

Lab Assignment No. 1**Introduction**

We would have an introduction to MATLAB and get started with working in its wonderfully simple environment. This lab, as a matter of fact, would lay the foundation for our next labs. In the following paragraphs, you are provided with a tutorial on MATLAB. In addition, we'll go over a detailed introduction to MATLAB in our first discussion session. This would enable to you to do the simple but useful things in MATLAB.

MATLAB is the most popular tool used for Digital Signal Processing. It provides one of the strongest environments for study and simulation of the real-world problems and their solutions, especially in the field of engineering. For Signal Processing, it has a very comprehensive and easy-to-use toolbox with lots of DSP functions implemented. Moreover, with the aid of Simulink, we can create much more complex situations very easily, and solve them.

MATLAB utilizes the following arithmetic operators:

+	Addition
-	Subtraction
*	Multiplicatio
/	Division
^	Power operator
'	Transpose

There are several predefined variables which can be used at any time, in the same manner as user-defined variables:

I	sqrt(-1)
J	sqrt(-1)
pi	3.1416...

PRE DEFINED FUNCTIONS IN MATLAB.

There are also a number of predefined functions that can be used when defining a variable. Some common functions that are used in this text are:

abs	magnitude of a number (absolute value for real numbers)
angle	Angle of a complex number, in radians.
Cos	cosine function, assumes argument is in radians
Sin	sine function, assumes argument is in radians

For example, with y defined as above,

```
c = abs(y);  
yields: c = 8.2462;  
c = angle(y);  
yields: c = 1.3258;
```

With a=3 as defined previously,

```
c = cos(a);  
yields: c = -0.9900;  
c = exp(a);  
yields: c = 20.0855;
```

Note that exp can be used on complex numbers. For example, with y = 2+8i as defined above,

```
c = exp(y);  
yields: c = -1.0751 + 7.3104i;  
which can be verified by using Euler's formula.  
c = exp(2)*cos(8) + j*(exp)2*sin(8).
```

DEFINITION OF MATRICES:

MATLAB is based on matrix and vector algebra; even scalars are treated as 1x1 matrices. Therefore, vector and matrix operations are as simple as common calculator operations. Vectors can be defined in two ways. The first method is used for arbitrary elements $v = [1\ 3\ 5\ 7]$; creates a 1x4 vector with elements 1, 3, 5 and 7. Note that commas could have been used in place of spaces to separate the elements. Additional elements can be added to the vector: $v(5)=8$; yields the vector $v = [1\ 3\ 5\ 7\ 8]$. Previously defined vectors can be used to define a new vector.

PLOTTING GRAPHS:**Commands :**

Plot, xlabel, ylabel, title, grid, axis, axes, stem, subplot, zoom, hold The command most often used for plotting is plot , which creates linear plots of vectors and matrices; plot(t,y) plots the vector t on the x-axis versus vector y on the y-axis. There are options on the line type and the color of the plot which are obtained using plot(t,y,'option'). The linetype options are '-' solid line (default), '--' dashed line, '-.' dot dash line, '.' dotted line. The points in y can be left unconnected and delineated by a variety of symbols: + . * o x.

The following colors are available options: r, g, b, k, y, m etc. For example, plot(t,y,'--') uses a dashed line, plot(t,y,'*') uses * at all the points defined in t and y without connecting the points, and plot(t,y,'g') uses a solid green line. The options can also be used together, for example, plot(t,y,'g.') plots a dotted green line. To plot two or more graphs on the same set of axes, use the command plot(t1,y1,t2,y2), which plots y1 versus t1 and y2 versus t2. To label your axes and give the plot a title, type xlabel ('time (sec)') ylabel('step response') title('my plot') Finally, add a grid to your plot to make it easier to read. Type grid .

GENERAL INFORMATION:

- MATLAB is case sensitive so "a" and "A" are two different names.
- Comment statements are preceded by a "%".
- You can make a keyword search by using the help command.
- The number of digits displayed is not related to the accuracy. To change the format of the display, type format short e for scientific notation with 5 decimal places, format long e for scientific notation with 15 significant decimal places and format bank for placing two significant digits to the right of the decimal.
- The commands who and whos give the names of the variables that have been defined in the workspace.
- The command length(x) returns the length of a vector x and size(x) returns the dimension of the matrix x.

Lab Assignment No. 2**Introduction to matrices and basic signals****DEFINITION OF MATRICES:**

MATLAB is based on matrix and vector algebra; even scalars are treated as 1x1 matrices. Therefore, vector and matrix operations are as simple as common calculator operations. Vectors can be defined in two ways. The first method is used for arbitrary elements

$$v = [1 \ 3 \ 5 \ 7];$$

creates a 1x4 vector with elements 1, 3, 5 and 7. Note that commas could have been used in place of spaces to separate the elements. Additional elements can be added to the vector:

$$v(5)=8;$$

yields the vector $v = [1 \ 3 \ 5 \ 7 \ 8]$.

Previously defined vectors can be used to define a new vector.

For example, with 'v'

$$a = [9 \ 10];$$

$$b = [v \ a];$$

creates the vector

$$b = [1 \ 3 \ 5 \ 7 \ 8 \ 9 \ 10].$$

The second method is used for creating vectors with equally spaced elements.

$$t = 0:0.1:10;$$

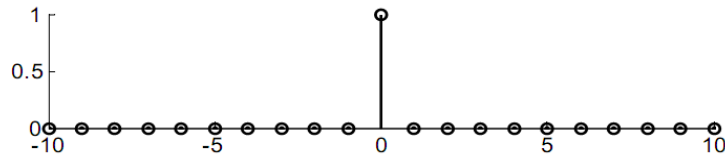
creates a 1x101 vector with the elements 0, .1, .2, .3,...,10. Note that the middle number defines the increment. If only two numbers are given, then the increment is set to a default of 1

$$k = 0:10;$$

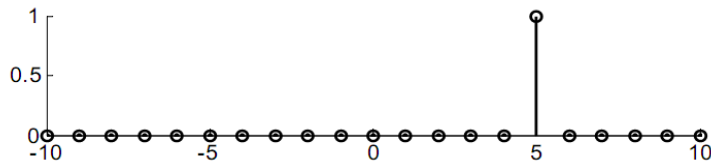
creates a 1x11 vector with the elements 0, 1, 2, ..., 10.

UNIT IMPULSE:

$$s[n] = 1 \quad \text{if } n=0$$
$$\text{and } 0 \quad \text{other wise.}$$

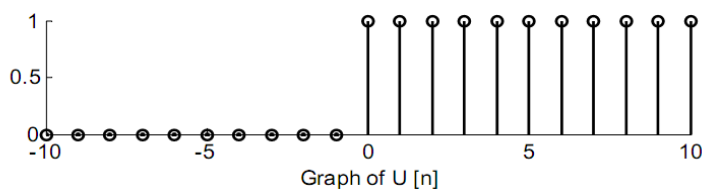


$$s[n-n_0]=1 \quad \text{if } n= n_0$$
$$\text{and } 0 \quad \text{other wise.}$$

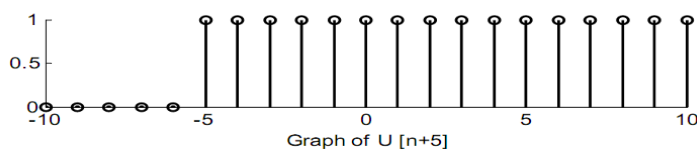


UNIT STEP :

$$U [n] = 1 \quad n \geq 0$$
$$0 \quad \text{otherwise}$$



$$U [n - n_0] = 1 \quad n \geq n_0$$
$$0 \quad \text{otherwise}$$



Program

```
>> a=2;  
b=3;  
c=a+b  
c = 5
```

Program

```
>> c=a*b  
c = 6
```

Program

```
>> c=a/b  
c = 0.6667
```

Program

```
>> c=a^2  
c = 4
```

Program

```
>> y=2*(1+4*j);  
>> abs(y)  
ans = 8.2462
```

Program

```
>> angle(y)  
ans = 1.3258
```

Program

```
>> a=[1 2 3]
a = 1  2  3
```

```
>> b=[3 4 5]
b = 3  4  5
>> c=a+b
c = 4  6  8
```

Program

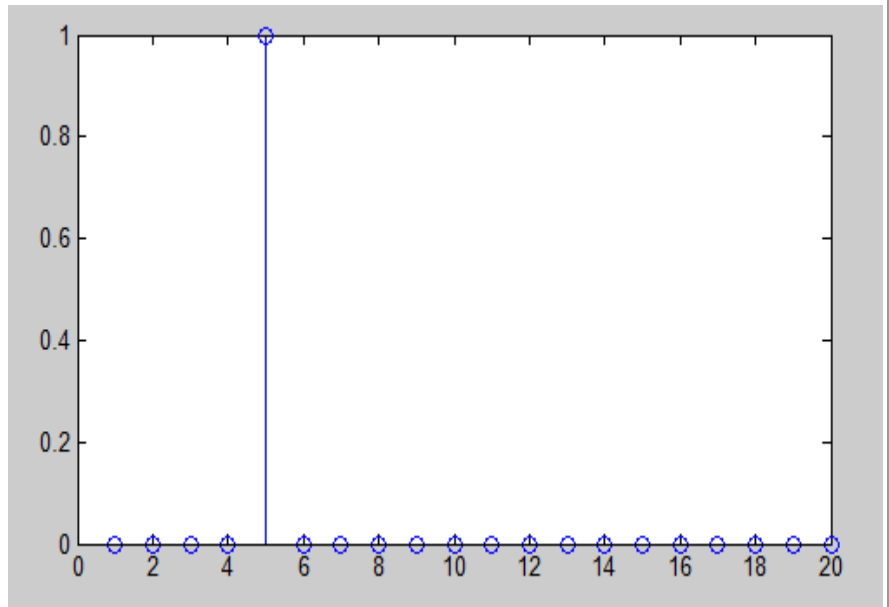
```
>> a=[1 2 3;4 5 6;7 8 9]
a =1  2  3
    4  5  6
    7  8  9
>> b=[4 5 6;7 8 9;1 2 3]
b =4  5  6
    7  8  9
    1  2  3
>> d=a*b
d=21  27  33
    57  72  87
    93 117 141
```


Lab Assignment No: 3

Basic signals & system

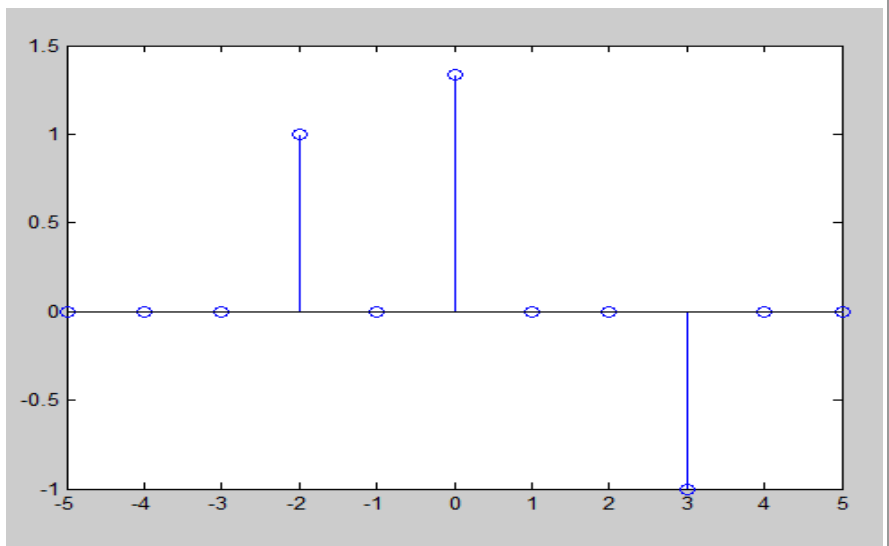
Code :

```
clc  
clear all  
t=1:20  
x(1,20)=0  
x(1,5)=1  
stem(t,x)  
end
```

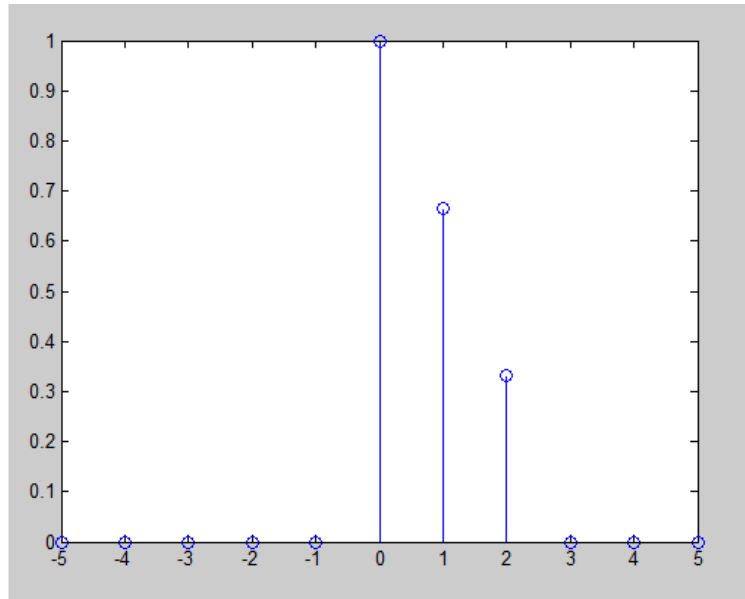


Problem:

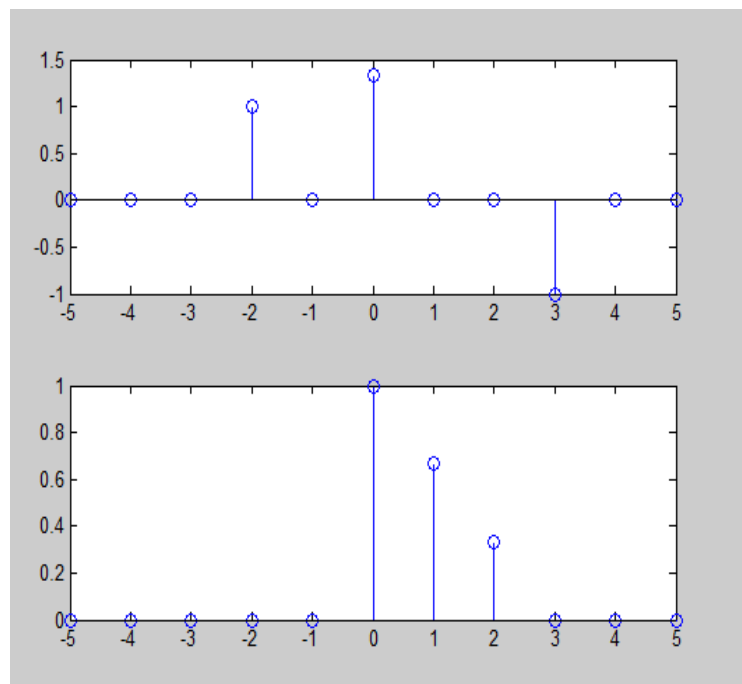
```
clc;  
clearall;  
n=-5:5  
x=[0 0 0 1 0 4/3 0 0 -1 0 0]  
stem(n,x)  
end
```



```
clc;  
clearall;  
n=-5:5  
x=[0 0 0 0 0 1 2/3 1/3 0 0 0]  
stem(n,x)  
end
```



```
clc;  
clearall;  
n=-5:5  
x=[0 0 0 1 0 4/3 0 0 -1 0 0]  
subplot(211)  
stem(n,x)  
  
n1=-5:5  
x1=[0 0 0 0 0 1 2/3 1/3 0 0 0]  
subplot(212)  
stem(n1,x1)  
end
```



PROBLEM :

Generate and plot

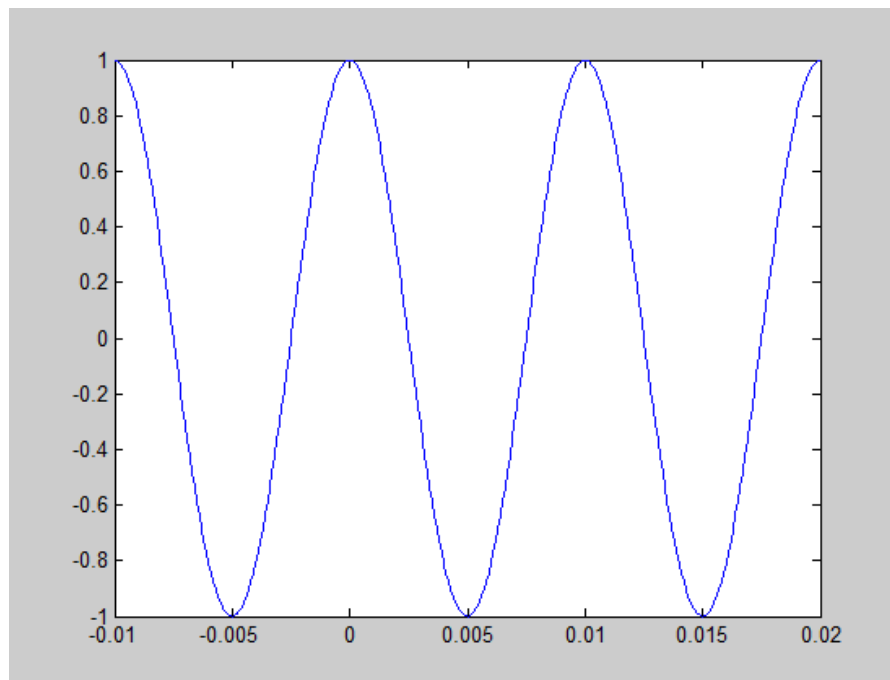
$$x = \cos(w*t)$$

Theory :

The above function generates and plot a wave of the described parameter give in the below code are various parameter on which it depends.

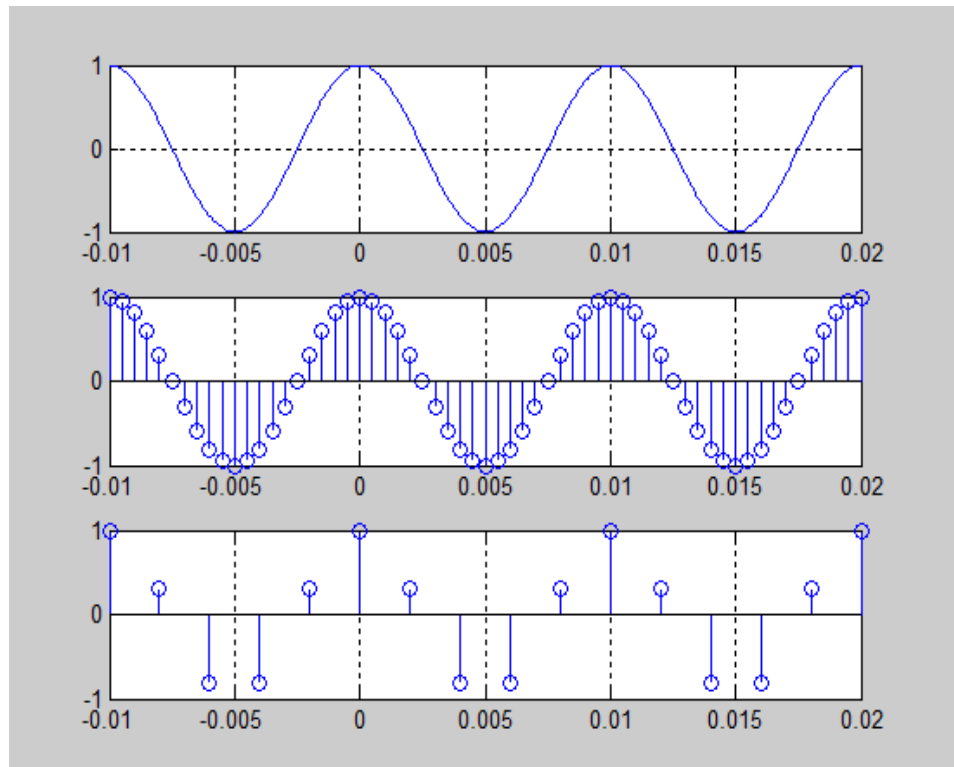
Code:

```
clc;
clearall;
f=100
w=2*pi*f
t=-0.01:0.0001:0.02
x=cos(w*t)
plot(t,x)
end
```



Code:

```
clc;
clearall;
fs=2000;
f=100
w=2*pi*f
t=-0.01:0.0001:0.02
x=cos(w*t)
subplot(311)
plot(t,x)
grid
f1=100
ts=1/fs
w=2*pi*f1
t1=-0.01:ts:0.02
x2=cos(w*t1)
subplot(312)
stem(t1,x2)
grid
f2=100
fs2=500
ts1=1/fs2
w=2*pi*f2
t2=-0.01:ts1:0.02
x3=cos(w*t2)
subplot(313)
stem(t2,x3)
grid
```



Lab Assignment No. 4**SAMPLING AND ALIASING****PERIODIC SIGNALS:**

A sequence $X[n]$ is said to be periodic when it satisfies the following relation

$$X[n] = X[n + N]$$

The fundamental period of the signal is defined as the smallest positive value of 'N' for which above equation holds true.

Now consider a signal $Y[n]$ which is formed by the addition of the two signals $x_1[n]$ and $x_2[n]$.

$$Y[n] = X_1[n] + X_2[n]$$

If the signal $x_1[n]$ is periodic with the fundamental period 'N1' and $x_2[n]$ is periodic with the fundamental period 'N2', then the signal $y[n]$ will be periodic with the period 'N' given by the following relation

$$N = (N_1 \times N_2) / \text{gcd}(N_1, N_2)$$

THE SAMPLING THEOREM:

A continuous time signal $x(t)$ with frequencies no higher than F_{\max} can be reconstructed exactly from its samples $x[n] = x(n T_s)$, if the samples are taken at a rate $F_s = 1 / T_s$ that is greater than $2F_{\max}$ "

$$F_s \geq 2F_{\max}$$

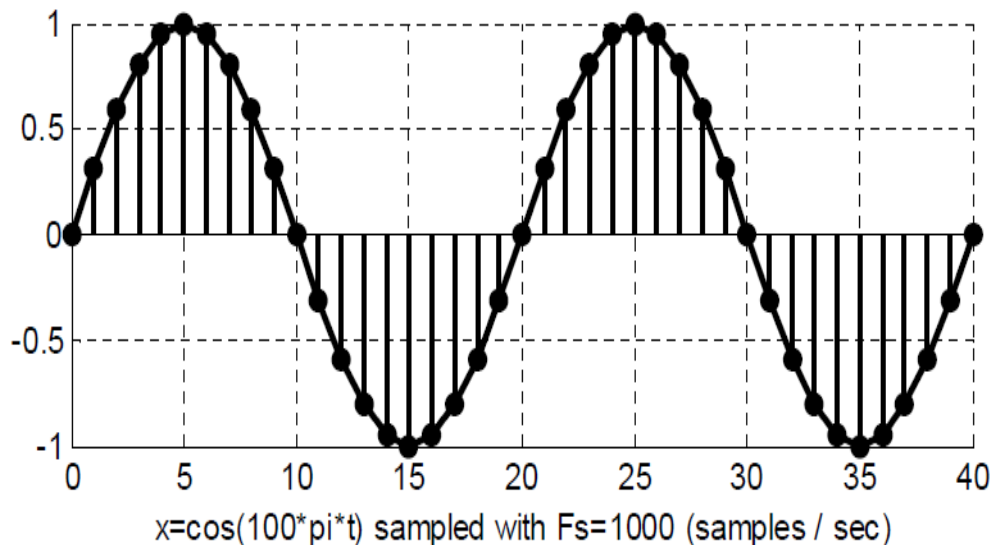
The minimum sampling rate of $2F_{\max}$ is called the Nyquist Rate. From Sampling theorem it follows that the reconstruction of a sinusoid is possible if we have at least 2 samples per period. If we don't sample at a rate that satisfies the sampling theorem then aliasing occurs.

SAMPLING A CONTINUOUS TIME SIGNAL:

For example we want to sample continuous time signal $x = \cos(100 \pi t)$. The frequency of this signal is 50 Hz and according to sampling theorem the

minimum sampling frequency should be 100 samples / sec .But we sample it at a rate much higher than the Nyquist rate so that it has many samples over one cycle giving an accurate representation of the sampled discrete time signal.

In the figure below the continuous time signal $x = \cos (100 \pi t)$ is sampled at $F_s=1000$ (samples / second).Therefore it will have $F_s / F = 1000 / 50 = 20$ (samples per cycle)

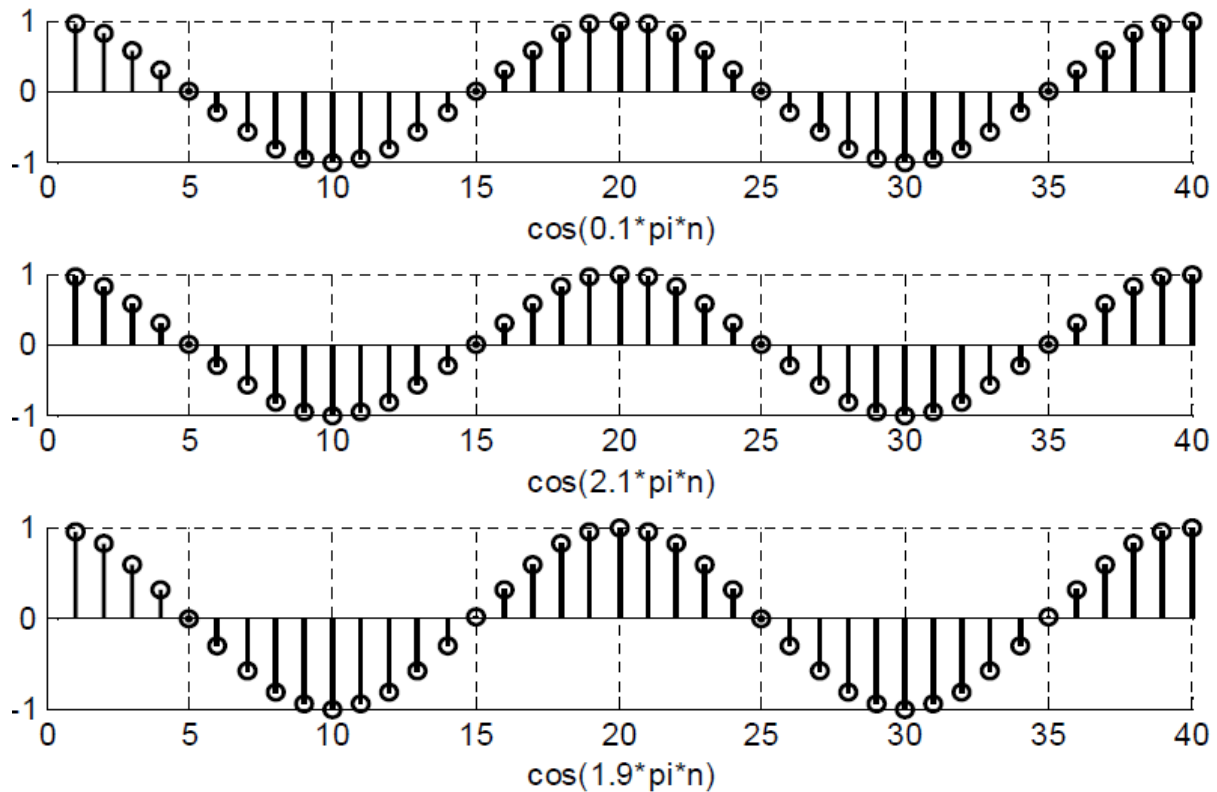


CONCEPT OF ALIASING:

Consider the general formula for a discrete time sinusoid $X = \cos (\hat{\omega} \pi n + \Phi)$.

Now consider

$x_1 = \cos (0.1 \pi n)$, $x_2 = \cos (2.1 \pi n)$ and $x_3 = \cos (1.9 \pi n)$ apparently with different values of “ $\hat{\omega}$ ”. If we display the graph of these 3 signals over the range $n = 0:40$ then we see that all the above 3 signals are equivalent.



Therefore the signals x_1 , x_2 and x_3 are different names for the same signal. This phenomenon is called aliasing. The signals x_2 and x_3 are called aliases of the signal x_1 . Coding for plotting the above 3 signals is shown below

Coding:

```
n=1:40;
x1=cos(0.1*pi*n);
x2=cos(2.1*pi*n);
x3=cos(1.9*pi*n);
subplot(311);stem(n,x1);
grid;
xlabel('cos(0.1*pi*n)');
subplot(312);
stem(n,x2);
grid;
xlabel('cos(2.1*pi*n)');
subplot(313);
stem(n,x3);
```

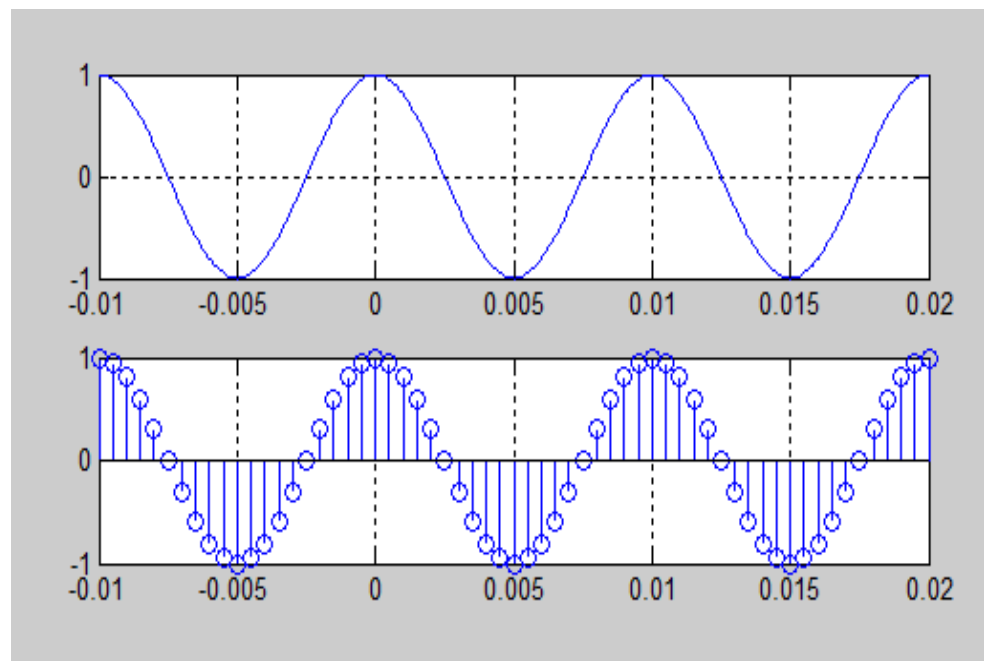
```
grid;  
xlabel('cos(1.9*pi*n)');
```

problem :

In this program what we require is to generate a signal with frequency hz, and then we are to sample this signal such that are same samples in each cycle and there are a total of same cycle in the figure as shown in the figure below

Code:

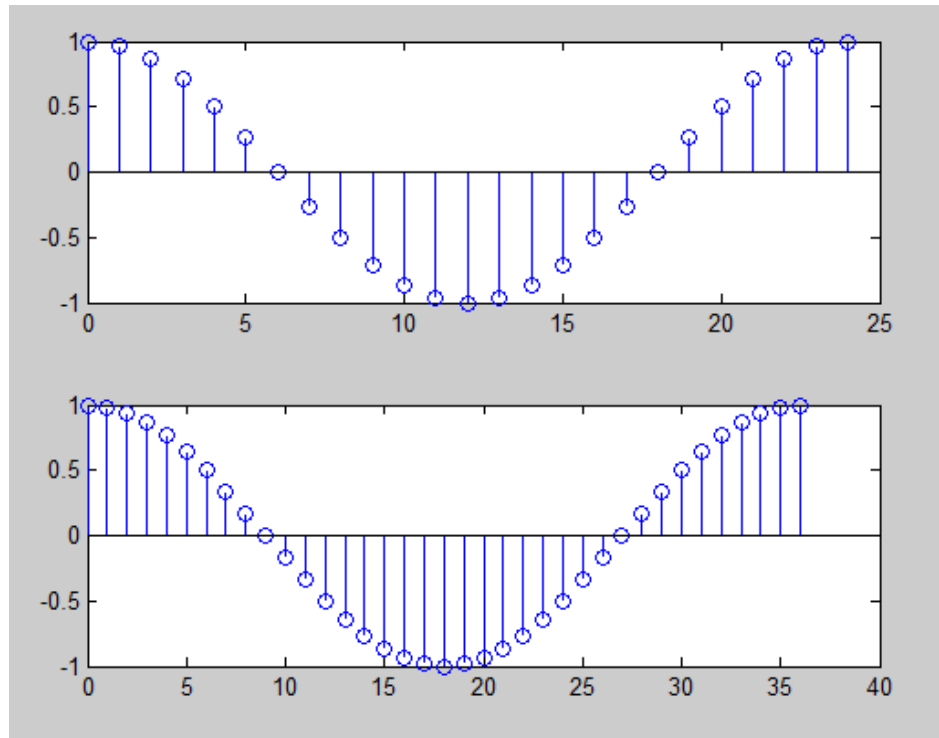
```
clc;  
clearall;  
fs=2000;  
f=100  
w=2*pi*f  
t=-0.01:0.0001:0.02  
  
x=cos(w*t)  
subplot(311)  
plot(t,x)  
grid  
f1=100  
ts=1/fs  
w=2*pi*f1  
t1=-0.01:ts:0.02  
x2=cos(w*t1)  
subplot(312)  
stem(t1,x2)  
grid
```



Problem :

```

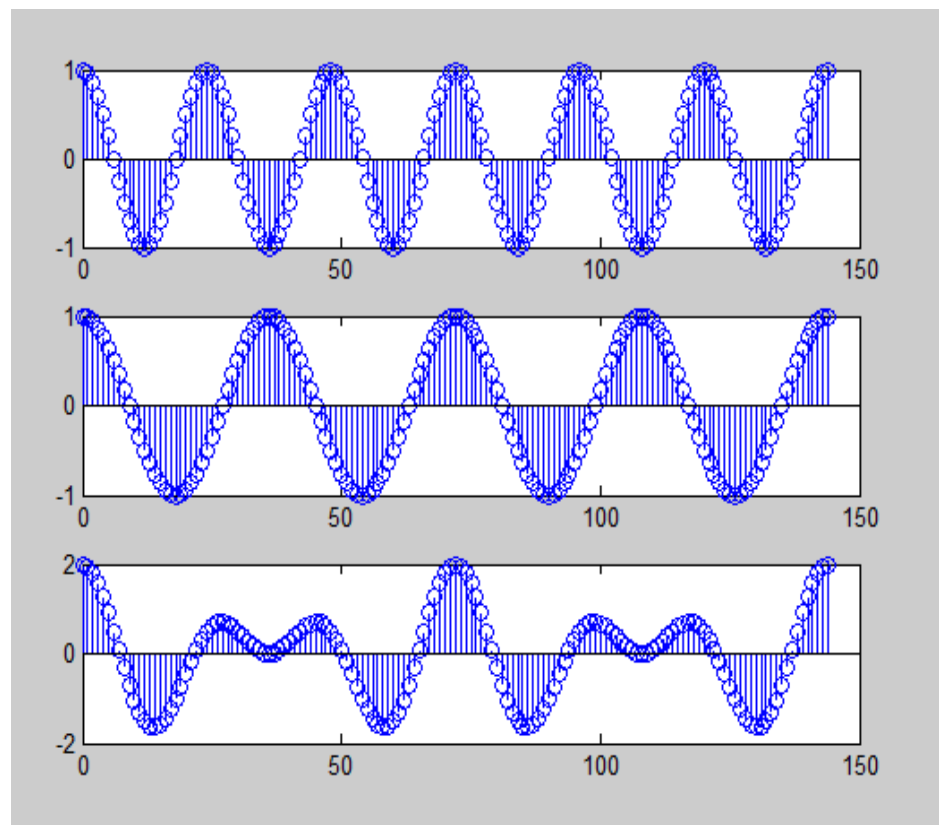
clc;
clearall;
n=0:24
x1=cos(n*pi/12)
subplot(211)
stem(n,x1)
n1=0:36
x2=cos(n1*pi/18)
subplot(212)
stem(n1,x2)
    
```



PROBLEM :

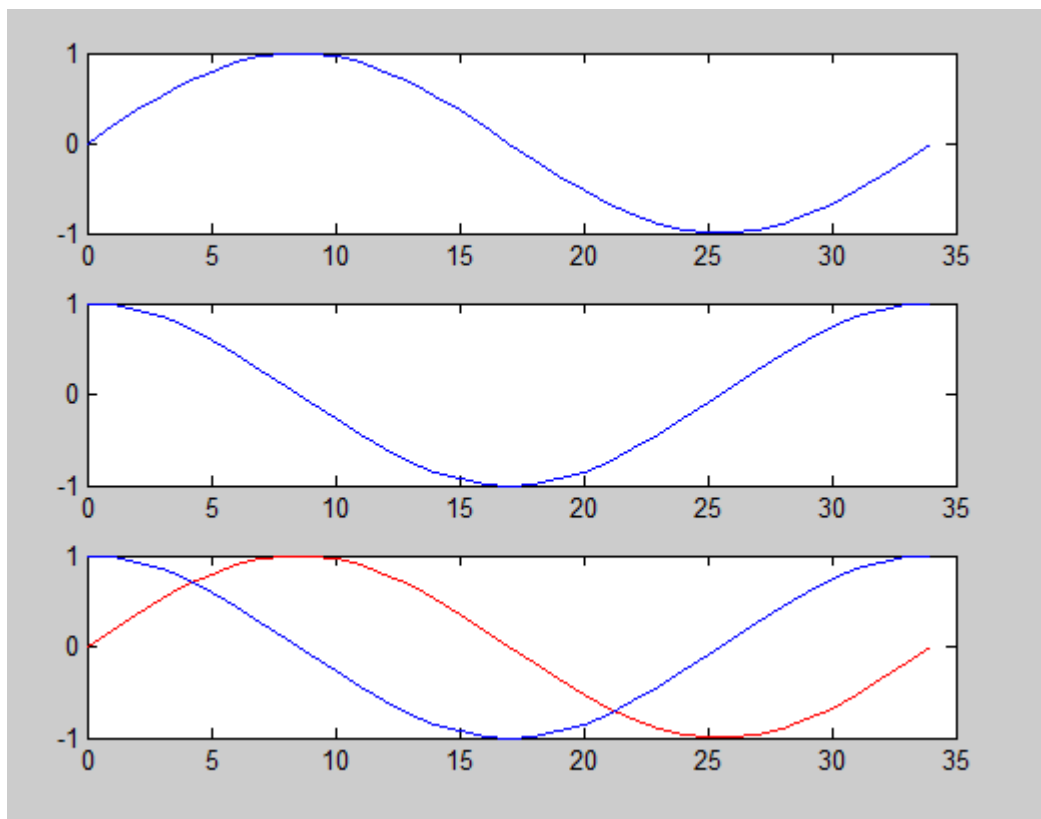
```

clc;
clearall;
n=0:144
x1=cos(n*pi/12)
subplot(311)
stem(n,x1)
n1=0:144
x2=cos(n1*pi/18)
subplot(312)
stem(n1,x2)
n3=0:144
x3=x1+x2
subplot(313)
stem(n3,x3)
    
```



Problem.

```
clc;  
clearall;  
n=0:34  
x=sin((n*pi)/17);  
subplot(3 1 1)  
plot(n,x);  
x2=sin((n*pi)/17+pi/2 );  
subplot(3 1 2)  
plot(n,x2);  
subplot(3 1 3);  
plot(n,x,'r' ,n,x2,'b')
```



Lab Assignment No. 5

Convolution

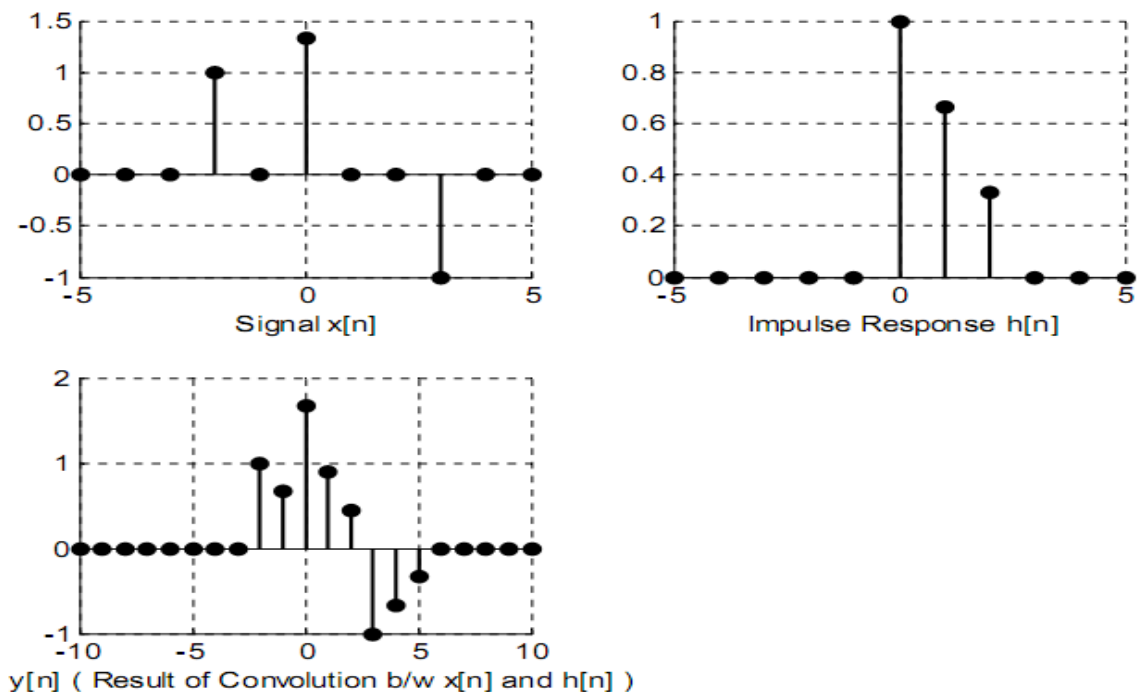
A linear time invariant (LTI) system is completely characterized by its impulse response $h[n]$. “We can compute the output $y[n]$ due to any input $x[n]$ if know the impulse response of that system”. We say that $y[n]$ is the convolution of $x[n]$ with $h[n]$ and represent this by notation

$$y[n] = x[n] * h[n] = \sum_{k=-8}^{k=+8} x[k] h[n-k] \quad \text{Equation A}$$

$$= h[n] * x[n] = \sum_{k=-8}^{k=+8} h[k] x[n-k]$$

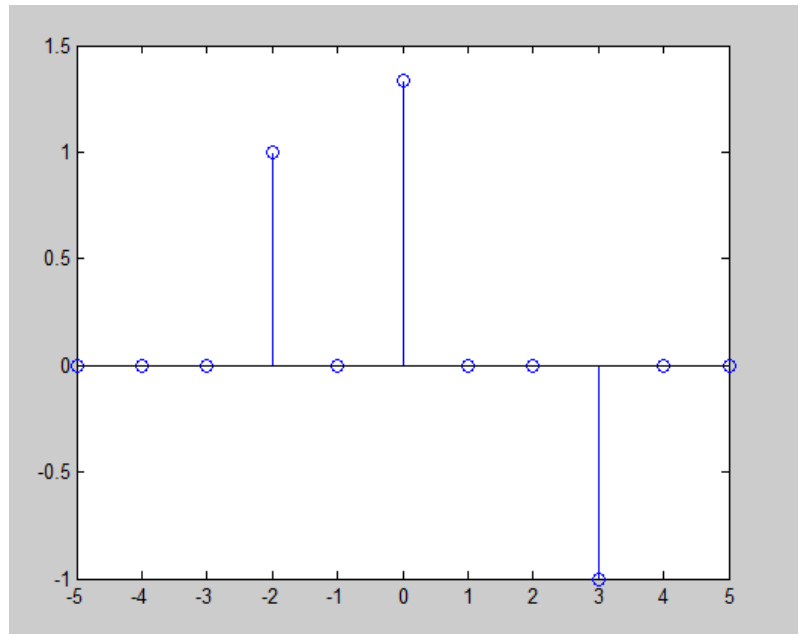
CONVOLUTION IMPLEMENTATION IN MATLAB :

In Matlab “CONV” function is used for “convolution and polynomial multiplication”. $y = \text{conv}(x, h)$ convolves vectors x and h . The resulting vector ‘ y ’ is of “ $\text{length}(A)+\text{length}(B)-1$ ” If ‘ x ’ and ‘ h ’ are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials



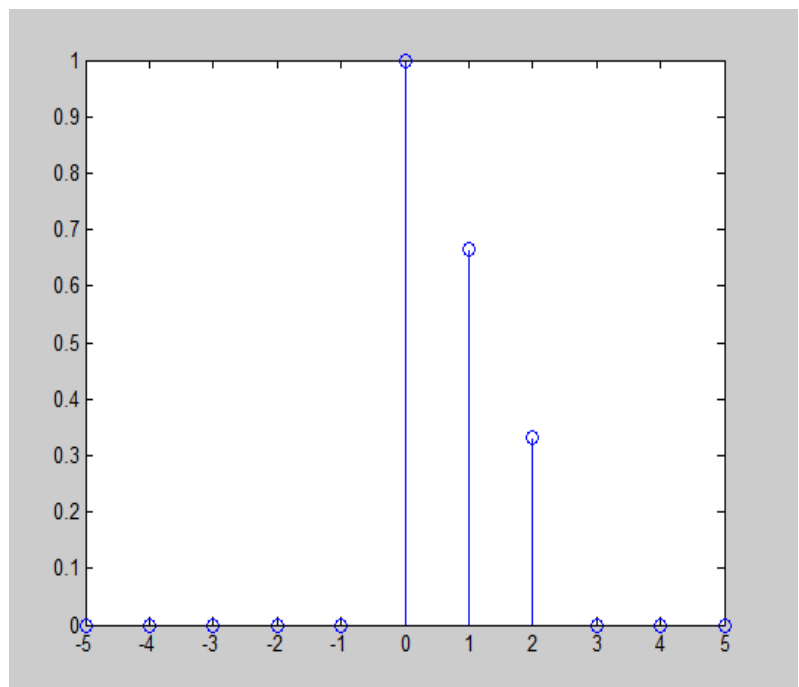
Program :

```
a=-5;  
b=1;  
c=5;  
n=-5:1:5  
x=[0 0 0 1 0 4/3 0 0 -1 0 0]  
stem(n,x)
```



Program :

```
clc;  
clear all;  
n=-5:1:5;  
x=[0 0 0 0 0 1 2/3 1/3 0 0 0];  
stem(n,x)
```

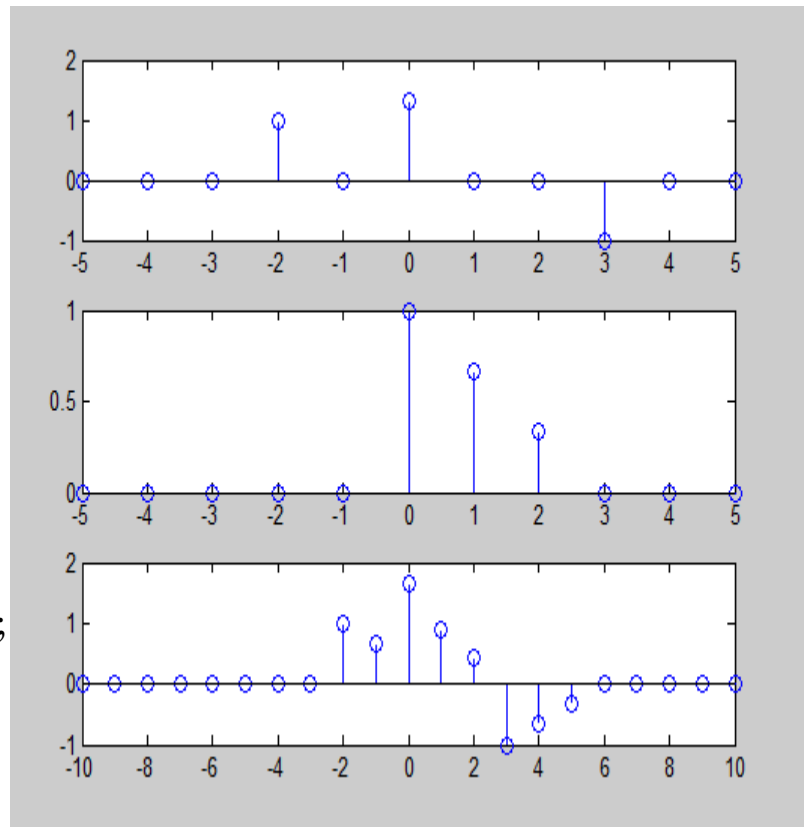


Program :

```

clc;
clear all;
a=-5;
b=1;
c=5;
n=-5:1:5
x=[0 0 0 1 0 4/3 0 0 -1 0 0]
subplot(311)
stem(n,x)
y=[0 0 0 0 0 1 2/3 1/3 0 0 0];
subplot(312)
stem(n,y)
d=-10:1:10;
h=conv(x,y);
subplot(313)
stem(d,h)

```

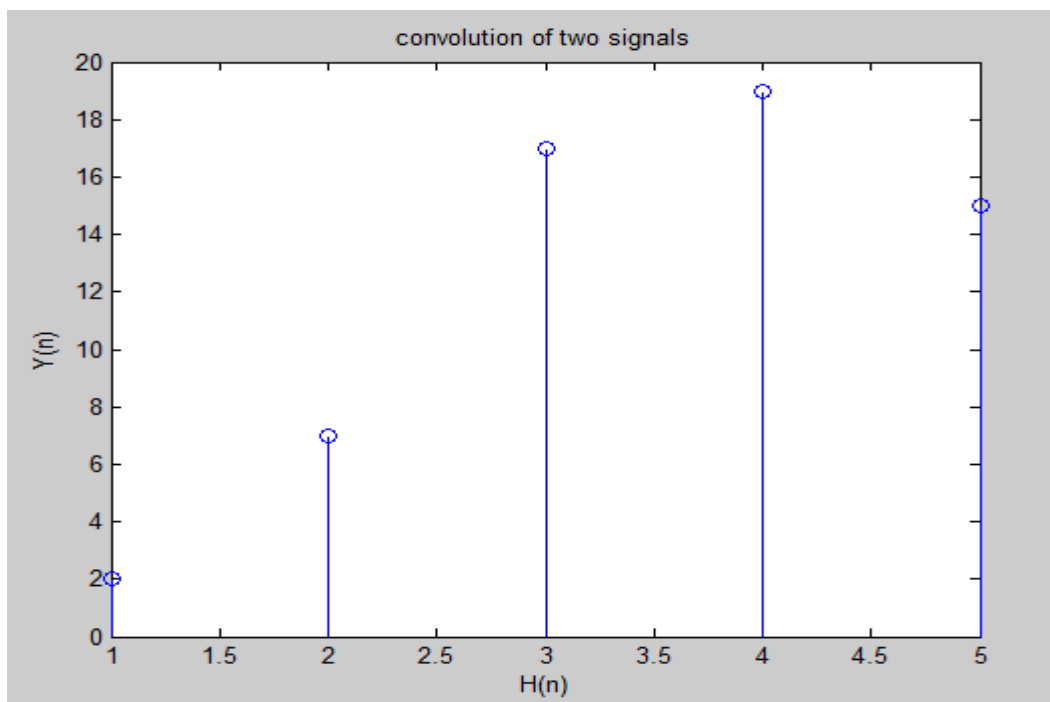
**Program :**

```

clc;
clear all;
x=input('enter the value of x(n):')
h=input('enter the value of h(n):')
n1=length(x);
n2=length(h);
X=[x,zeros(1,n2)];
H=[h,zeros(1,n1)];
for i=1:n1+n2-1;
    Y(i)=0;
    for j=1:n1;

```

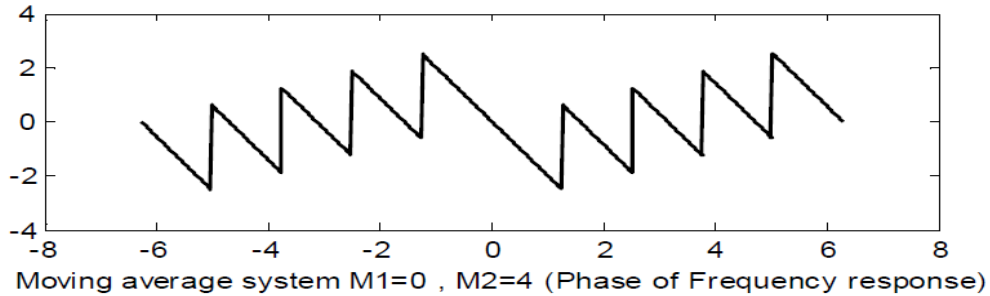
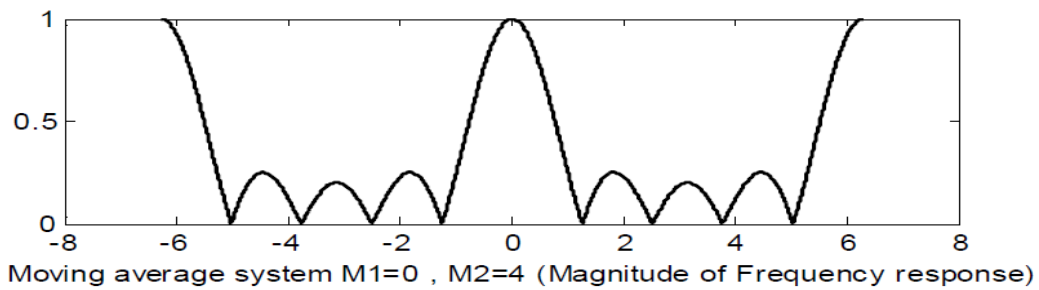
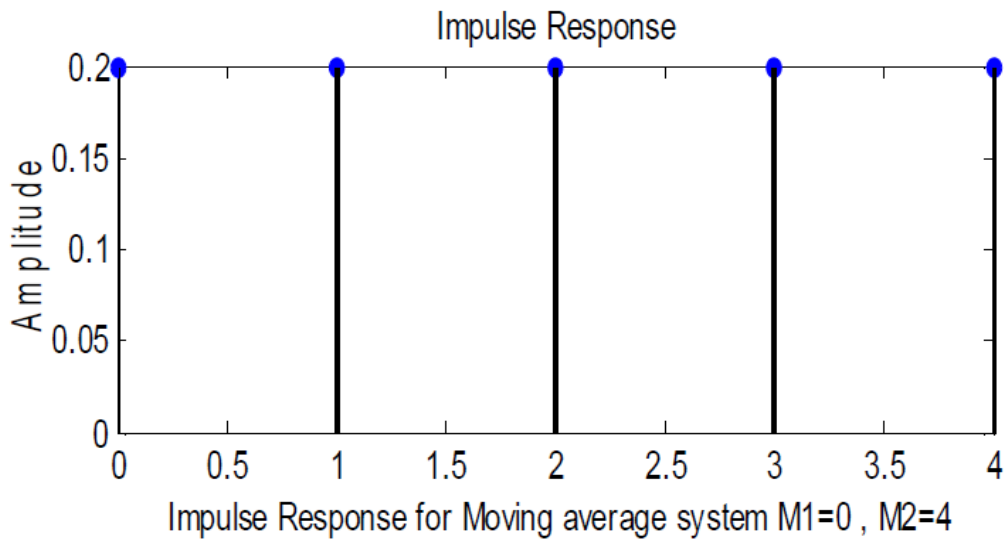
```
if(i-j+1>0)
    Y(i)=Y(i)+X(j) * H(i-j+1);
else
    end
end
end
stem(Y)
ylabel('Y(n)');
xlabel ('H(n)');
title('convolution of two signals');
when X(n) =[1 2 3]
h(n)=[2 3 5]
```

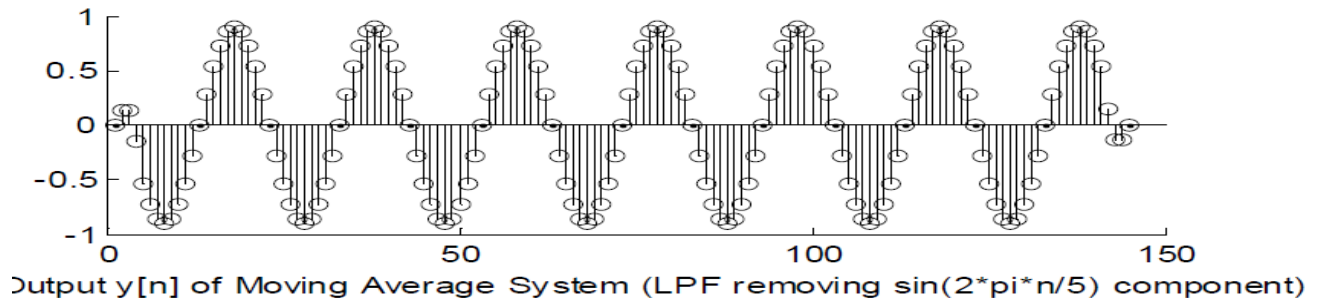
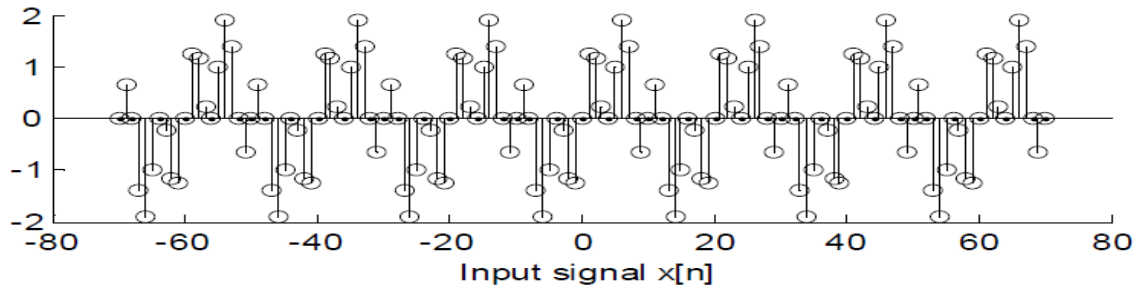


Lab Assignment No. 6**Moving Average System acts as Low Pass Filter.****Code:**

```
close all;
n=-70:70;
xn=sin(2*pi*n/5)+sin(0.1*pi*n);
figure;
subplot(2 1 1);
stem(n,xn);
xlabel('Input signal x[n]');
hn=(1/5)*ones(1,5);
yn=conv(xn,hn);
subplot(2 1 2);
stem(yn);
xlabel('Output y[n] of Moving Average System (LPF removing sin(2*pi*n/5)
component)');
b=(1/5)*ones(1,5);
a=1;
w=-2*pi:0.01:2*pi;
figure;
impz(b,a);
xlabel('Impulse Response for Moving average system M1=0 , M2=4');
figure;
H=freqz(b,a,w);
subplot(2,1,1);
plot(w,abs(H));
xlabel('Moving average system M1=0 , M2=4 (Magnitude of Frequency
response)');
subplot(2,1,2);
plot(w,angle(H));
xlabel('Moving average system M1=0 , M2=4 (Phase of Frequency response)');
```

Result.





Lab Assignment No: 7

Z-transform

Z-TRANSFORM :

The Fourier Transform of a sequence $x[n]$ is defined as

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n} \quad \rightarrow \text{Equation A}$$

The Z Transform of a sequence $x[n]$ is defined as

$$X(z) = \sum_{n=-\infty}^{+\infty} x[n] z^{-n} \quad \rightarrow \text{Equation B}$$

The Fourier Transform of a sequence $x[n]$ is defined as $\sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n}$

$$X(e^{j\omega}) = \sum_{n=-\infty}^{+\infty} x[n] e^{-j\omega n}$$

The Z Transform of a sequence $x[n]$ is defined as $\sum_{n=-\infty}^{+\infty} x[n] z^{-n}$

$$X(z) = \sum_{n=-\infty}^{+\infty} x[n] z^{-n}$$

Therefore the Fourier Transform is $X(z)$ with $z = e^{j\omega}$

Matlab provides a function “ztrans” for evaluating the Z Transform of a symbolic expression. This is demonstrated by the following Example # 1 which uses Equation C to evaluate Z Transform given by Equation D

Problem.

In this we are to evaluate the z transform of a^n by using the above formulae it can be calculated easily and it is given below by using the matlab commands syms, ztrans, pretty.

Code:

```

x(z)=summition x(n)z^-n
clc
clear all
syms a n z;
X(n)=a^n
Xn=a^n(n)
Xz=z tran(Xn);
Disp('Xn='); pretty(Xn)
Disp('Xz='); pretty(Xz)

```

ANSWER

```

xn = a^n
xz=      z
      - ----      =-z/a-z =z/a(z(a-1))
      a -z

```

Problem.

```

clc;
clearall;
symsanzu;
xn=(1/2).^n*u+(-1/3).^n*u
xz=ztrans(xn);
disp('xn='); pretty(xn);
disp('xz='); pretty(xz);

```

Answer

```

xn = (1/2)^n*u + (-1/3)^n*u
xn=nn (1/2) u + (-1/3) u
xz=u z      u z ----- + -----
z - 1/2  z + 1/3

```

PROBLEM:

```
clc;
clearall;
symsanz ;
xz=1/1-a*z^-1
xn=iztrans(xz);
disp('xn='); pretty(xn);
disp('xz='); pretty(xz);
```

Answer

```
xz = 1 - a/z
xn=kroneckerDelta(n, 0) - a kroneckerDelta(n - 1, 0)
xz=a 1 - - z
```

Problem.

```
clc;
clearall;
symsanz ;
xz=1/(1-(1/4)*z^-1)*(1-(1/2)*z^-1)
xn=iztrans(xz);
disp('xn='); pretty(xn);
disp('xz='); pretty(xz); clc;
clearall;
symsanz ;
xz=1/(1-(1/4)*z^-1)*(1-(1/2)*z^-1)
xn=iztrans(xz);
disp('xn='); pretty(xn);
disp('xz='); pretty(xz);
```

Answer

$$XZ = (1/(2*z) - 1)/(1/(4*z) - 1)$$

$$x_n = n^2 \text{kroneckerDelta}(n, 0) - (1/4)$$

$$XZ = \frac{1 - \frac{1}{2}z^{-1}}{1 - \frac{1}{4}z^{-1}}$$

$$\frac{1 - \frac{1}{2}z^{-1}}{1 - \frac{1}{4}z^{-1}}$$

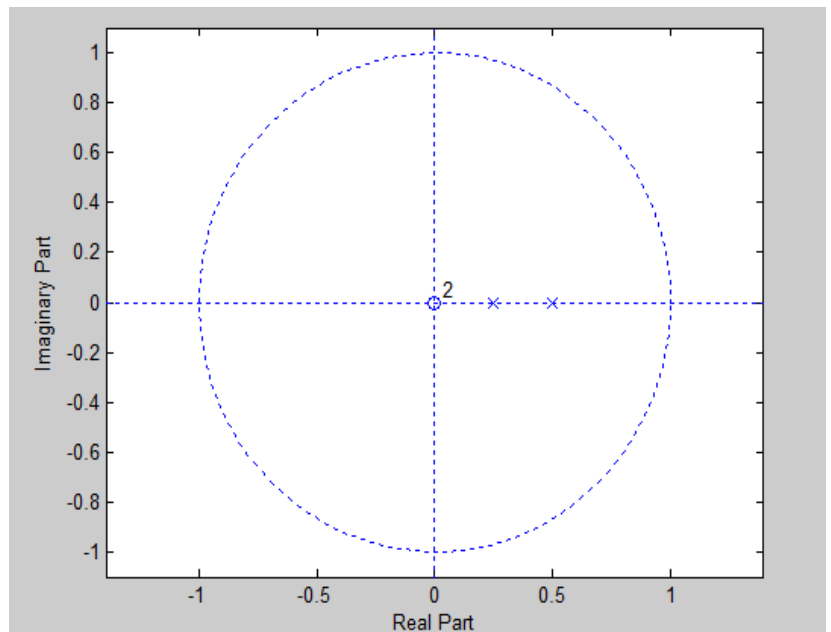
Zero Pole Plot:

In the problem given below to have an understanding of zero pole plot, we use these methods first is to given b & a second is to used convolution and the third is through use of poly.

Code:

```

clc;
clearall;
symsanz ;
num=[1]
den=[1 -3/4 1/8]
xz=1/1-(3/4)*z^-1+(1/8)*z^-1
zplane(num, den)
[z p k]=tf2zp(num,den)
    
```



Answer

$$XZ = 1 - \frac{5}{8}z^{-1}$$

z(zero) = Empty matrix: 0-by-1

p(pole) = 0.5000

0.2500

k(gain)= 1

Lab Assignment No: 8**DISCRETE FOURIER TRANSFORM:****DISCRETE TIME FOURIER TRANSFORM (DTFT)**

DTFT of a signal $x[n]$ can be defined as:

$$X(\omega) = \sum_{n=0}^{N-1} x[n] e^{-j\omega n}$$

It gives frequency spectrum of the signal. Since ω is continuous, its range is continuous. So it is impossible to get a vector in MATLAB which covers entire range of the DTFT.

DISCRETE FOURIER TRANSFORM (DFT)

N point DFT of a signal $x[n]$ is defined as:

$$X(K) = \sum_{n=0}^{N-1} x[n] e^{-j(2\pi/N)kn}$$

and IDFT can be obtained by the relation:

$$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{-j(2\pi/N)kn}$$

N point DFT means taking N equidistant samples (Sampling in frequency domain). DFT is a sampled version of DTFT.

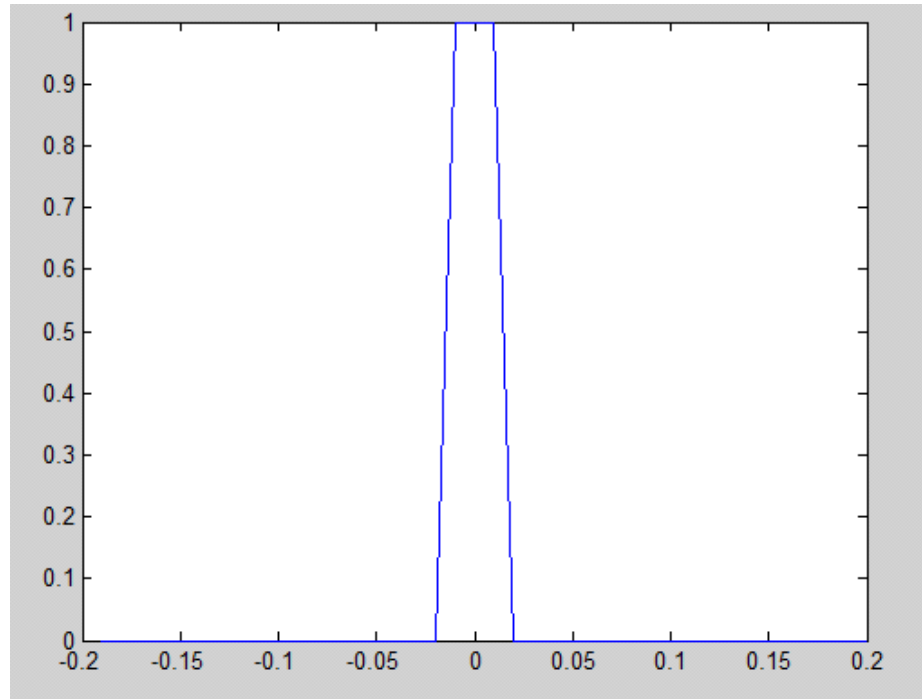
Now you are dividing 2π in N samples.

$$\text{Width} = \frac{2\pi}{N}$$

In order to estimate DTFT of the signal, we may take DFT of the signal, while taking large number of DFT points.

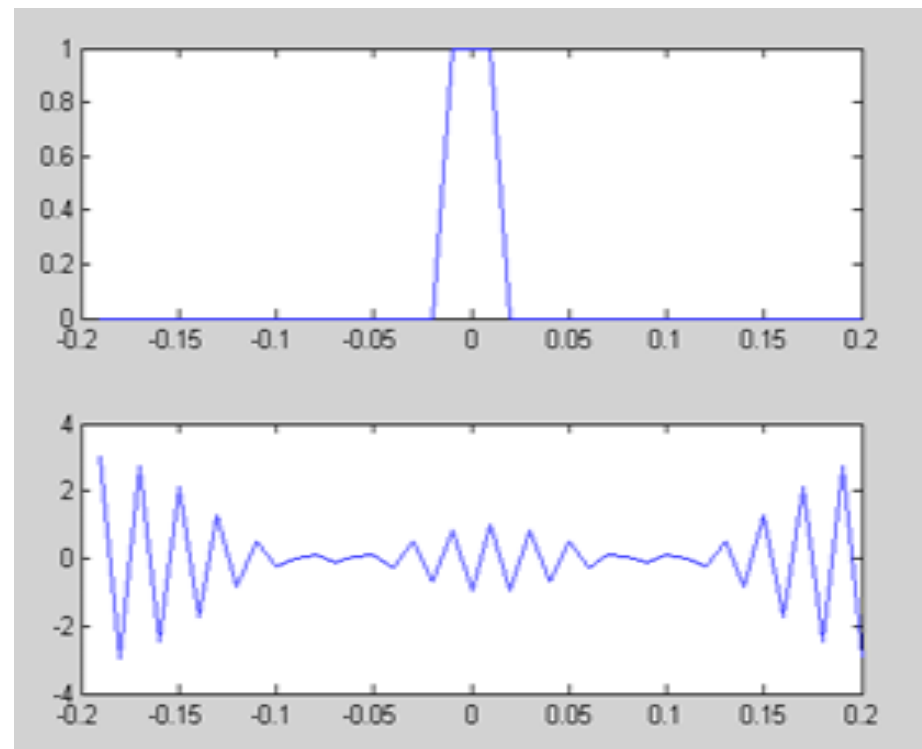
Program

```
clc  
clear all  
clf  
inc=0.01  
t=-0.2+inc:inc:0.2  
w=0.025  
ra=rectpuls(t,w)  
plot(t,ra)
```



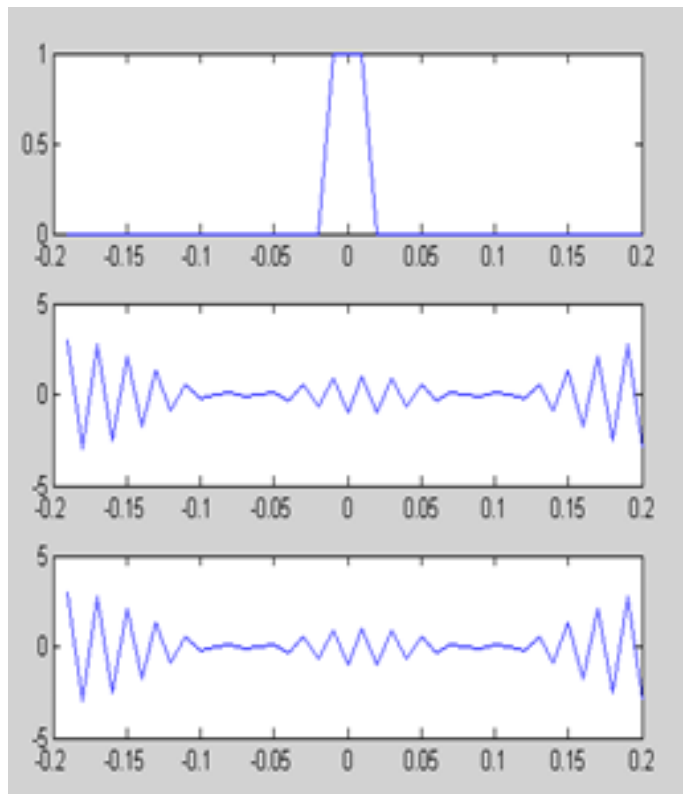
Program

```
clc  
clear all  
clf  
inc=0.01  
t=-0.2+inc:inc:0.2  
w=0.025  
ra=rectpuls(t,w)  
subplot(211)  
plot(t,ra)  
freq=fft(ra)  
subplot(212)  
plot(t,freq)
```



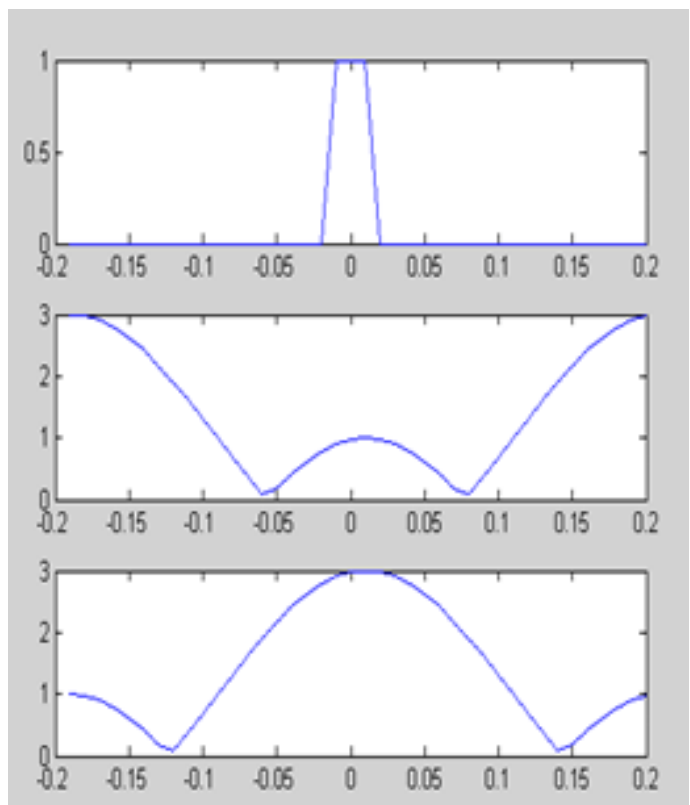
Program

```
clc  
clear all  
clf  
inc=0.01  
t=-0.2+inc:inc:0.2  
w=0.025  
ra=rectpuls(t,w)  
subplot(311)  
plot(t,ra)  
freq=fft(ra)  
subplot(312)  
plot(t,freq)  
fft2=fftshift(freq)  
subplot(313)  
plot(t,fft2)
```



Program

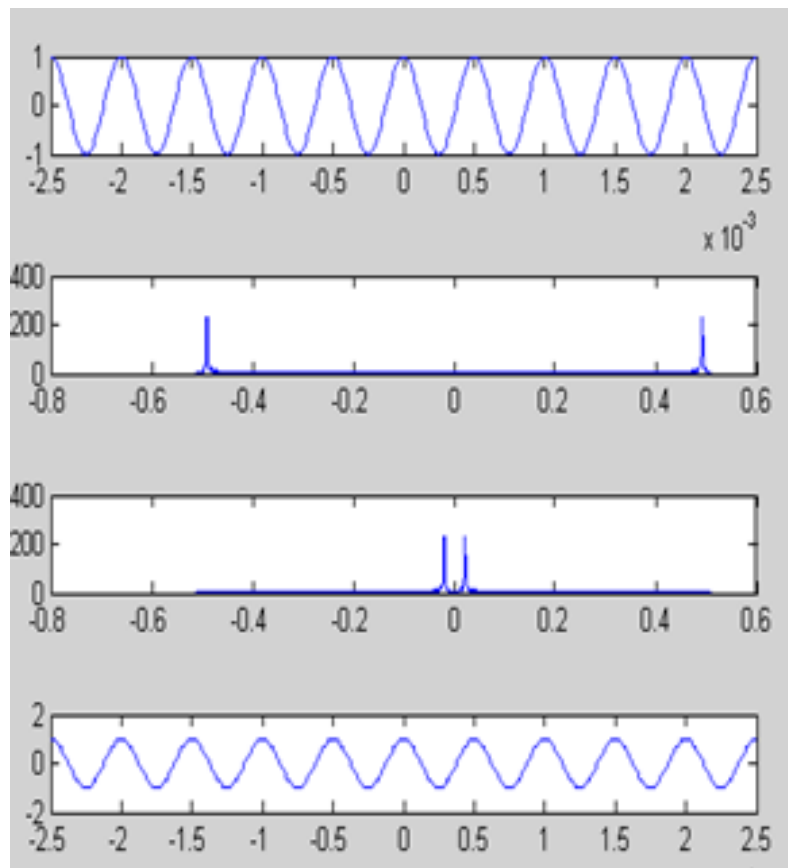
```
clc  
clear all  
inc=0.01  
t=-0.2+inc:inc:0.2  
w=0.025  
ra=rectpuls(t,w)  
subplot(311)  
plot(t,ra)  
freq=fft(ra)  
subplot(312)  
plot(t,abs(freq))  
fft2=fftshift(freq)
```




```
subplot(313)  
plot(t,abs(fft2))
```

Program

```
clc  
clear all  
t=1/2000  
n=2^10  
inc=t/50  
t=-5*t:inc:5*t  
l=length(t)  
xt=cos(4000*pi*t)  
y=fft(xt,n)  
z=abs(y)  
i=ifft(y,n)  
f=(-n/2)+1:n/2  
subplot(411)  
plot(t,xt)  
subplot(412)  
plot(f/1000,z)  
subplot(413)  
plot(f/1000,fftshift(z))  
subplot(414)  
plot(t,i(1:501))
```



Lab Assignment No. 9

Low pass filter:

1. LOW PASS FIR DIGITAL FILTER

The simplest Low Pass FIR filter is the moving average filter with $M = 2$ and has the transfer function given by

$$H_{LP}(z) = \frac{1+z^{-1}}{2} \rightarrow \text{Equation A}$$

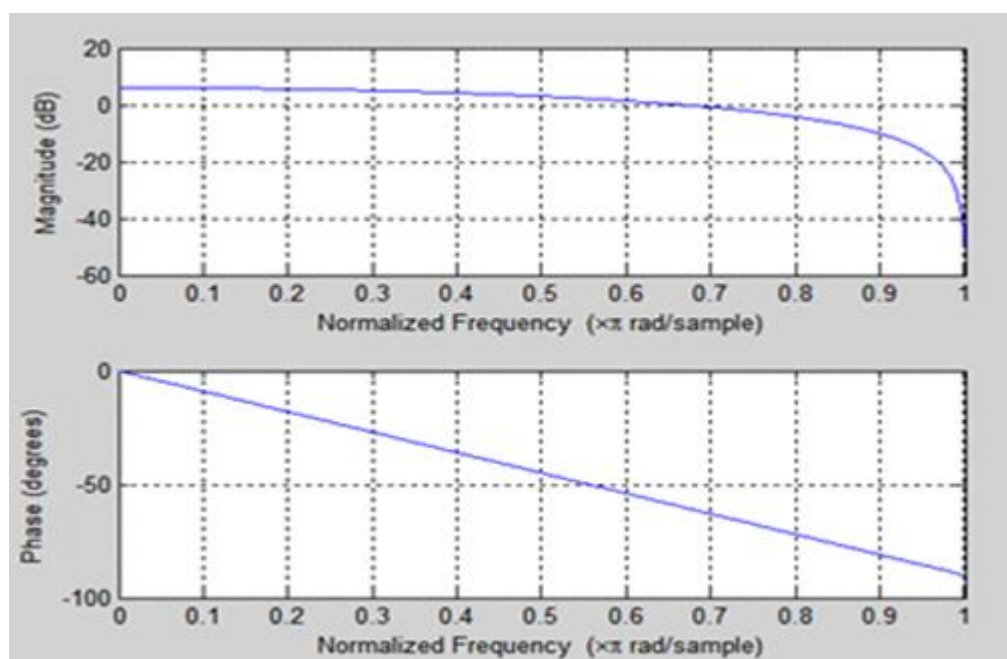
The above transfer function has a zero at $z = -1$ and a pole at $z = 0$. The pole vector has a magnitude of unity, the radius of the unit circle, for all values of w . On the other hand, as w increases from 0 to π , the magnitude of the zero vector decreases from a value of 2, the diameter of the unit circle to zero. Hence the magnitude response is a monotonically decreasing function of w from $w = 0$ to $w = \pi$. The maximum value of the magnitude function is unity at $w = 0$, and the minimum value is zero at $w = \pi$.

The frequency response for this filter is given by

$$H_{LP}(e^{jw}) = e^{-jw/2} \cos(w/2) \rightarrow \text{Equation B}$$

Program

```
clear all
clc
clf
a=[1 0]
b=[1 1]
N=1000
freqz(b,a,N)
```

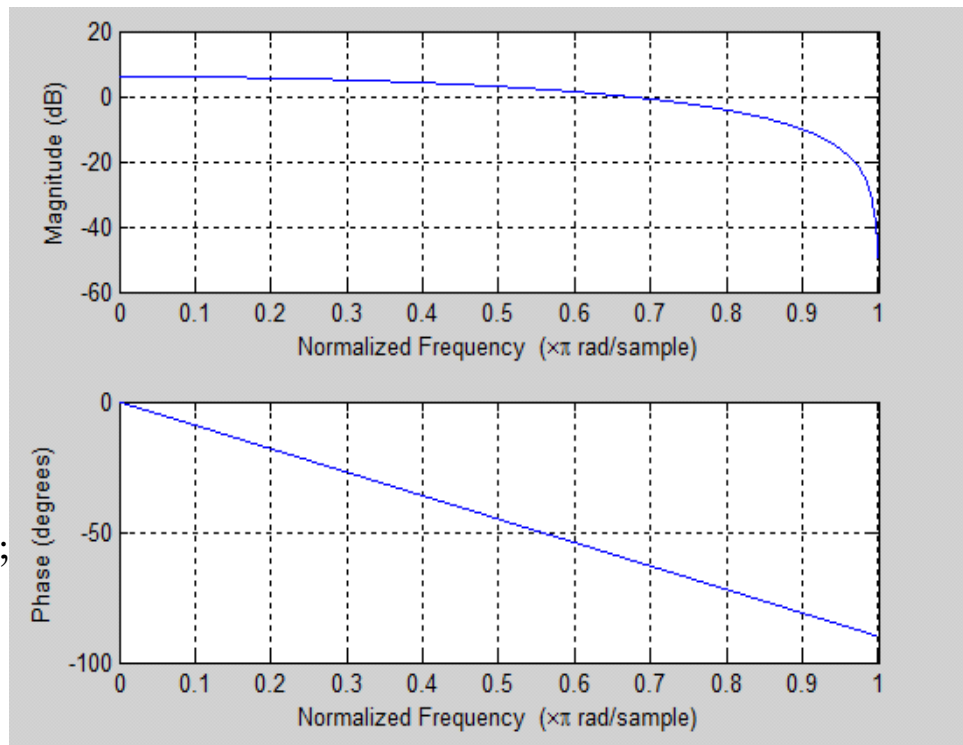


Program

```

clear all
clc
clf
a=[1 0]
b=[1 1]
N=1000
freqz(b,a,N);
figure;
[H,w]=freqz(b,a,N);
subplot(211)
plot(N,abs(H))
subplot(212)
plot(N,angle(H))

```

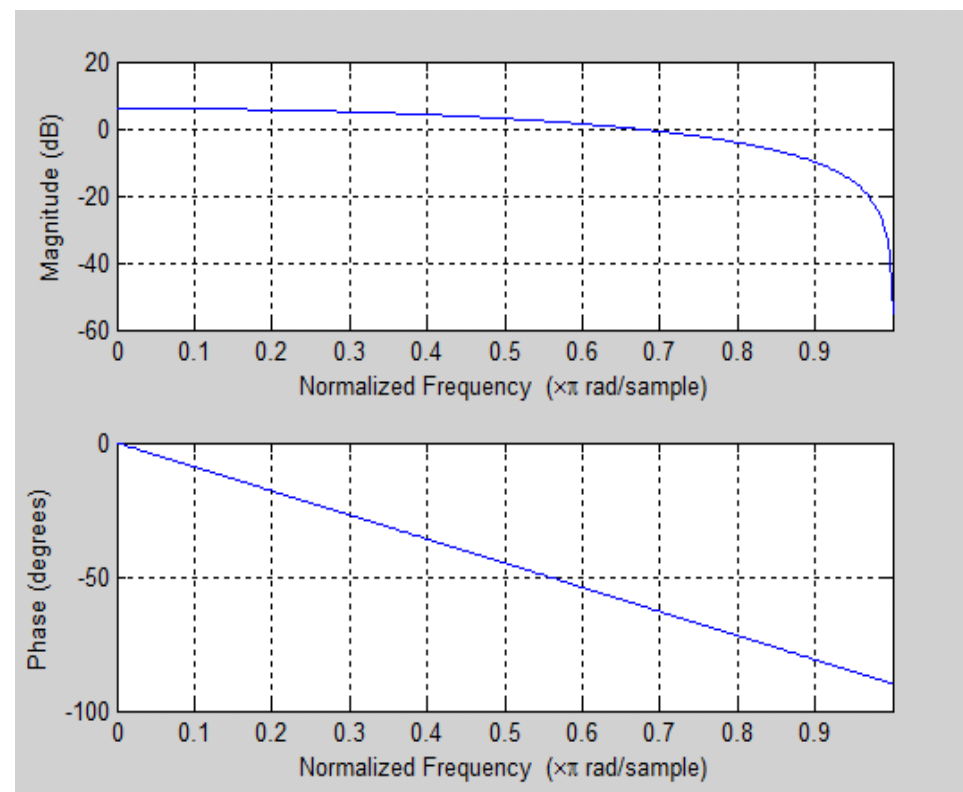


Program

```

clear all
clc
clf
a=[1 0]
b=[1 1]
w=0*pi:0.01:pi
freqz(b,a,w);
figure;
H=freqz(b,a,w);
subplot(211)
plot(w,abs(H))
subplot(212)
plot(w,angle(H))

```

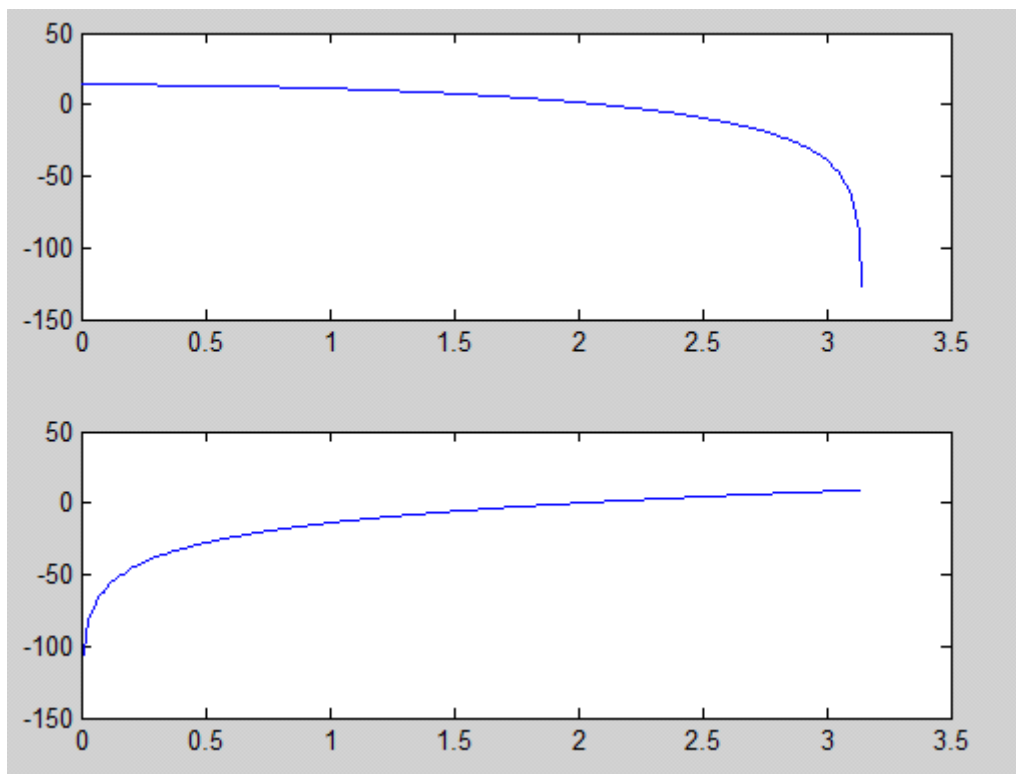
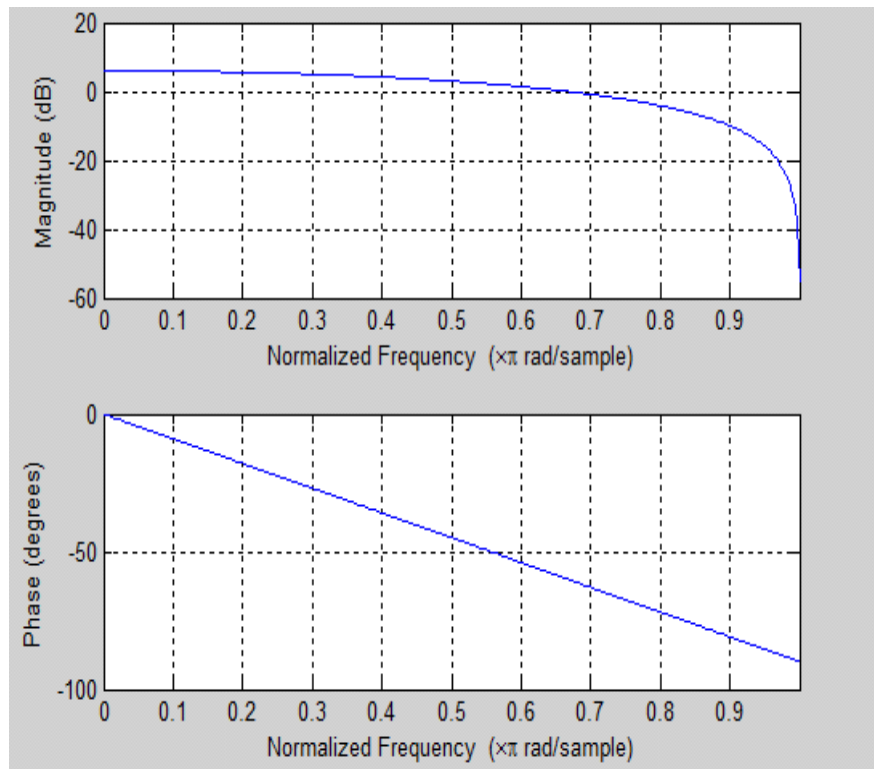


Program

```

clear all
clc
clf
a=[1 0]
b=[1 1]
w=0*pi:0.01:pi
freqz(b,a,w);
figure;
H=freqz(b,a,w);
subplot(211)
plot(w,20*log(abs(H)))
subplot(212)
plot(w,20*log(angle(H)))

```

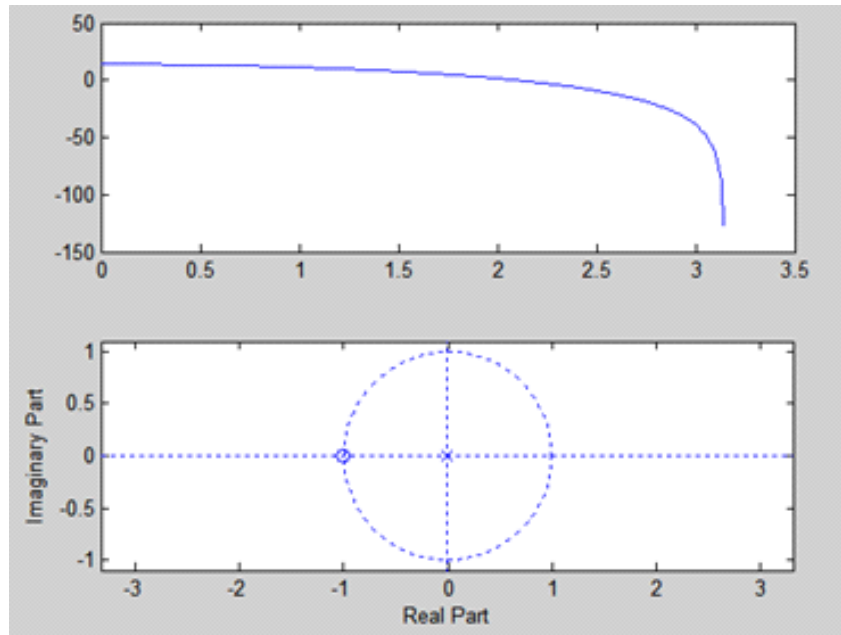


Program

```

clear all
clc
clf
a=[1 0]
b=[1 1]
figure;
w=0*pi:0.01:pi;
H=freqz(b,a,w);
subplot(211)
plot(w,20*log(abs(H)))
subplot(212)
plot(w,20*log(angle(H)))
[z,p,k]=tf2zp(b,a)
zplane(z,p)

```

**Output**

```

z = -1
p = 0
k = 1

```

Program

```

clear all
clc
a=[1 0]
b=[1 1]
figure;
[z,p,k]=tf2zp(b,a)
zplane(z,p)

```

```

Command Window

a =

    1    0

b =

    1    1

z =

   -1

p =

    0

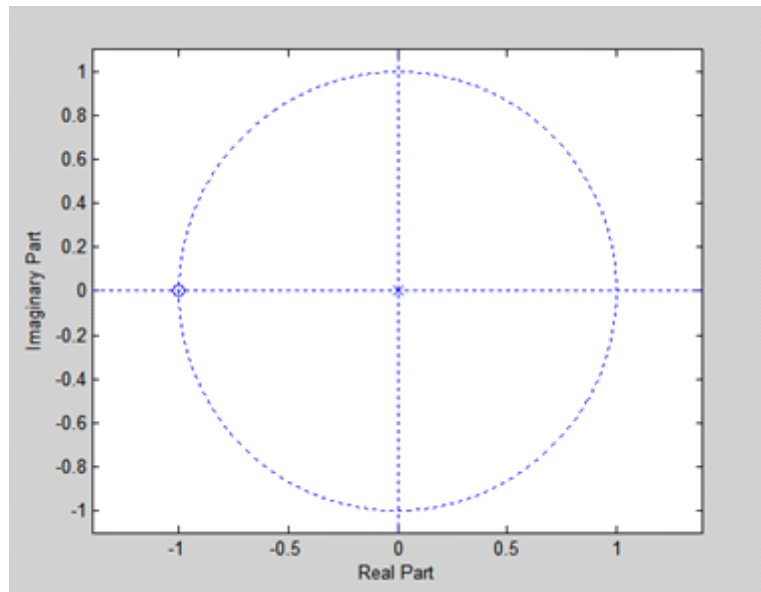
k =

    1

>> |

```

Output:



High pass filter:

2. HIGH PASS FIR DIGITAL FILTER

The simplest High Pass FIR filter is obtained by replacing z with $-z$ in Equation A , resulting in a transfer function

$$H_{HP}(z) = \frac{1 - z^{-1}}{2} \quad \rightarrow \text{Equation C}$$

The frequency response for this filter is given by

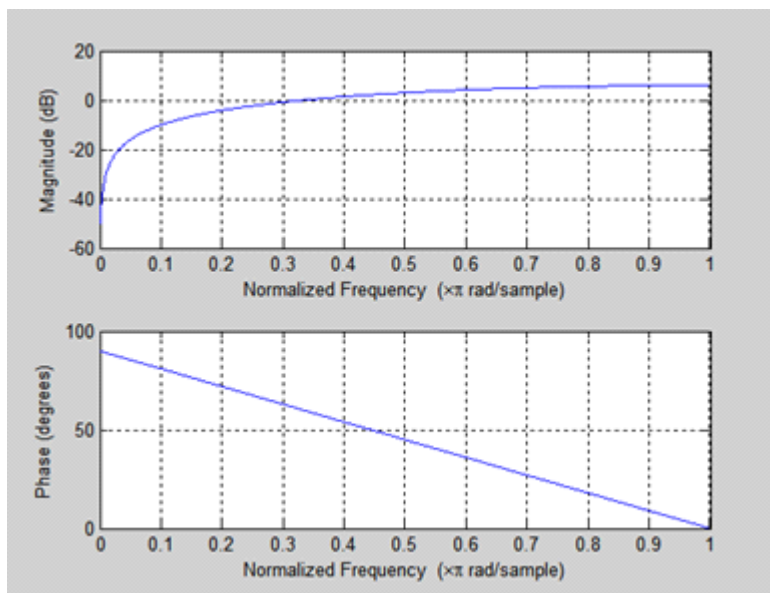
$$H_{HP}(e^{j\omega}) = j e^{-j\omega/2} \sin(\omega/2) \quad \rightarrow \text{Equation D}$$

The transfer function given by Equation C has a zero at $z = 1$ or $\omega = 0$, which is in the stop band of the filter

Program

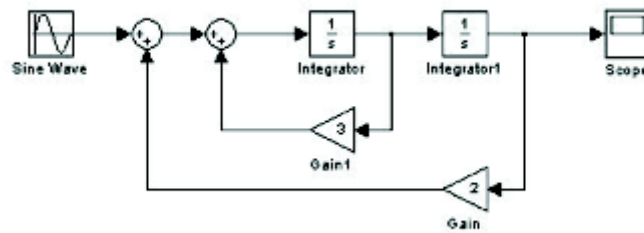
```
clear all  
clc  
clf  
a=[1 0]  
b=[1 -1]  
N=1000  
freqz(b,a,N)
```

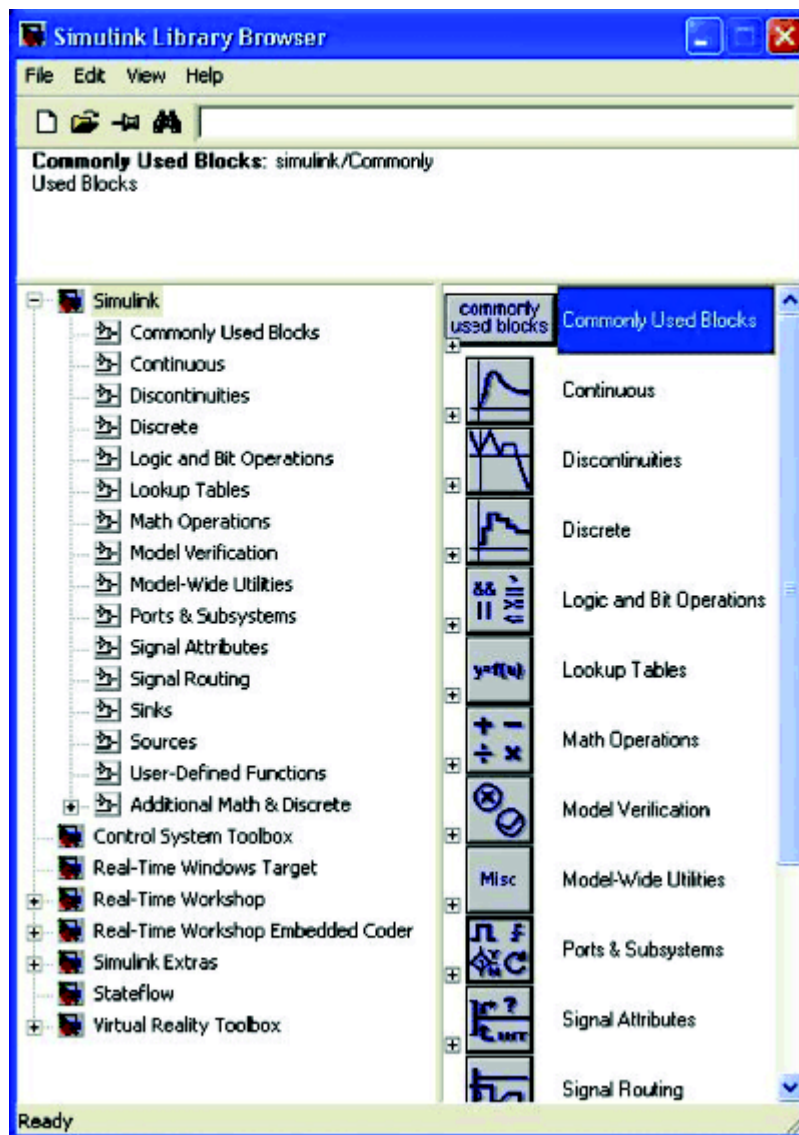
```
Command Window  
  
a =  
    1    0  
  
b =  
    1   -1  
  
N =  
   1000
```



Lab Assignment No: 10**Simulink model**

To compute trajectories with MATLAB by means of block diagrams, the tool Simulink should be used. To open Simulink, type 'simulink' in your command window in your MATLAB window. Now, the Simulink Library Browser should appear. A new file can be opened by selecting File> New> Model or selecting the icon on the toolbar. In the window appearing, one should build the model. Basically, you will have to draw the block diagram. Hereto, blocks from the Library Browser should be dragged towards the model while holding your left mouse button.





In the library browser, you will find the integrator block in the category "Continuous", the gain and sum block are part of the category "Math Operations". When a block is inserted in the model, one can flip the block (press the right mouse button on block, select Format> Flip). To connect two blocks, click on the >> exiting the first block, hold your right-mouse-button and drag a line towards the >> entering the second block. You can give a name to a signal by double-clicking on the connection.

Sometimes, you will need to split a signal, for example the signal dx in the example of the previous section. Hereto, first make one of the connections,

click on the >> entering the third block you want to connect, hold your left-mouse-button and drag a line towards the already existing connection.

Blocks and arrows can be moved in your model by using either the arrows on your keyboard or by dragging them with your mouse. Many blocks in Simulink have certain parameters. To edit these parameters, double click on the block and change the value. The particular gain used is a parameter of the gain block. The initial value of a signal exiting an integrator block is a parameter of this block. In the parameters of the add-block, one can make the block subtract signals instead of add them, by changing the "list of signs" from " | + + " to " | + - ".

The integrator block has its initial condition as parameter. Here, one specifies the initial output of the integrator block. By changing these initial conditions, a Simulink model can be used to compute trajectories from predefined initial conditions.

Inputs in Simulink have to be defined as a function of time, for example $u(t) = 3 \sin(2t)$. You will find many possible inputs in the Sources-category of your Library browser. Most used are the sine wave, step or constant blocks. A sine wave block has the properties amplitude, frequency and phase, the latter describing the initial phase of the signal. A step block has parameters initial time, initial value and final value. A constant source has the constant value as a parameter.

To handle outputs in Simulink, one needs to use the blocks in the category Sinks in the Library browser. Most used are the "Scope", "To File", and "To Workspace blocks". A scope will visualize the signals entering it. When you have finished a simulation, double-click on the scope to see the results. The "To File" block saves the entering signal to a .mat file. When you select the block properties, you can change the file name, and the name of the stored variable.

The block "To Workspace" saves the signals to the workspace, such that one can use them later in the command window. In this block's properties, one can change the variable name, and choose whether the data should be saved as structure, structure with time or array.

It is allowed to call variables that are present in your workspace in block parameters. For example, a gain-block with gain "k" will work properly, when a scalar "k" is available in the MATLAB workspace.

To save your model, press the "save" button in the toolbar of your model window. Simulink models will be saved with the extension .mdl. Later, you can open these models by selecting the file saved.

Please note, that the Library browser contains many more blocks as mentioned in this section. When you double click on a certain block in the library browser, you will see the purpose and parameters of the block.

Lab Assignment No: 11**High Pass Active Filter Trainer**

A high-pass filter is usually modeled as a linear time-invariant system. It is sometimes called a low-cut filter or bass-cut filter. High-pass filters have many uses, such as blocking DC from circuitry sensitive to non-zero average voltages or RF devices. They can also be used in conjunction with a low-pass filter to make a bandpass filter.

The simple first-order electronic high-pass filter shown in Figure 1 is implemented by placing an input voltage across the series combination of a capacitor and a resistor and using the voltage across the resistor as an output. The product of the resistance and capacitance ($R \times C$) is the time constant (τ); it is inversely proportional to the cutoff frequency f_c , that is,

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi RC},$$

where f_c is in hertz, τ is in seconds, R is in ohms, and C is in farads.

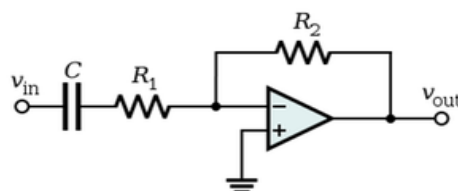
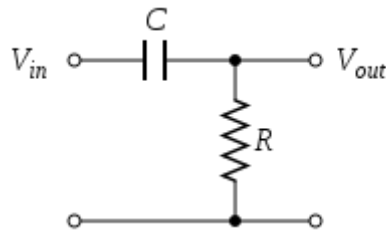


Figure shows an active electronic implementation of a first-order high-pass filter using an operational amplifier. In this case, the filter has a passband gain of $-R_2/R_1$ and has a corner frequency of

$$f_c = \frac{1}{2\pi\tau} = \frac{1}{2\pi R_1 C},$$

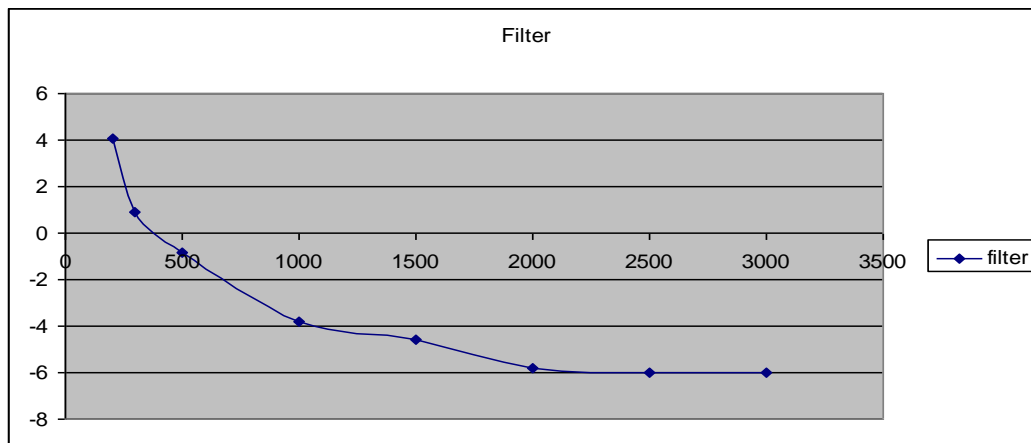
Because this filter is active, it may have non-unity passband gain. That is, high-frequency signals are inverted and amplified by R_2/R_1 .



Observation and Calculations:

1st Order High Pass Active Filter:

Frequency	V _{in}	V _{out}	Gain	Gain
Hz	Volts	Volts		Db
3000	4	8	2	-6.0206
2500	4	8	2	-6.0206
2000	4	7.8	1.95	-5.8006
1500	4	6.8	1.7	-4.6089
1000	4	6.2	1.55	-3.8066
500	4	4.4	1.1	-0.8278
300	4	3.6	0.9	0.9151



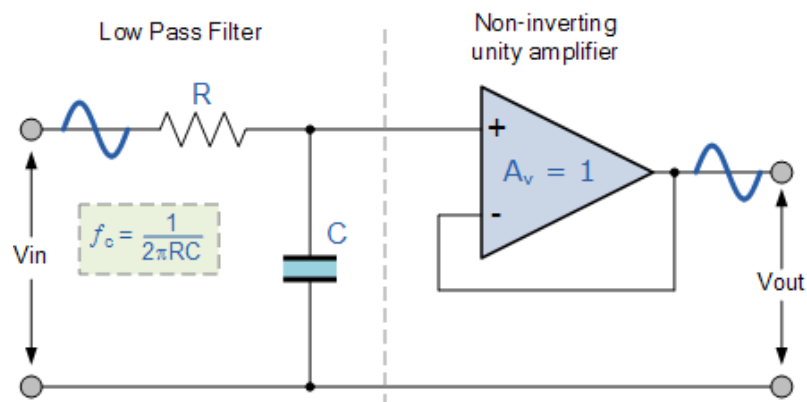
Lab Assignment No: 12**Low Pass Active Filter Trainer**

A low-pass filter is an electronic filter that passes low-frequency signals and attenuates (reduces the amplitude of) signals with frequencies higher than the cutoff frequency. The actual amount of attenuation for each frequency varies from filter to filter. It is sometimes called a high-cut filter, or treble cut filter when used in audio applications. A low-pass filter is the opposite of a high-pass filter. A band-pass filter is a combination of a low-pass and a high-pass.

Low-pass filters exist in many different forms, including electronic circuits (such as a hiss filter used in audio), anti-aliasing filters for conditioning signals prior to analog-to-digital conversion, digital filters for smoothing sets of data, acoustic barriers, blurring of images, and so on. The moving average operation used in fields such as finance is a particular kind of low-pass filter, and can be analyzed with the same signal processing techniques as are used for other low-pass filters. Low-pass filters provide a smoother form of a signal, removing the short-term fluctuations, and leaving the longer-term trend.

An optical filter could correctly be called low-pass, but conventionally is described as "longpass" (low frequency is long wavelength), to avoid confusion. The most common and easily understood active filter is the Active Low Pass Filter. Its principle of operation and frequency response is exactly the same as those for the previously seen passive filter, the only difference this time is that it uses an op-amp for amplification and gain control. The simplest form of a low pass active filter is to connect an inverting or non-inverting amplifier, the same as those discussed in the Op-amp tutorial, to the basic RC low pass filter circuit as shown.

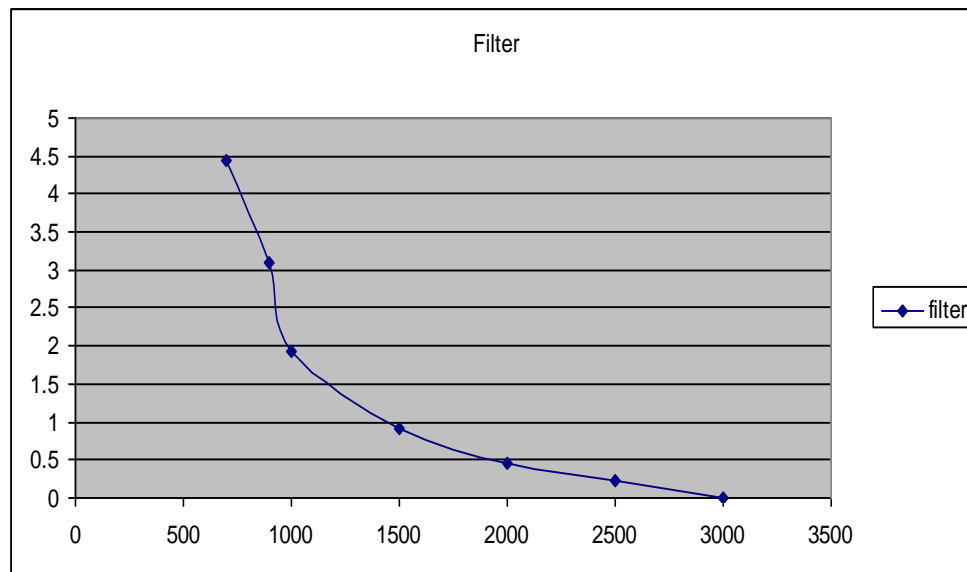
First Order Active Low Pass Filter:



A first-order filter, for example, will reduce the signal amplitude by half (so power reduces by a factor of 4), or 6 dB, every time the frequency doubles (goes up one octave); more precisely, the power rolloff approaches 20 dB per decade in the limit of high frequency. The magnitude Bode plot for a first-order filter looks like a horizontal line below the cutoff frequency, and a diagonal line above the cutoff frequency. There is also a "knee curve" at the boundary between the two, which smoothly transitions between the two straight line regions. If the transfer function of a first-order low-pass filter has a zero as well as a pole, the Bode plot will flatten out again, at some maximum attenuation of high frequencies; such an effect is caused for example by a little bit of the input leaking around the one-pole filter; this one-pole-one-zero filter is still a first-order low-pass. See Pole-zero plot and RC circuit.

Observations And Calculations:

Frequency	Vin	Vout	Gain	Gain
Hz	Volts	Volts		Db
3000	4	4	1	0
2500	4	3.9	0.975	0.219908
2000	4	3.8	0.95	0.445528
1500	4	3.6	0.9	0.91515
1000	4	3.2	0.8	1.9382
900	4	2.8	0.7	3.098039
700	4	2.4	0.6	4.436975

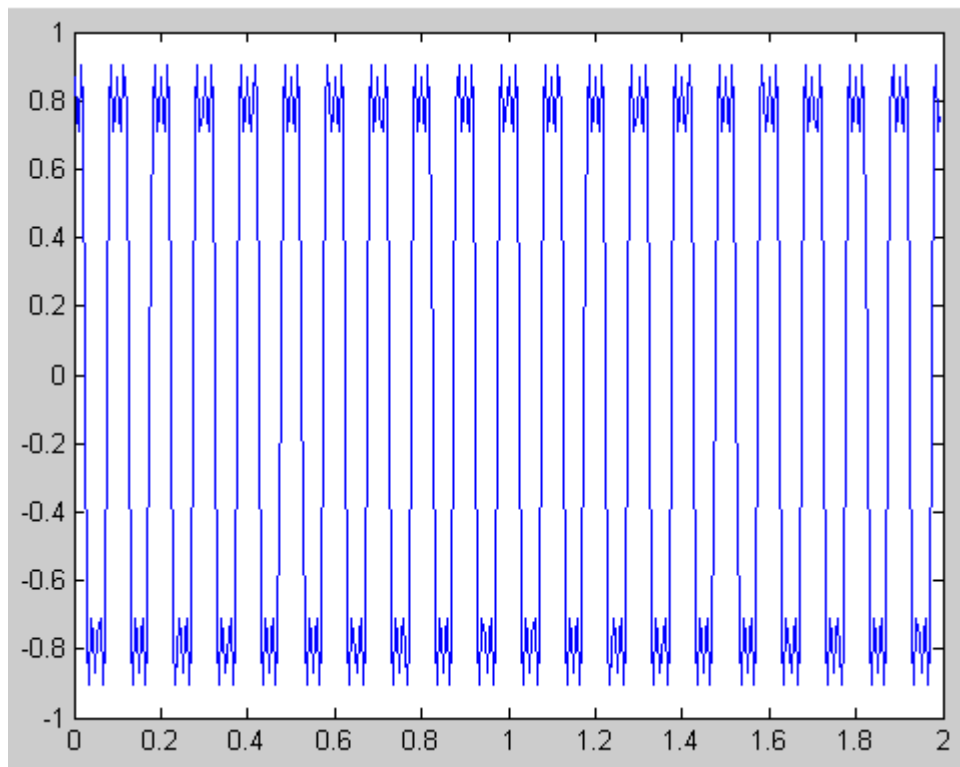


Lab Assignment No: 13

ButterWorth Filter

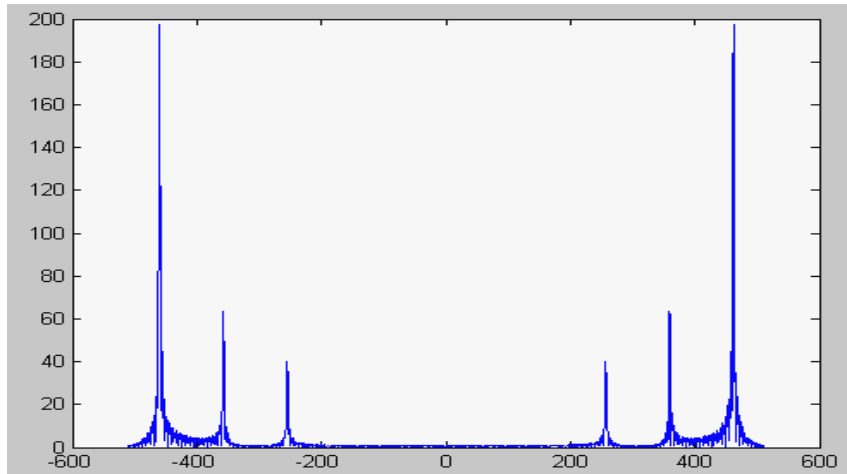
Sampled Signal.

```
clc;  
t=0:0.005:1.995;  
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);  
plot (t,x);
```



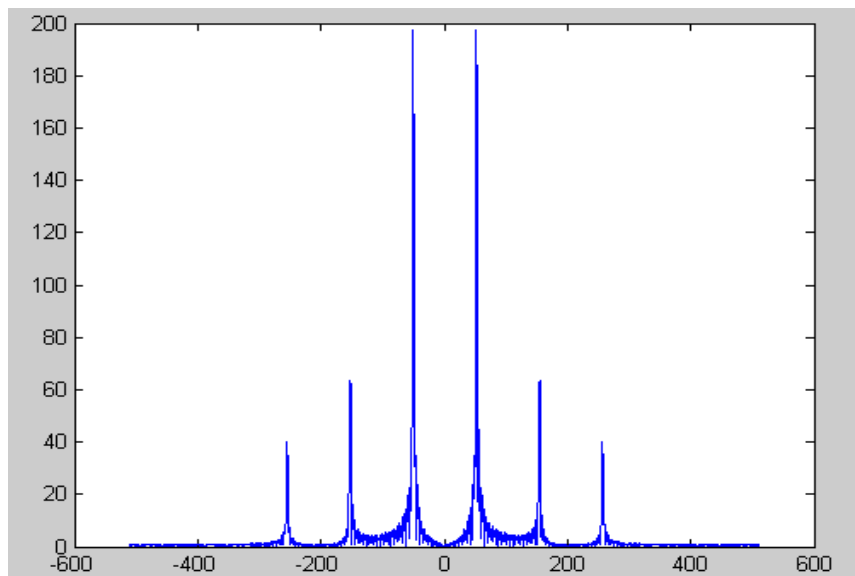
Fast Fourier Transform of Signal :

```
clc;  
t=0:0.005:1.995;  
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);  
N=1024;  
f=-(N/2)+1: N/2;  
y= fft(x,N);  
z=abs(y);  
plot(f,z);
```



Fast Fourier Transform Shift of Absolute Signal.

```
clc;  
t=0:0.005:1.995;  
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);  
N=1024;  
f=-(N/2)+1: N/2;  
y= fft(x,N);  
a=fftshift(y);  
z=abs(a);  
plot(f,z);
```

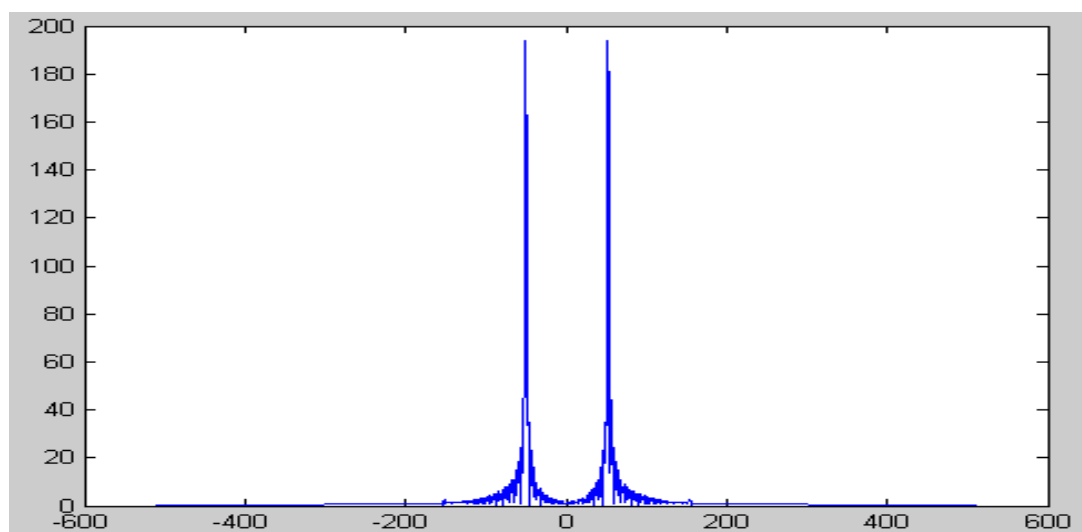
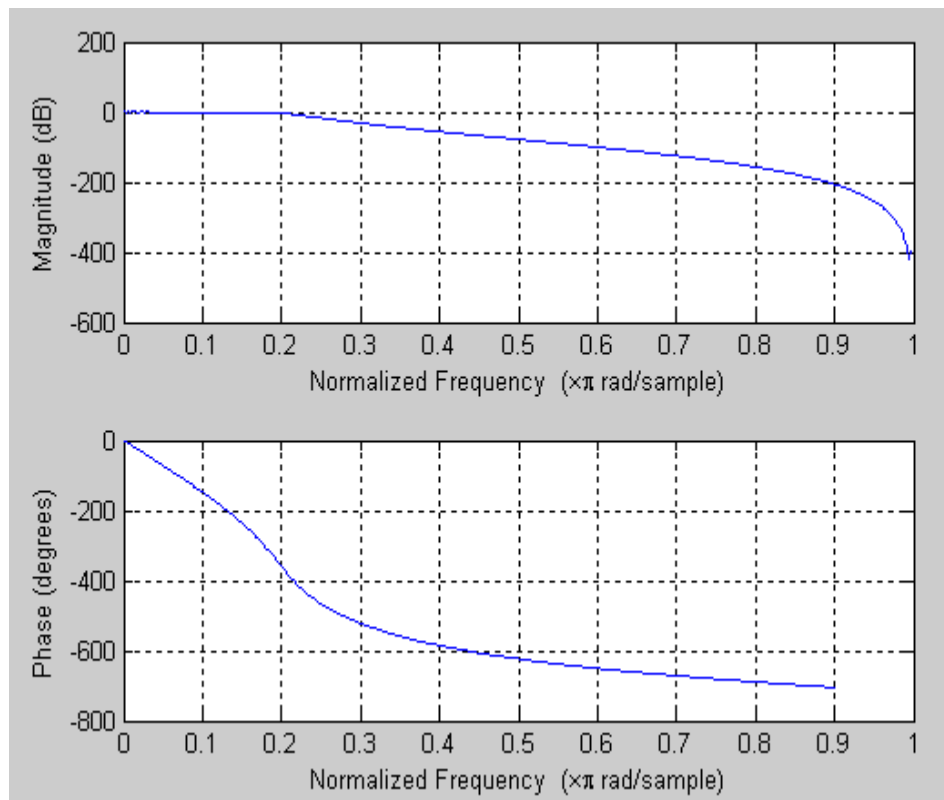


ButterWorth Specification and Rectified Signal.

```

clc;
t=0:0.005:1.995;
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);
N=1024;
f=-(N/2)+1:N/2;
y=fft(x,N);
a=fftshift(y);
z=abs(a);
fc=40;
fs=200;
wn=fc/fs;
[B,A]=butter(8,wn);
freqz(B,A);
D=filter(B,A,x);
F=fft(D,N);
S=abs(F);
s=fftshift(S);
figure
plot(f,s);

```



Lab Assignment No. 14**Chebyshev Filter**

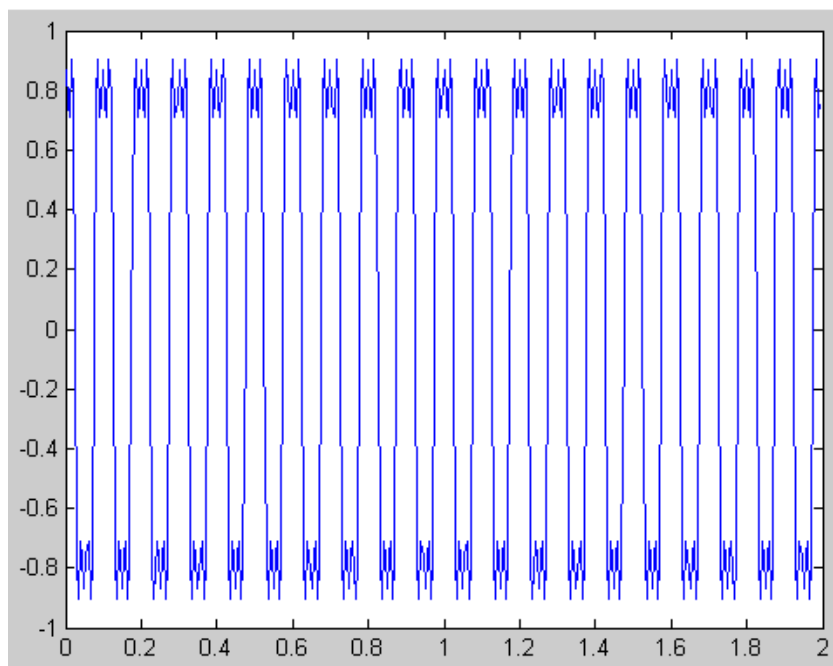
Sampled Signal.

```
clc;
```

```
t=0:0.005:1.995;
```

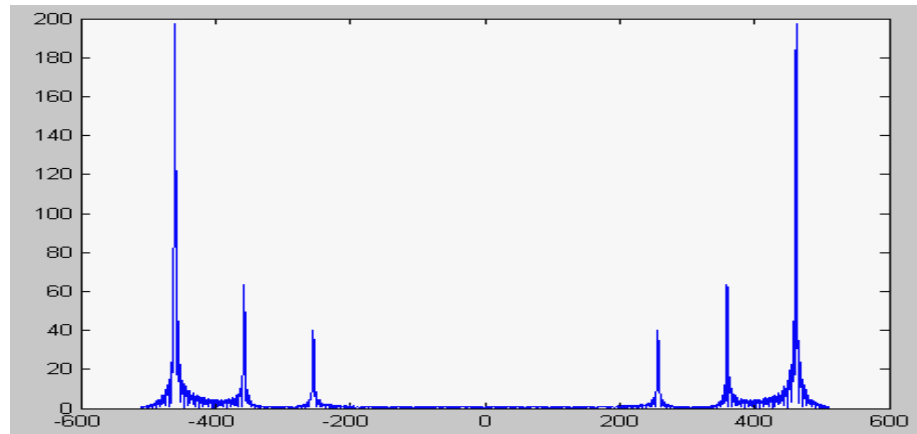
```
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);
```

```
plot (t,x);
```

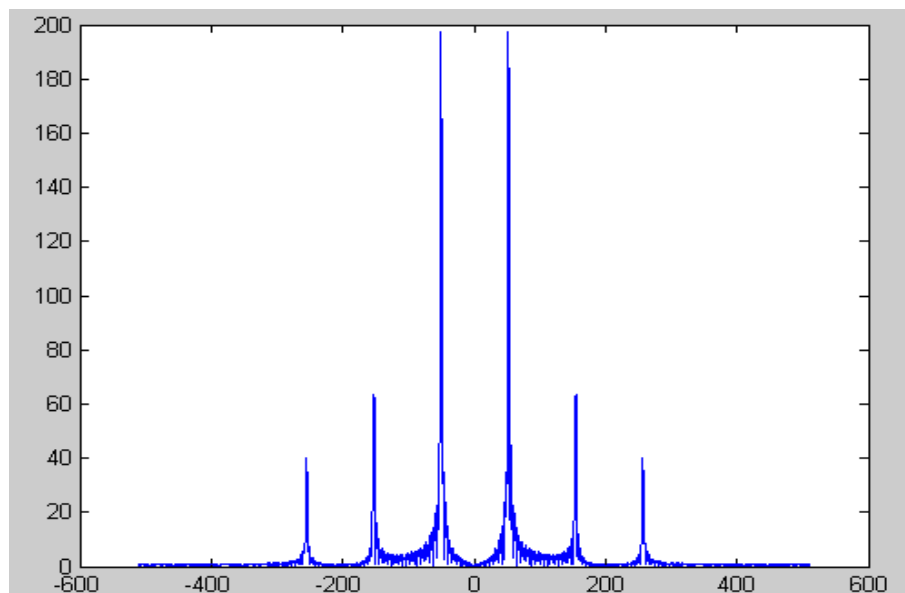


Fast Fourier Transform of Signal .

```
clc;  
t=0:0.005:1.995;  
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);  
N=1024;  
f=-(N/2)+1: N/2;  
y= fft(x,N);  
z=abs(y);  
plot(f,z);
```

**Fast Fourier Transform Shift of Absolute Signal**

```
clc;  
t=0:0.005:1.995;  
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);  
N=1024;  
f=-(N/2)+1: N/2;  
y= fft(x,N);  
a=fftshift(y);  
z=abs(a);  
plot(f,z);
```



Chebyshev Filter:

```
clc;
t=0:0.005:1.995;
x=cos(2*pi*10*t)-1/3*cos(2*pi*30*t)+1/5*cos(2*pi*50*t);
N=1024;
f=-(N/2)+1:N/2;
y=fft(x,N);
a=fftshift(y);
z=abs(a);
fc=40;
fs=200;
wn=fc/fs;
rp=0.5;
[B,A]=cheby1(8,rp,wn);
freqz(B,A);
D=filter(B,A,x);
F=fft(D,N);
S=abs(F);
s=fftshift(S);
figure
plot(f,s);
```

