y))).^2/N;

figure(5);

%subplot(325)

plot(f1,10*log10(p(1:256)));

title('*** Power Spectral Density Calculation Using FFT ***');
```




*** Sin Signal 2 ***
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
100
200
300
400
500
600


*** Sum of Two Sinusoidal Signals ***
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
100
200
300
400
500
600





RESULT: The Power Spectral Density of given sequence is observed