Laboratory Journal of SIGNAL AND SYSTEMS

For completion of term work of 5th *semester curriculum program*

Bachelor of Technology

In

ELECTRONICS AND TELECOMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

Dr. BABASAHEB AMBEDKAR TECHNOLOGICAL UNIVERSITY

Lonere-402 103, Tal. Mangaon, Dist. Raigad (MS)

INDIA

List of Experiment

Experiment No.1

MATLAB code to generate standard signals

Theory:

The methods we use in processing a signal or in analyzing the response of a system to a signal depend heavily on the characteristic attributes of the specific signal. There are techniques that apply only to specific families of signals. Consequently, any investigation in signal processing should start with a classification of the signals involved in the specific application.

Signals can be classified into different categories depending on the characteristics of the time (independent) variable and the values they take. continuous-time signals are defined for every value of time and they take on values in the continuous interval (a,b).Discrete time signals are defined only at certain specific values of time.

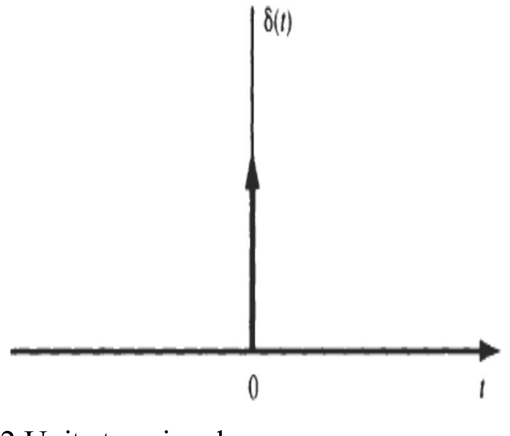
A. Standard Continuous Time signals

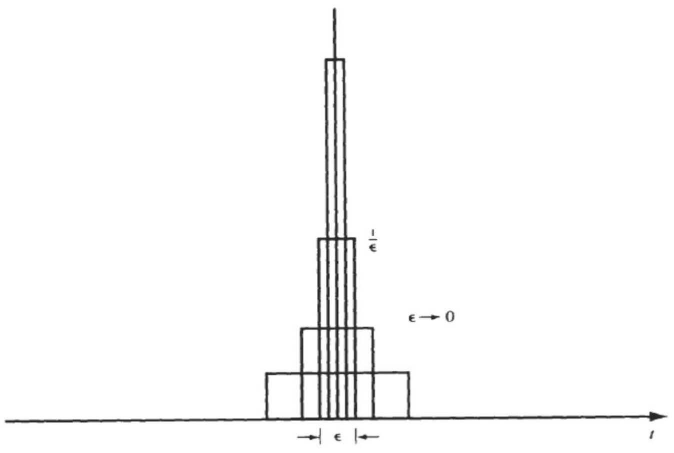
1. Unit Impulse signal

The impulse signal is a signal is a signal with infinite magnitude and zero duration,but with unit area . Mathematically ,impulse signal is defined as,

$$\delta(t) = \begin{cases} 0 & t \neq 0 \\ \infty & t = 0 \end{cases}$$

$$\int_{-\varepsilon}^{\varepsilon} \delta(t)\, dt = 1$$

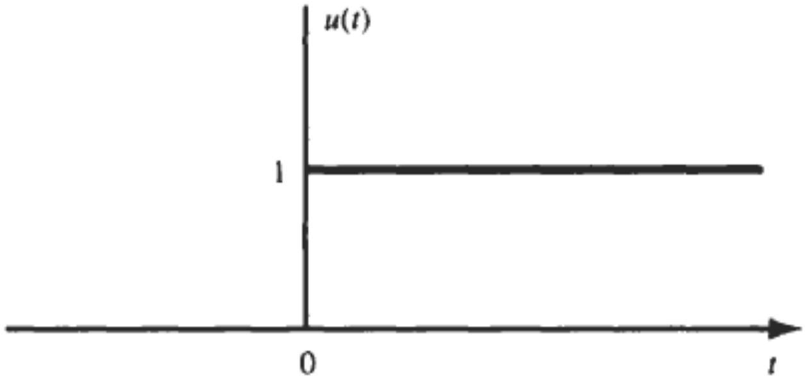




2.Unit step signal

The unit step signal is defined as,

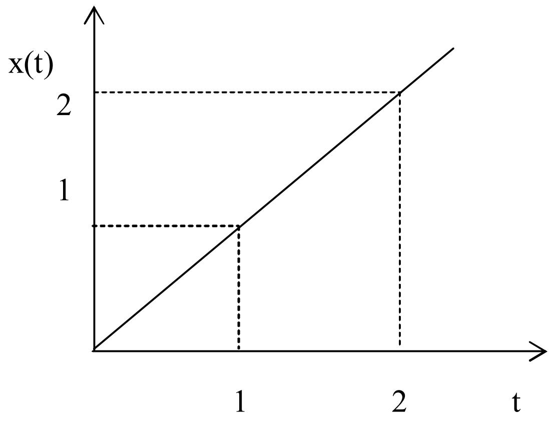
$$u(t) = \begin{cases} 1 & t > 0 \\ 0 & t < 0 \end{cases}$$



3 .Unit Ramp signal

The unit ramp signal is defined as

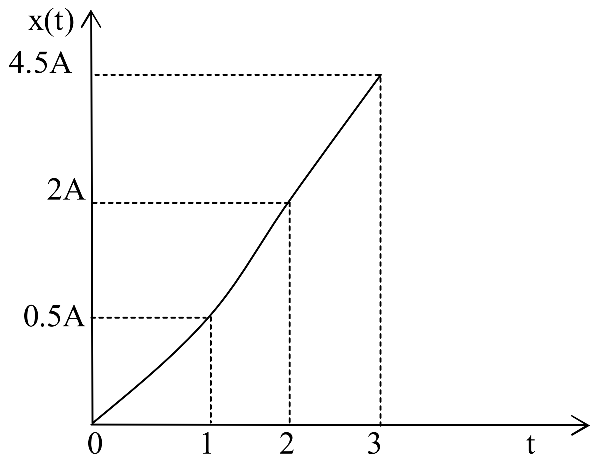
$$U(t) = t \; ; \; t \geq 0$$
$$= 0 \; ; \; t < 0$$



4.Parabolic signal

The parabolic signal is defined as,

$$x(t) = A t^2 / 2 \; ; \; \text{for } t \geq 0$$
$$= 0 \; ;$$



5. Sinusoidal signal

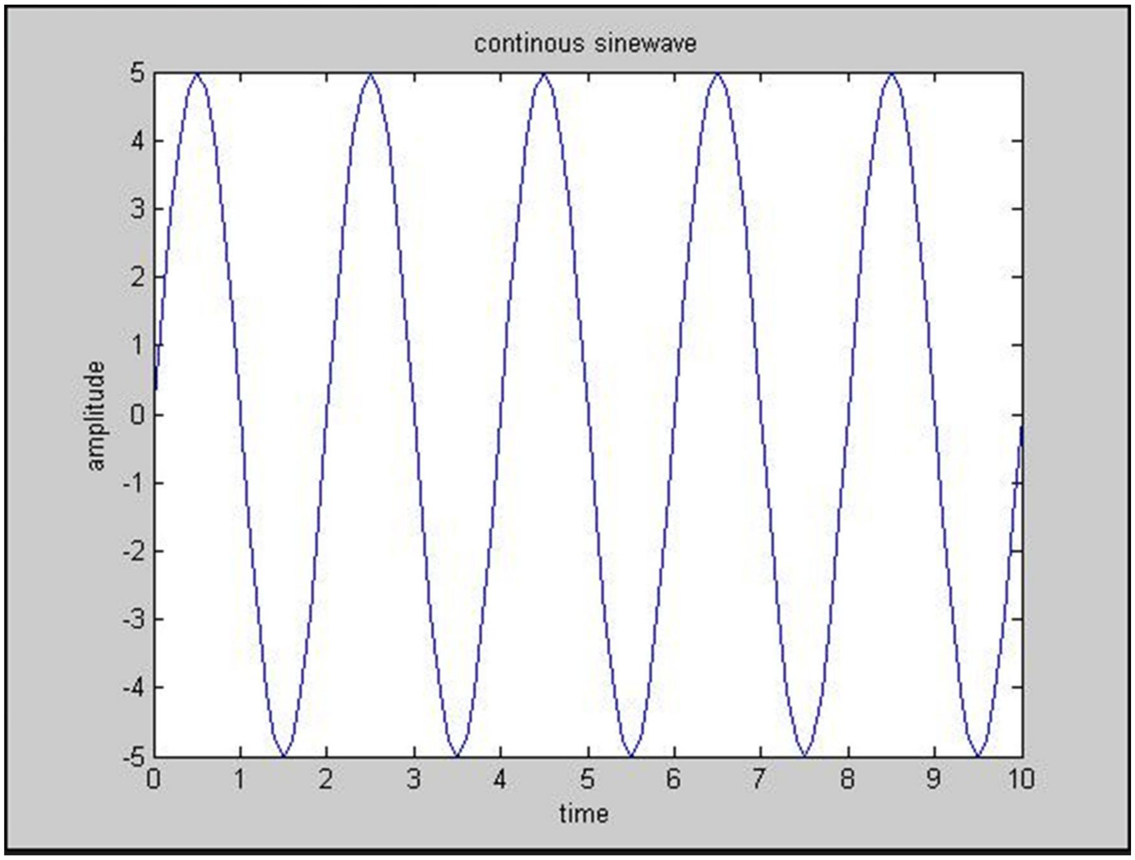
The sinusoidal signal is defined as,

$$X(t) = A \sin(\Omega_0 t + \Phi)$$

Where, $\Omega_0 = 2\pi F_0 = 2\pi/T$ =Angular frequency in rad/sec

F_0 =frequency in cycles/sec or Hz

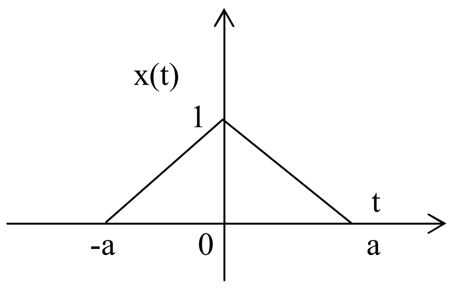
T= Time period in sec



6.Triangular pulse signal

The triangular pulse signal is defined as

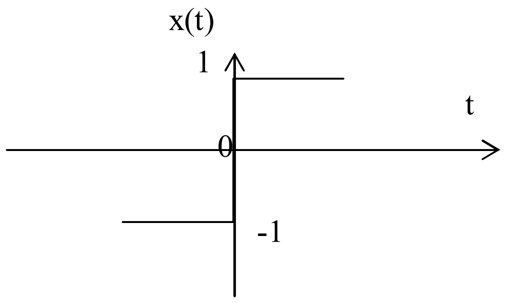
$$x(t) = \Delta_a(t) = 1 - |t|/a \quad ; \quad |t| \le a$$
$$= 0 \quad ; \quad |t| > a$$



7.signum signal

The signum signal is defined as the sign of independent variable t.
Therefore ,the signum signal is expressed as,

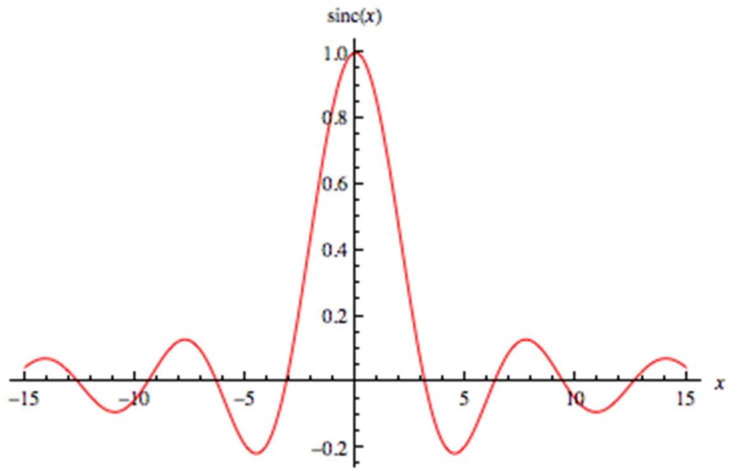
$$x(t) = \mathrm{sgn}(t) = 1 \;;\; t > 0$$
$$= 0 \;;\; t = 0$$
$$= -1 \;;\; t < 0$$



8.sinc singnal

The sinc signal is defined as,

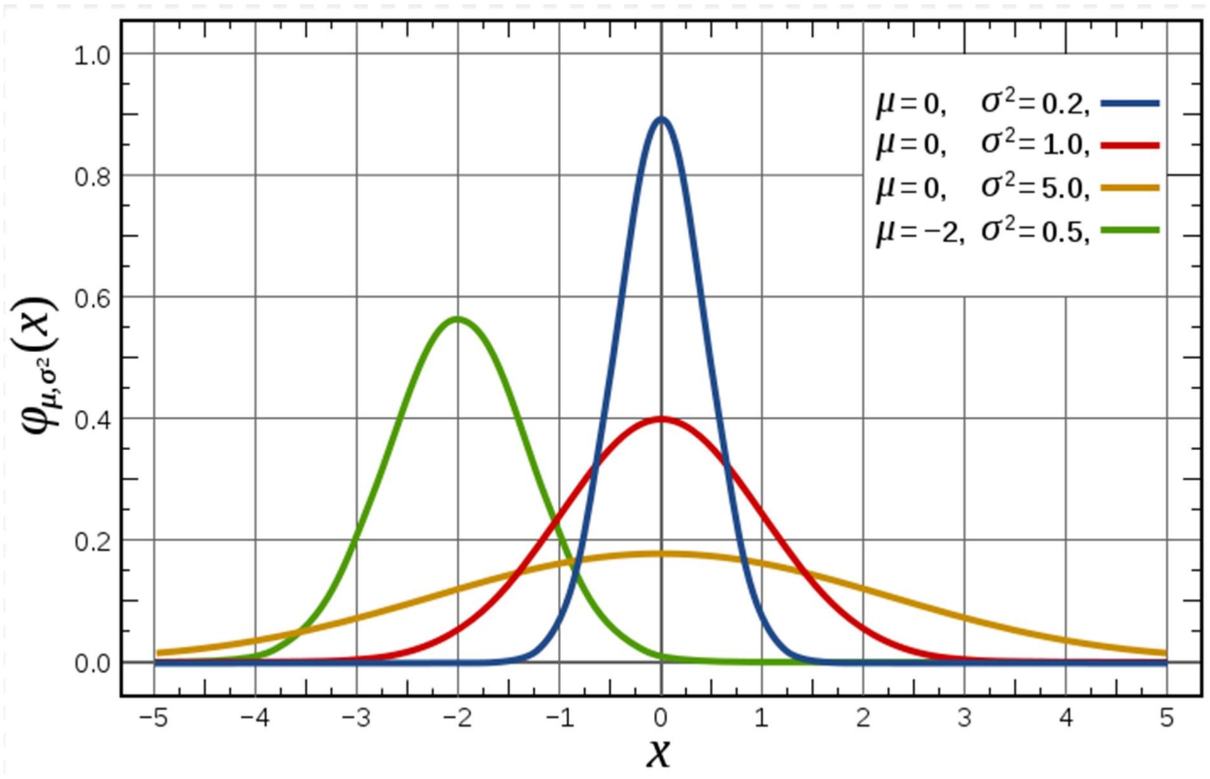
$$x(t) = \mathrm{sinc}\,(t) = \sin t/\ t \;;\; -\infty < t < \infty$$



9. Gaussian signal

The Gaussian signal is defined as,

$$x(t) = g_a(t) = e^{-(a^2)(t^2)} \;;\; -\infty < t < \infty$$

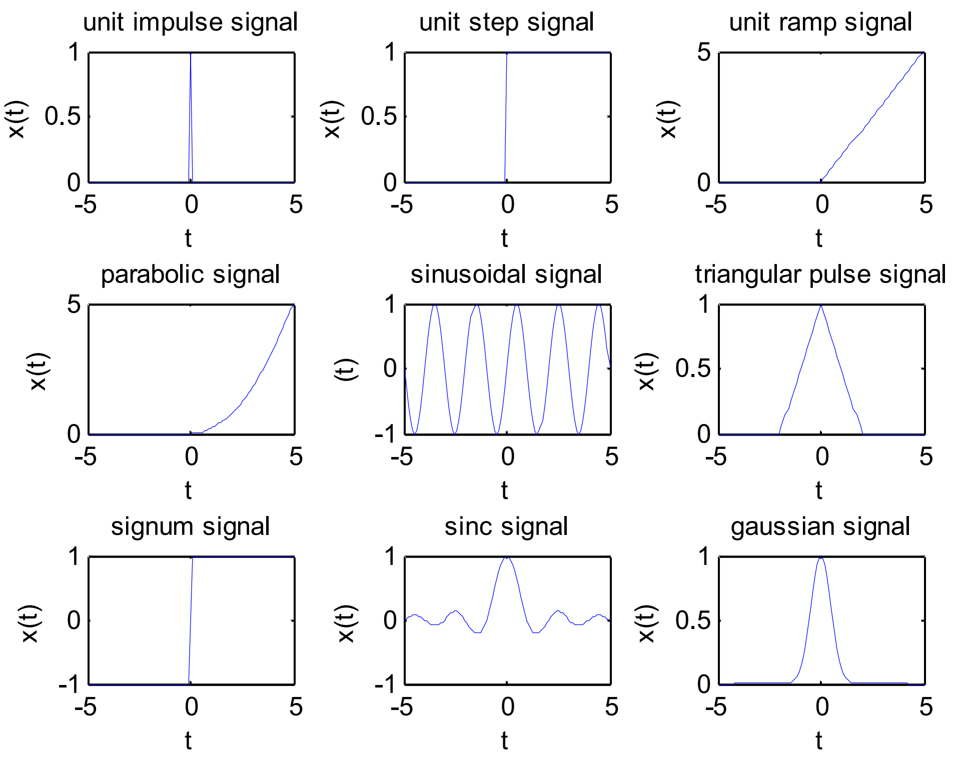


MATLAB CODE:

```
%************************program to plot some standard signals*****************
tmin=-5;
dt=0.1;
tmax=5;
t=tmin :dt:tmax;          % set a time vector.
%********************* unit impulse signal*************************************
x1=1;
x2=0;
x=x1.*(t==0)+x2.*(t~=0);
subplot(3,3,1);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('unit impulse signal');
```

```
****************************  unit step signal************************
x1=1;
x2=0;
x=x1.*(t>=0)+x2.*(t<0);
subplot(3,3,2);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('unit step signal');
 %*************************** unit  ramp signal ****************************
x1=t;
x2=0;
x=x1.*(t>=0)+x2.*(t<0);
subplot(3,3,3);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('unit ramp signal');
%************************** parabolic signal ***********************
A=0.4;
x1=(A*(t.^2))/2;
x2=0;
x=x1.*(t>=0)+x2.*(t<0);
subplot(3,3,4);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('parabolic signal');
%********************************  sinusoidal signal
T=2;
F=1/T;
x= sin(2*pi*F*t);
subplot(3,3,5);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('sinusoidal signal');

%***************************** triangular pulse signal
a=2;
```

```
x1=1-abs(t)/a;
x2=0;
x=x1.*(abs(t)<=a)+x2.*(abs(t)>a);
subplot(3,3,6);
plot(t,x);
xlabel(‘t’);
ylabel(‘x(t)’);
title(‘triangular pulse signal’);
%******************************** signum signal
x1=1;
x2=0;
x3=-1;
x=x1.*(t>0)+x2.*(t==0)+x3.*(t<0);
subplot(3,3,7);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('signum signal');
%*************************sinc pulse
x=sinc(t);
subplot(3,3,8);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('sinc signal');
%*************************** gaussian signal
a=2;
x=exp(-a.*(t.^2));
subplot(3,3,9);
plot(t,x);
xlabel('t');
ylabel('x(t)');
title('gaussian signal');
```


unit impulse signal
unit step signal
unit ramp signal
parabolic signal
sinusoidal signal
triangular pulse signal
signum signal
sinc signal
gaussian signal
x(t)
t

Experiment No.2

MATLAB code to perform basic operations on signals

Theory:

1 .Scaling of Continuous Time signals

The two types of scaling continuous time signals are,

a. Amplitude Scaling

b. Time Scaling

a. Amplitude Scaling:

The Amplitude Scaling is performed by multiplying the amplitude of the signal by a constant.

Let x(t) be a continuous time signal. Now Ax(t) is the amplitude scaled version of x(t) , where A is a constant. when |A| >1,then Ax(t) is amplitude magnified version of x(t) and when |A| <1,then Ax(t) is the amplitude attenuated version of x(t).

b. Time Scaling:

The Time scaling is performed by multiplying the variable time by a constant.

If x(t) is continuous time signal ,then x(At) is the time scaled version of x(t),where A is a constant. when |A|>1,then x(At) is the time compressed version of x(t) and when |A|<1,then x(At) is the time expanded version of x(t).

2.Folding:

The folding of continuous time signal x(t) is performed by changing the sign of time base t in the signal x(t)

The folding operation produces a signal x(-t) which is the mirror image of the original signal x(t) with respect to the time origin t=0.

3. Time shifting

The time shifting of a continuous time signal x(t) is performed by replacing the independent variable t by –m to get the time shifted signal x(t-m),where m represents the time shift in seconds.

In x(t-m),if m is positive, then the time shift results in a delay by m seconds. The Delay results in shifting the original signal x(t) to right, to generate the time shifted signal x(t-m).

In x(t-m), if m is negative, then the time shift results in a advance by m seconds. The advance results in shifting the original signal x(t) to left, to generate the time shifted signal x(t-m).

4. Addition of continuous time signal:

The addition of two continuous time signals is performed by adding the value of the two signals corresponding to same instant of time.

The sum of two $x_1(t)$ and $x_2(t)$ is a signal y(t), whose value at any instant is equal to the sum of the value of these two signals at that instant.

i.e. $y(t) = x_1(t) + x_2(t)$

5. Multiplication of continuous time signal:

The Multiplication of two continuous time signals is performed by Multiplying the value of the two signals corresponding to same instant of time.

The product of two $x_1(t)$ and $x_2(t)$ is a signal y(t), whose value at any instant is equal to the product of the value of these two signals at that instant.

i.e. $y(t) = x_1(t) \times x_2(t)$

6. Even and odd signals:

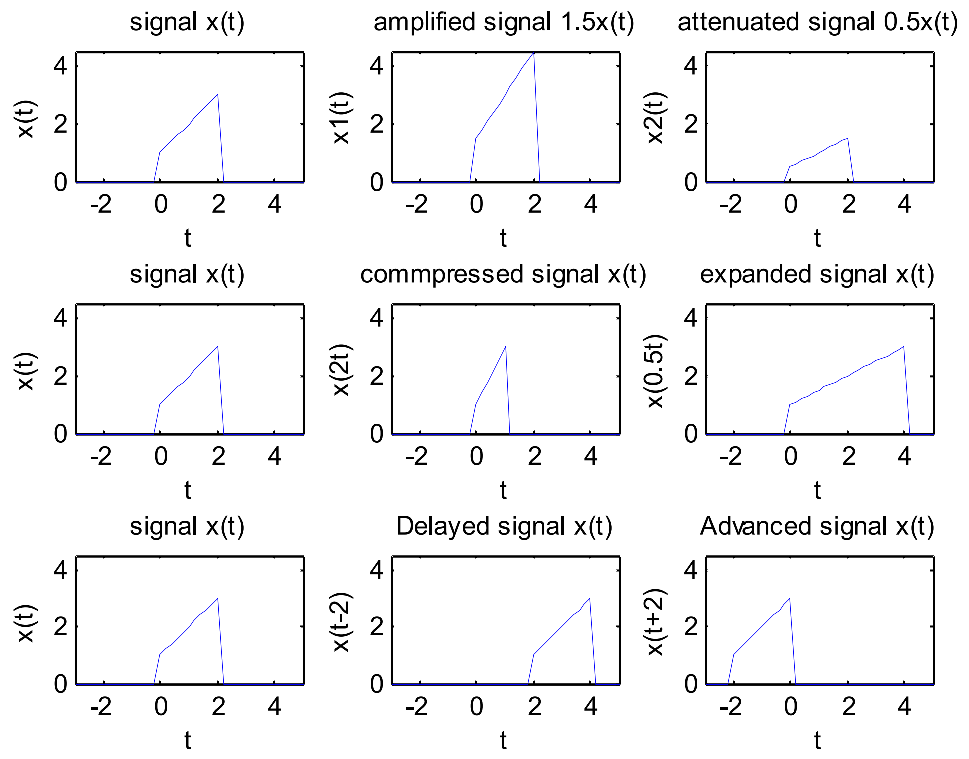
The signals may exhibit symmetry or antisymmetry with respect to t=0.when a signal exhibit symmetry with respect to t=0, then it is called an even signal. Therefore, the even signal satisfies the condition the condition, x(-t) = x(t).

when a signal exhibit antisymmetry with respect to t=0, then it is called an odd signal. Therefore, the odd signal satisfies the condition the condition, x(-t) = -x(t).

MATLAB code

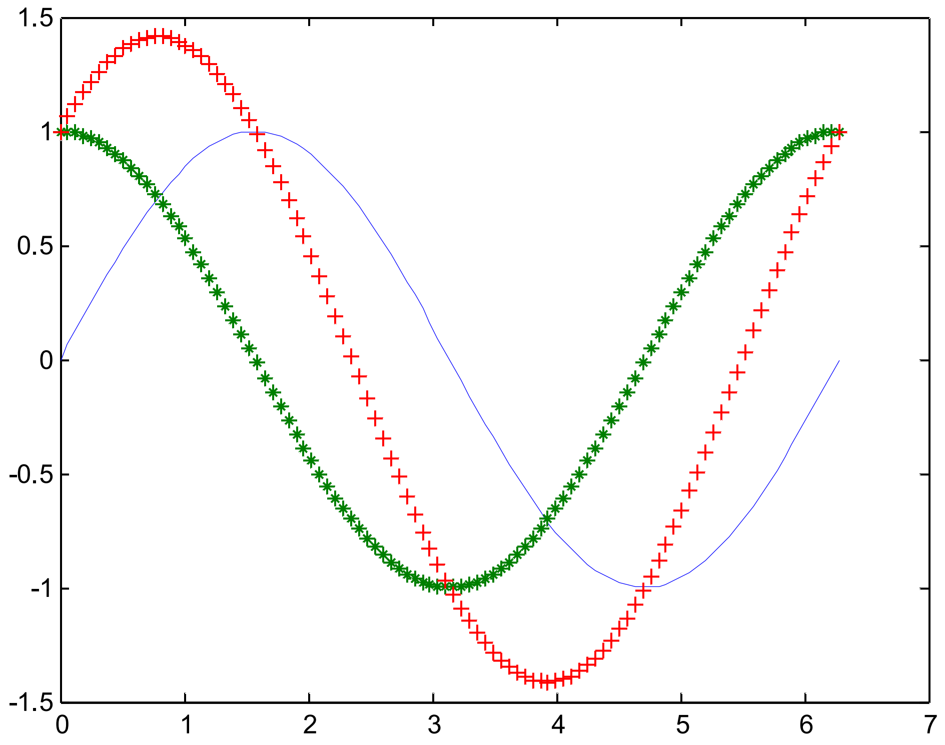
```
        %declare the given signal as function y(t)
function x = y(t)
x=(1.0 + t).*(t>=0 & t<=2);
%to perform amplitude scaling,time scaling and time shifting on
%the signal x(t)=1.0 +t;for t=0 to 2


tmin =-3;
tmax=5;
dt=0.2;
t=tmin:dt:tmax;
```

```
y0 =y(t);
y1 =1.5 * y(t);
y2 =0.5 * y(t);
y3 =y(2*t);
y4 = y(0.5 *t);
y5 =y(t-2);
y6=y(t+2);

%compute the min and max values for y-axis
ymin = min([min(y0),min(y1),min(y2),min(y3),min(y4),min(y5),min(y6)]);
ymax=max ([max(y0),max(y1),max(y2),max(y3),max(y4),max(y4),max(y5),max(y6)]);

%plot the given signal and amplitude scaled signal
subplot(3,3,1);plot(t,y0);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(t)');title('signal x(t)');
subplot(3,3,2);plot(t,y1);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x1(t)');title('amplified signal 1.5x(t)');
subplot(3,3,3);plot(t,y2);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x2(t)');title('attenuated signal 0.5x(t)');

%plot the given signal and time scaled signal
subplot(3,3,4);plot(t,y0);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(t)');title('signal x(t)');
subplot(3,3,5);plot(t,y3);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(2t)');title('commpressed signal x(t)');
subplot(3,3,6);plot(t,y4);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(0.5t)');title('expanded signal x(t)');

%plot the given signal and time shifted signal
subplot(3,3,7);plot(t,y0);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(t)');title('signal x(t)');
subplot(3,3,8);plot(t,y5);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(t-2)');title('Delayed signal x(t)');
subplot(3,3,9);plot(t,y6);axis([tmin,tmax,ymin,ymax]);
xlabel('t');ylabel('x(t+2)');title('Advanced signal x(t)');
```



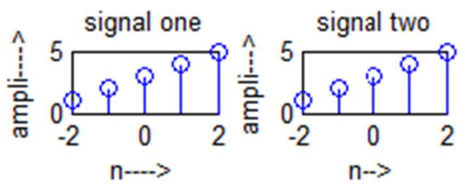
Matlab code for addition and multiplication:

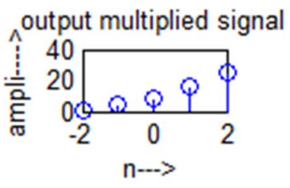
```
x=linspace(0,2*pi,100);

a=sin(x);

b=cos(x);

c=a+b;

y=plot(x,a,x,b,'*',x,c,'+');
```



```
t=-2:1:2;

x1=[1 2 3 4 5 ];

x2=[1 2 3 4 5 ];

subplot(4,4,3);

stem(t,x1);

title('signal one');

ylabel('ampli---->');

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

subplot(4,4,4);

stem(t,x2);

title('signal two');

ylabel('ampli--->');
```

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

y=x1.*x2;

subplot(4,4,5);

stem(t,y);

title('output multiplied signal');

ylabel('ampli---->');

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

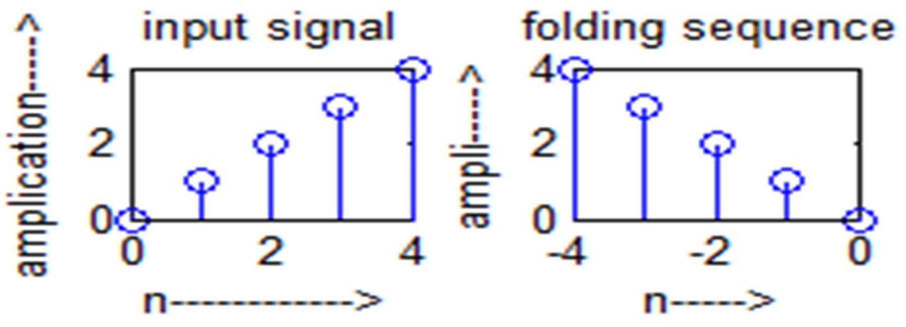




Folding

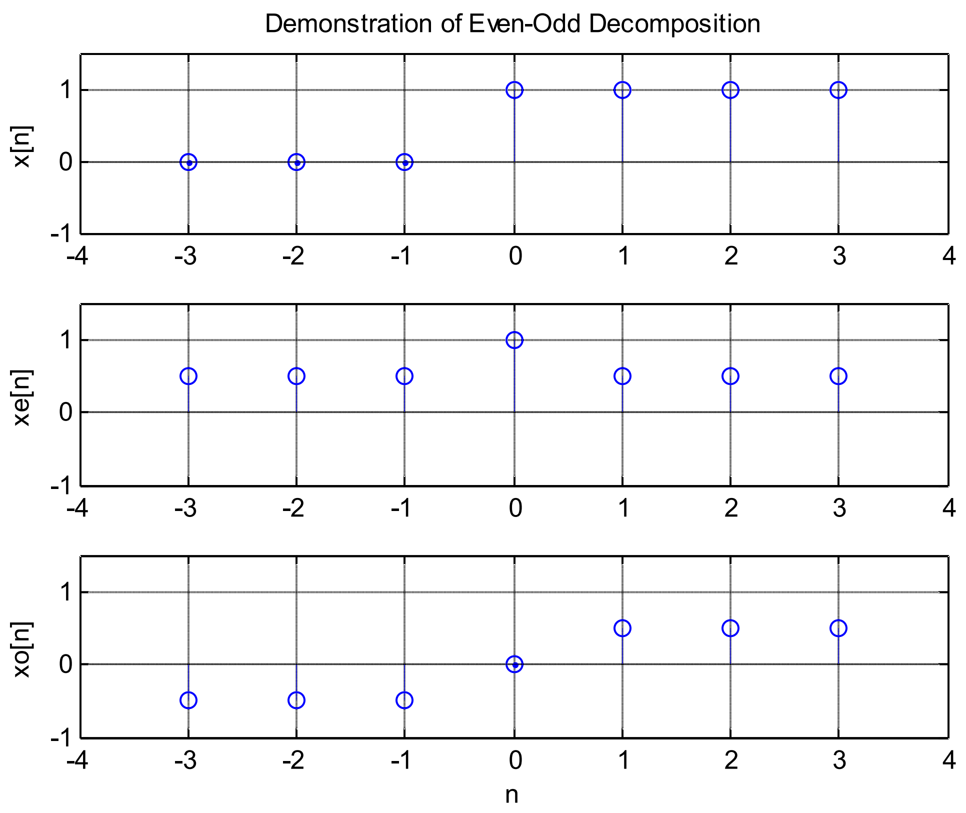
```
t=0:1:4;

x=[0 1 2 3 4];

subplot(4,4,13);

stem(t,x);

title('input signal');

ylabel('amplication----->');

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

hold on;
```

```
t1=t.*-1;

subplot(4,4,14);

stem(t1,x);

hold off;

title('folding sequence');

ylabel('ampli----->');

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



```
n =-3:3;

xn = [0 0 0 1 1 1 1];

xN = [1 1 1 1 0 0 0];

xN = fliplr(xn);

xN = xn(length(xn):-1:1);

xe =(xn+xN)/2;

xo =(xn-xN)/2;

subplot(3,1,1);stem(n,xn);axis([-4 4 -1 1.5])

 grid;

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

```
title('Demonstration of Even-Odd Decomposition')

subplot(3,1,2);stem(n,xe);axis([-4 4 -1 1.5])

grid;ylabel('xe[n]')

subplot(3,1,3);stem(n,xo);axis([-4 4 -1 1.5])

grid;ylabel('xo[n]');xlabel('n')
```



Experiment No.3

MATLAB code to verify Properties of system

Theory:

The systems are classified depending on following properties:

1. Static and dynamic
2. Time invariant and time variant
3. Linear and nonlinear
4. Causal and non-causal
5. Stable and unstable

1. Static and dynamic

A system is said to be ***static*** or ***memoryless*** if the output at any time depends on only the input at that same time. Otherwise, the system is said to be ***dynamic*** or to have ***memory***.

2. **Time invariant and time variant**

A system is called ***time-invariant*** if its input-output characteristics do not change with time i.e. if a time shifts (delay or advance) in the input signal causes the same time shift in the output signal. Thus, for a continuous-time system, the system is time-invariant if

$$\mathbf{T}\,\{\,x(t-\tau)\} = y\,(t-\tau)$$

for any real value of τ. For a discrete-time system, the system is time-invariant (or shift-invariant) if

$$\mathbf{T}\{\,x[n-k]\} = y\,[n-k]$$

for any integer k. if system does not satisfy the above equations then it is said to be time-variant. To check a system for time-invariance, we can compare the shifted output with the output produced by the shifted input.

3. Linear and nonlinear

The system is said to be ***linear*** if it possesses two important properties i.e. additivity and homogeneity (or scaling) properties. In other words, the system is linear if it satisfies the following two conditions

a. Additivity

Given that Tx_1 = y_1 and Tx_2 = y_2 then

$$\mathbf{T}\{x_1 + x_2\} = y_1 + y_2$$

for any signals x_1 and x_2.

b. Homogeneity (or **Scaling)**

$$\mathbf{T}\{\alpha x\} = \alpha y$$

for any signals x and any scalar α.

The system which does not satisfy the above two conditions is said to be ***nonlinear*** system.

4. Causal and non-causal

A system is called ***causal*** if its output ***y (t)*** at an arbitrary time $t = t_0$ depends on only the input ***x (t)*** for $t \leq t_0$. That is, the output of a causal system at the present time depends on only the present and/or past values of the input, not on its future values. Thus, in a causal system, it is not possible to obtain an output before an input is applied to the system. **A** system is called ***non-causal*** if it is not causal. Examples of non-causal systems are

$$y(t) = x(t + 1)$$

$$y[n] = x[-n]$$

Note that all memoryless systems are causal, but not vice versa.

5. Stable and unstable

A system is ***bounded-input/bounded-output (BIBO) stable*** if for any bounded input ***x*** defined by

$$|\boldsymbol{x}| \leq \boldsymbol{k_1}$$

the corresponding output y is also bounded defined by

$$|\boldsymbol{y}| \leq \boldsymbol{k_2}$$

where k_1 and k_2 are finite real constants.

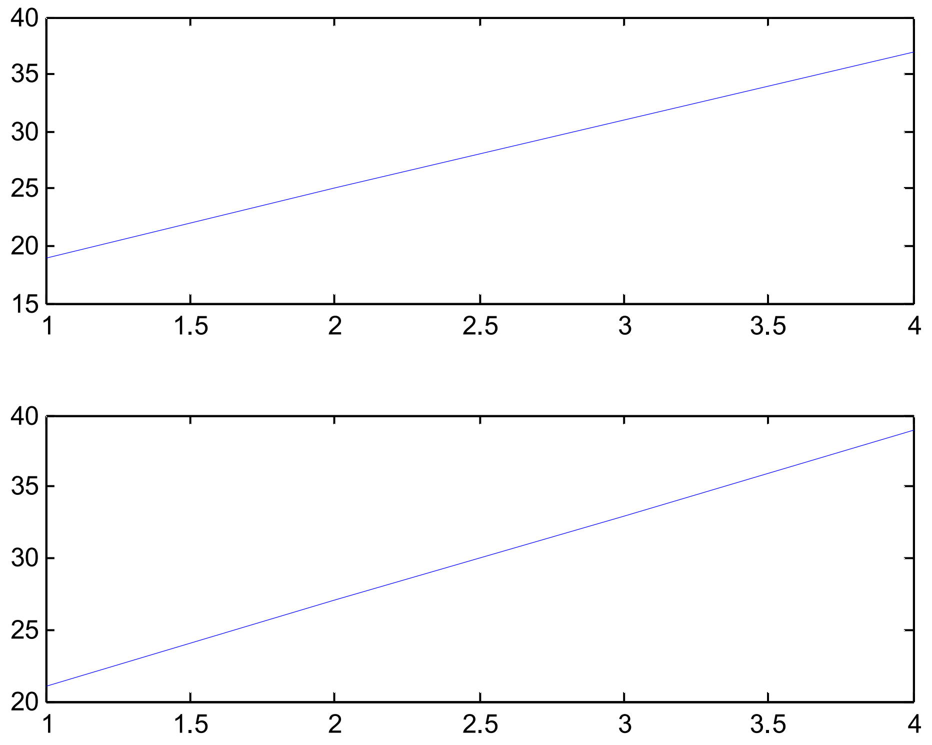
MATLAB Code:

```
% to check whether the given system is linear or not
% y(n)=a*x(n)+b
clear all; close all;
```

```
a=input('enter the coefficient a ');
b=input('enter the constant b ');
x1=input('enter the first input sequence ');
x2=input('enter the second input sequence ');
a1=input('enter the scale factor a1 ');
a2=input('enter the scale factor a2 ');

if length(x1)>length(x2)
   x2=[x2 zeros(1,length(x1)-length(x2))];
else
   x1=[x1 zeros(1,length(x2)-length(x1))];
end

X=a1*x1+a2*x2;
Y=a*X+b;

Y1=a*x1+b;
Y2=a*x2+b;

Y3=a1*Y1+a2*Y2;
if Y==Y3
   disp('given system is linear');
else
   disp('given system is non linear ');
end



subplot(2,1,1);
plot(abs(Y));
subplot(2,1,2);
plot(abs(Y3));
```

Result:

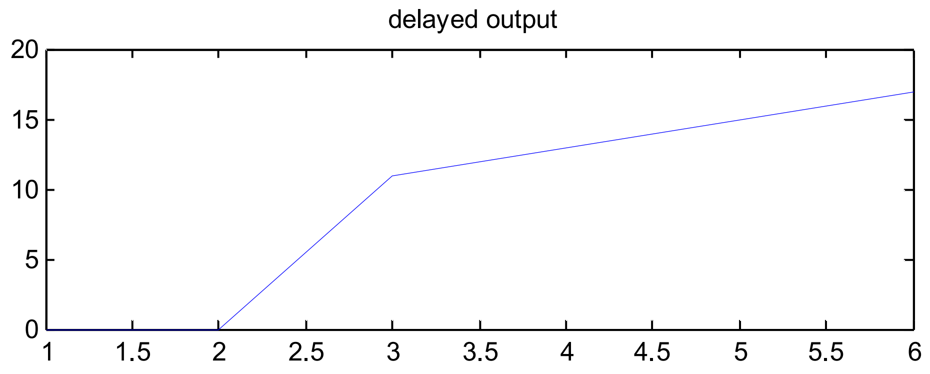
```
enter the coefficient a 2
enter the constant b 1
enter the first input sequence [1 2 3 4]
enter the second input sequence [4 5 6 7]
enter the scale factor a1 1
enter the scale factor a2 2
given system is non linear
```

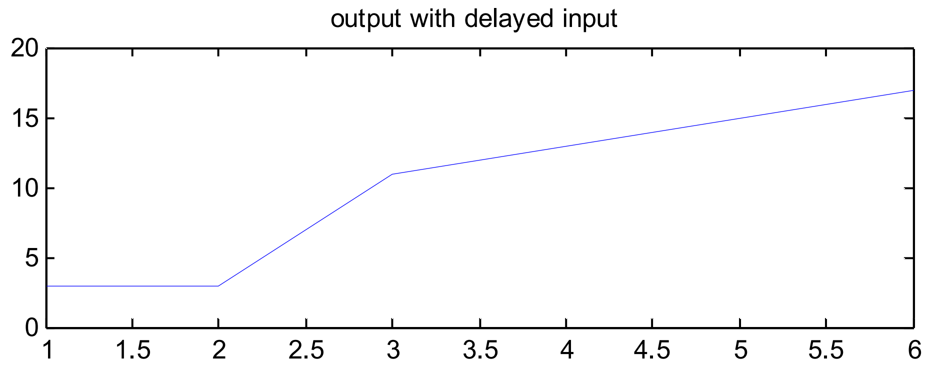


```
2) % to study time invariance property of system
% y(n)=a*x(n)+b
x=input('enter the sewquence ');
d=input('enter the delay factor ');
a=input('enter the coefficient a ');
b=input('enter the constant b ');
xd=[zeros(1,d) x];
y=a*x+b;
yd=[zeros(1,d) y];
y1=a*xd+b;
if y1==yd
    disp('given system is time invariant');
else
    disp('given system is time variant ');
end

subplot(2,1,1);
plot((yd));
title('delayed output ');
subplot(2,1,2);
plot((y1));
title('output with delayed input ');
```

Result:

```
enter the sequence [4 5 6 7]
enter the delay factor 2
enter the coefficient a 2
enter the constant b 3
given system is time variant
```





Experiment No.4

MATLAB code to perform convolution

Theory:

The convolution of two continuous time signals $x_1(t)$ and $x_2(t)$ is defined as,

$$X_3(t) = \int_{\infty}^{-\infty} x_1(\lambda)\, x_2(t-\lambda)\, d\lambda$$

Where, $x_3(t)$ is the signal obtained by convolving $x_1(t)$ and $x_2(t)$.

And λ is a dummy variable used for integration.

The convolution relation of above equation can be symbolically expressed as,

$$X_3(t) = x_1(t) * x_2(t)$$

Where, symbol * indicates convolution operation.

Properties of the Convolution Integral:

The convolution integral has the following properties.

1.**Commutative:**

$$x_1(t) * x_2(t) = x_2(t) * x_1(t)$$

2. **Associative:**

$$[x_1(t) * x_2(t)] * x_3(t) \quad = x_1(t) * [x_2(t) * x_3(t)]$$

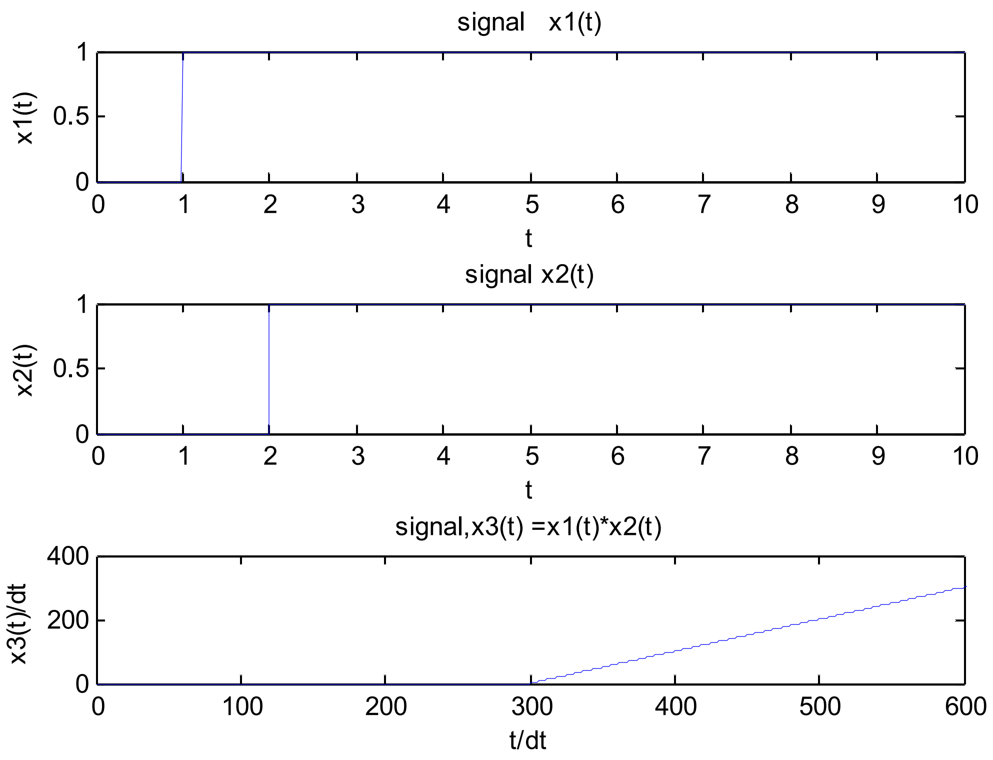
3. **Distributive:**

$$x_1(t) * [x_2(t) + x_3(t)] \quad = [x_1(t) * x_2(t)] + [x_2(t) * x_3(t)]$$

Matlab code:

```
%To perform convolution of the following two signal

tmin=0;  tmax=10; dt=0.01;
t=tmin:dt:tmax;     %set time vector for given signal
x1=1.*(t>=1  &   t<=10);     %generate signal x1(t)
x2=1.*(t>=2   &  t<=10);          %generate signal x2(t)
x3=conv(x1,x2);          %perform covolution of signal x1(t)  and x2(t)
```

```
n3=length(x3);
t1=0:1:n3-1;            %set time vector for x3(t) signal

subplot(3 ,1 ,1);plot(t,x1);
xlabel('t');ylabel('x1(t)');
title('signal   x1(t)');

subplot(3  ,1 ,2);plot(t,x2);
xlabel('t');ylabel('x2(t)');
title('signal x2(t)');


subplot( 3, 1,3);plot(t1,x3); xlim  ([0  600]);
xlabel('t/dt');ylabel('x3(t)/dt');
title('signal,x3(t) =x1(t)*x2(t)');
```



Experiment No. 5

MATLAB code to Verify properties of Fourier Transform

Theory:

PROPERTIES OF THE CONTINUOUS-TIME FOURIER TRANSFORM

Basic properties of the Fourier transform are presented in the following.

1. Linearity

If

$$x_1(t) \overset{F}{\leftrightarrow} X_1(\omega)$$

and

$$x_2(t) \overset{F}{\leftrightarrow} X_2(\omega)$$

then

$$a_1x_1(t) + a_2x_2(t) \overset{F}{\leftrightarrow} a_1X_1(\omega) + a_2X_2(\omega)$$

2. Time Shifting:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$x(t - t_0) \overset{F}{\leftrightarrow} e^{-j\omega t_0}X(\omega)$$

This is known as a ***linear phase shift*** of the Fourier transform $X(\omega)$.

3. Frequency Shifting:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$e^{-j\omega_0 t}x(t) \overset{F}{\leftrightarrow} X(\omega - \omega_0)$$

The multiplication of $x(t)$ by a complex exponential signal $e^{-j\omega_0 t}$ is sometimes called ***complex modulation.*** Thus, the above equation shoes that complex modulation in the time domain corresponds to a shift of $X(\omega)$ in the frequency domain.

4. Time Scaling:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$x(at) \overset{F}{\leftrightarrow} \frac{1}{|a|} X\left(\frac{\omega}{a}\right)$$

where ***a*** is a real constant. This property indicates that scaling the time variable ***t*** by the factor ***a*** causes an inverse scaling of the frequency variable ω by $1/a$ as well as an amplitude scaling of $X(\omega/a)$ by $1\ /\ |a|$. Thus the time scaling property implies that time compression of a signal *(a* > 1) results in its spectral expansion and that time expansion of the signal (a < 1) results in its spectral compression.

5. Time Reversal:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$x(-t) \overset{F}{\leftrightarrow} X(-\omega)$$

Thus, time reversal of $x(t)$ produces a like reversal of the frequency axis for $X(\omega)$.

6. Duality (or Symmetry):

Because of the similarity between the formulae for the Fourier transform and its inverse, a correspondence between the frequency-domain and time-domain can be derived by carefully choosing the arguments of a given transform pair. With appropriate substitutions into the integrals for the transform and its inverse, it can be shown that :
If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$X(t) \overset{F}{\leftrightarrow} 2\pi x(\omega)$$

This relationship is called ***duality.***

7. Differentiation in the Time Domain:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$\frac{dx(t)}{dt} \overset{F}{\leftrightarrow} j\omega X(\omega)$$

The above equation shows that the effect of differentiation in the time domain is the multiplication of $X(\omega)$ by $j\omega$ in the frequency domain.

8. Differentiation in the Frequency Domain:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$(-jt)x(t) \overset{F}{\leftrightarrow} \frac{dX(\omega)}{d\omega}$$

9. Integration in the Time Domain:

If

$$x(t) \overset{F}{\leftrightarrow} X(\omega)$$

then

$$\int_{-\infty}^{t} x(\tau)d\tau \overset{F}{\leftrightarrow} \pi X(0)\delta(\omega) + \frac{1}{j\omega}X(\omega)$$

Since integration is the inverse of differentiation, above equation shows that the frequency domain operation corresponding to time-domain integration is multiplication by $1/j\omega$, but an additional term is needed to account for a possible dc component in the integrator output. Hence, unless $X(0) = 0$, a dc component is produced by the integrator.

10. Convolution:

If

$$x_1(t) \overset{F}{\leftrightarrow} X_1(\omega)$$

and

$$x_2(t) \overset{F}{\leftrightarrow} X_2(\omega)$$

then

$$x_1(t) * x_2(t) \overset{F}{\leftrightarrow} X_1(\omega)X_2(\omega)$$

The above equation is referred to as the time convolution theorem, and it states that convolution in the time domain becomes multiplication in the frequency domain.

11. Multiplication:

If

$$x_1(t) \overset{F}{\leftrightarrow} X_1(\omega)$$

and

$$x_2(t) \overset{F}{\leftrightarrow} X_2(\omega)$$

then

$$x_1(t)x_2(t) \overset{F}{\leftrightarrow} \frac{1}{2\pi} X_1(\omega) * X_2(\omega)$$

The multiplication property is the dual property of convolution and is often referred to as the frequency convolution theorem. Thus, multiplication in the time domain becomes convolution in the frequency domain.

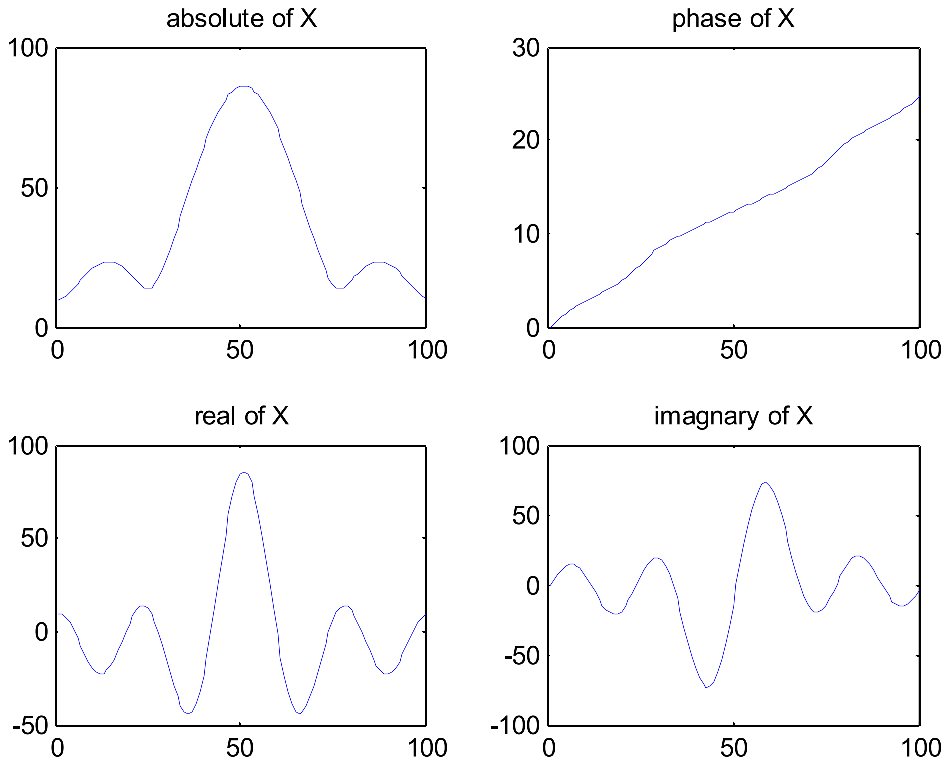
MATLAB code:

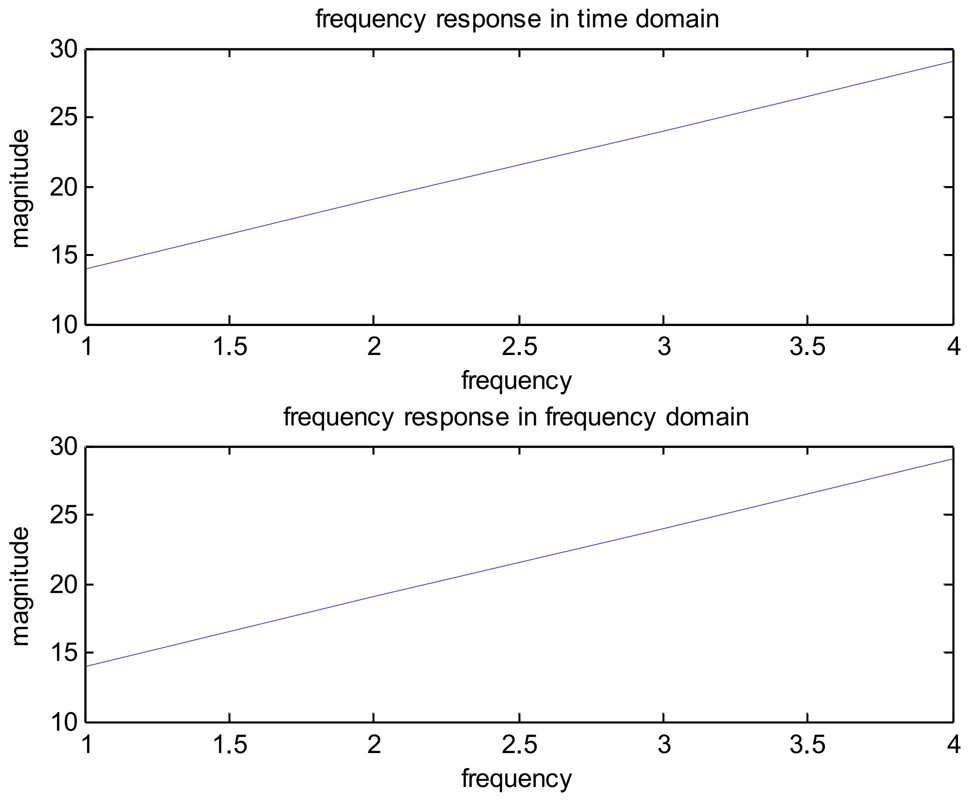
```
%Fourier Transform%
function[x]=Fourier_Transform(x);

N=length(x)
w=-pi
for i=1:1:100
   n=1:1:N
   e=exp(-j*w*n)
   xft(i)=x*e'
   w=w+(2*pi)/100
end
subplot(2,2,1)
plot(abs(xft))
title('absolute of X')
subplot(2,2,2)
plot(phase(xft))
title('phase of X')
subplot(2,2,3)
plot(real(xft))
title('real of X')
subplot(2,2,4)
plot(imag(xft))
title('imagnary of X')
```

```
1. %to study linearity property of Fourier transform
x1=input('enter the first sequence ');
x2=input('enter the second sequence ');
a1=input('enter the scaling factor a1 ');
a2=input('enter the scaling factor a2 ');


if length(x1)>length(x2)
   x2=[x2 zeros(1,length(x1)-length(x2))];
```

```
else
   x1=[x1 zeros(1,length(x2)-length(x1))];
end

x3=a1*x1+a2*x2;

X1=Fourier_Transform(x1);
X2=Fourier_Transform(x2);
X3=Fourier_Transform(x3);

X4=a1*X1+a2*X2;

figure;
subplot(2,1,1);
plot(abs(X3));
xlabel('frequency');
ylabel('magnitude');
title('frequency response in time domain');
subplot(2,1,2);
plot(abs(X4));
xlabel('frequency');
ylabel('magnitude');
title('frequency response in frequency domain');
```

Result:

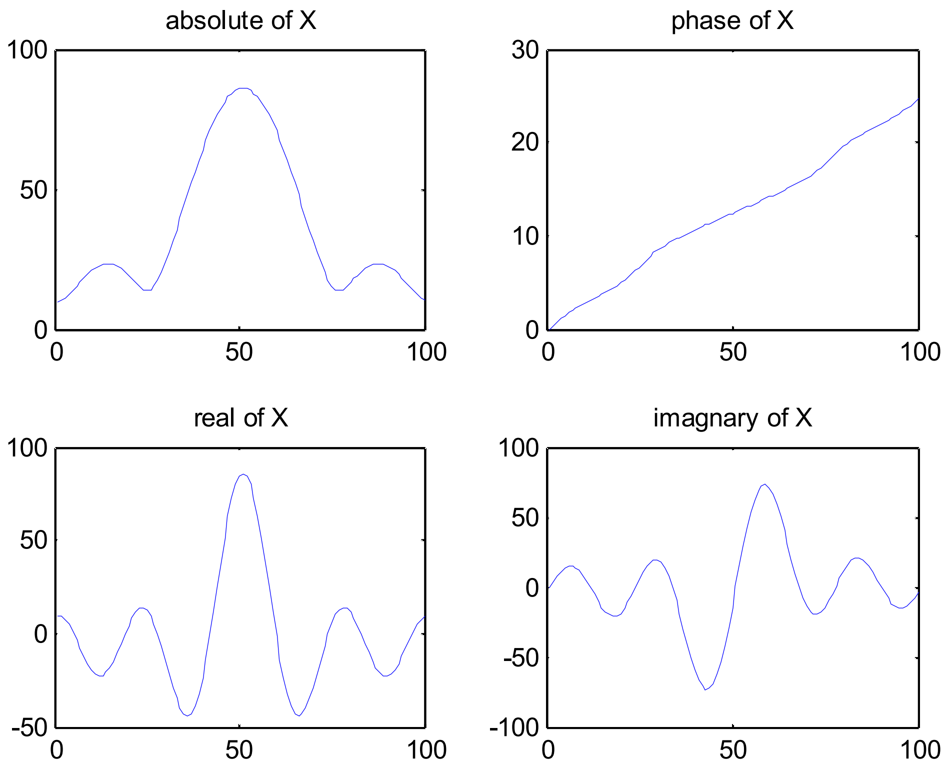


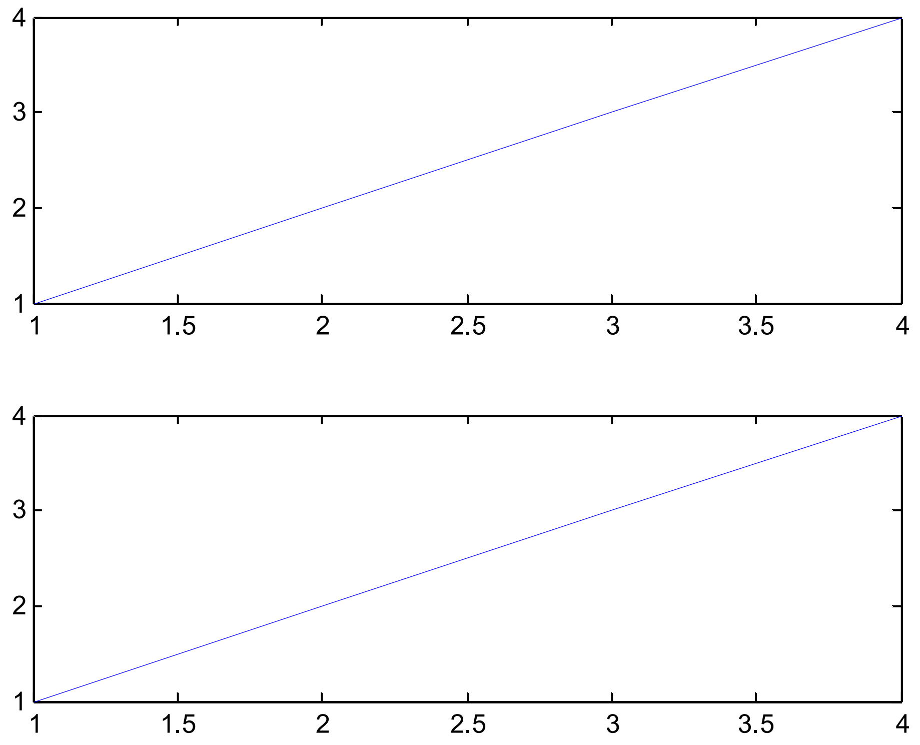


2. % to study time shift property of Fourier transform

```
x1=input('enter the sequence ');
d=input('enter the delay factor ');
xd=[zeros(1,d) x1];
X=Fourier_Transform(x1);
for t=1:1:length(xd)
e=exp(-j*d*t);
Xd=x*e';
end
subplot(2,1,1);
plot(abs(X));
subplot(2,1,2);
plot(abs(Xd));
```

Result:



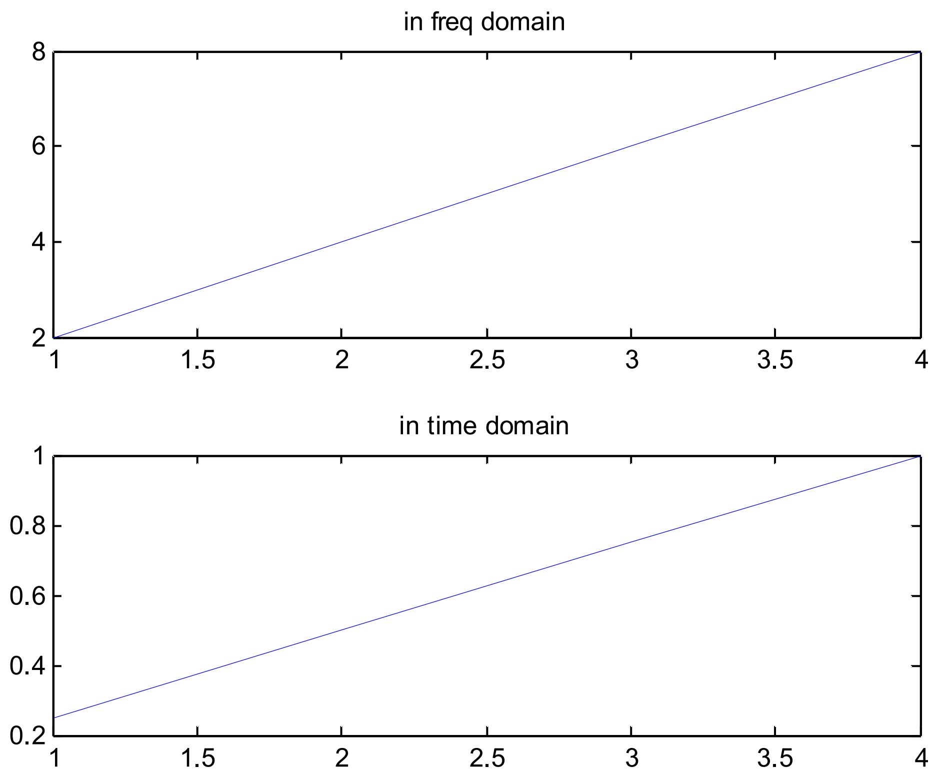


```
3. % to study frequency shift property of Fourier transform
clear all; close all;
x1=input('enter the sequence ');
w0=input('enter the frequency shift ');
X1=Fourier_Transform(x1);
X2=[zeros(1,w0) X1];
for t=1:1:length(x1)
    e=exp(j*w0*t);
    x2=e'*x1;
end
X3=Fourier_Transform(x2);
subplot(2,1,1);
plot((X2));
subplot(2,1,2);
plot((X3));
```

Result:

```
4) % to study time scale property of Fourier transform
x1=input('enter the sequence ');
a=input('enter the scale factor ');
xa=x1.*a;
Xa=Fourier_transform(xa);
X1=Fourier_transform(x1);
X2=X1/a;
X3=(1/abs(a))*X2;
subplot(2,1,1);
plot((Xa));
title('in freq domain');
subplot(2,1,2);
plot((X3));
title('in time domain');
```

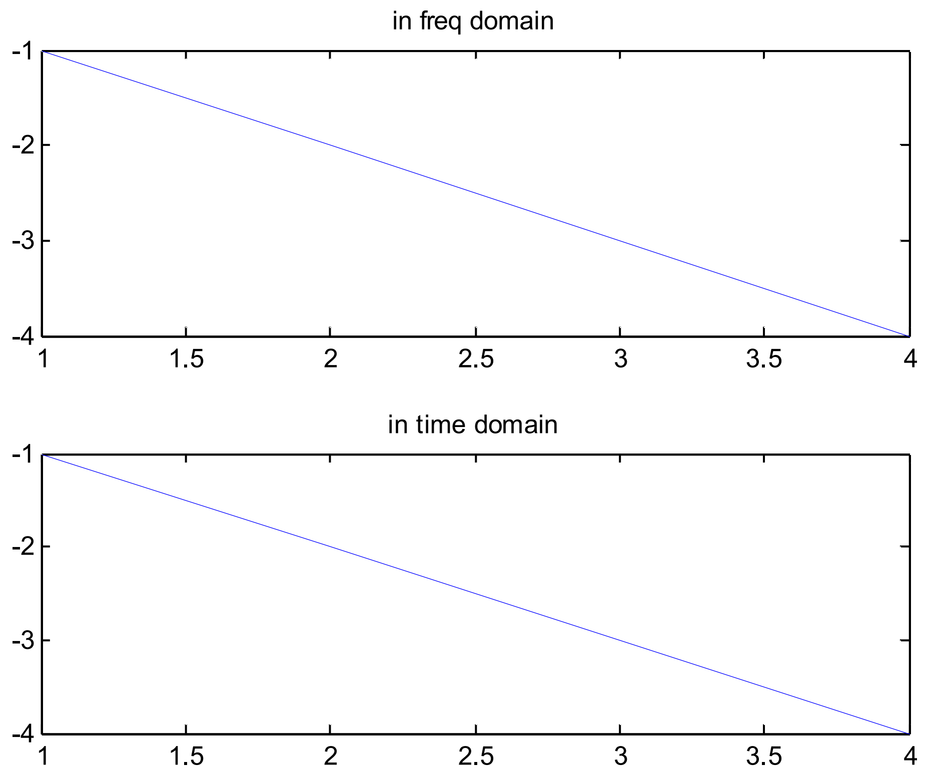
Result:



```
5) % to study time reversal property of Fourier transform
x1=input('enter the sequence ');
a=-1;
xa=x1.*a;
Xa=Fourier_transform(xa);
X1=Fourier_transform(x1);
```

```
X2=X1/a;
X3=(1/abs(a))*X2;
subplot(2,1,1);
plot((Xa));
title('in freq domain');
subplot(2,1,2);
plot((X3));
title('in time domain');
```

Result:



Experiment No. 6

MATLAB program to find one sided Z-transform of Standard causal signals

Theory:

Let x(n) be a discrete time signal defined in the range $0 < n < \infty$ .And X(z) is Z-transform of x(n).

The z-transform of a discrete time signal,x(n) is defined as,

$$X(z) = \sum_{n=-\infty}^{\infty} x(n) \quad Z^{-n}$$

Where, z is a complex variable.

MATLAB CODE:

```
% program to find the z-transform of some standard signals

clear all;
syms n T a real;
syms z complex;

%(a) n
x=n;
disp('(a) z-transform of " n " is');
ztrans(x)

%(b) a^n
x= a^n;
disp('(b) z-transform of "a^n" is ');
ztrans(x)


%(c) n*(a^n)

x=n*(a^n);
disp('(c) z-transform of "n*(a^n)" is ');
ztrans(x)

%(d) exp(-a*n*T)
x=exp(-a*n*T)
disp('(d) z-transform of "exp(-a*n*T)" is ');
ztrans(x)
```

RESULT:

```
(a) z-transform of " n " is

ans =

z/(z - 1)^2

(b) z-transform of "a^n" is

ans =

-z/(a - z)

(c) z-transform of "n*(a^n)" is

ans =

z/(a*(z/a - 1)^2)


x =

1/exp(T*a*n)

(d) z-transform of "exp(-a*n*T)" is

ans =

z/(z - 1/exp(T*a))
```

Experiment No. 7

MATLAB code to find residues and poles of Z-domain signal

Theory:

Let X(z) be Z-transform of x(n).when X(z) is expressed as ratio of two polynomial in z or z^{-1}, then X(z) is called a rational function of z.

Let X(z) be expressed as a ratio of two polynomials in z, as shown below.

$$X(z) = \frac{N(z)}{D(z)}$$

$$=b_0+b_1z^{-1} + b_2z^{-1} + b_3z^{-1}+\ldots\ldots\ldots+ b_Mz^{-1}/a_0 + a_1z^{-1} + a_2z^{-1}+ a_3z^{-1}+\ldots\ldots\ldots+a_Nz^{-1}$$

Where, N(z) =Numerator Polynomial of X(z)

D(z)=Denominator Polynomial of X(z).

Let scale the coefficient of numerator polynomial by b_0 and that of denominator polynomial by a_0 and then convert the polynomial to positive power of Z

$$X(z) =G (z-z_1) (z-z_2)(z-z_3)\ldots\ldots\ldots(z-z_N)/(z-p_1)(z-p_2)(z-p_3)\ldots\ldots\ldots(z-z_N)$$

Where,$z_1,z_2,z_3\ldots\ldots\ldots z_N$ are roots of numerator polynomial

$p_1,p_2,p_3\ldots\ldots\ldots,p_N$ are roots of denominator polynomial

G is scaling factor.

In above equation ,if the value of z is equal to one of the roots of numerator polynomial, then the function X(z) will become zero.

Therefore the roots of numerator polynomial $z_1,z_2,z_3\ldots\ldots\ldots z_N$ are called zeros of X(z). Hence the zeros are defined as values z at which the function X(z) become zero.

In above equation ,if the value of z is equal to one of the roots of denominator polynomial, then the function X(z) will become infinite.

Therefore the roots of denominator polynomial $p_1,p_2,p_3\ldots\ldots\ldots,p_N$ are called poles of X(z). Hence the poles are defined as values z at which the function X(z) become infinite.

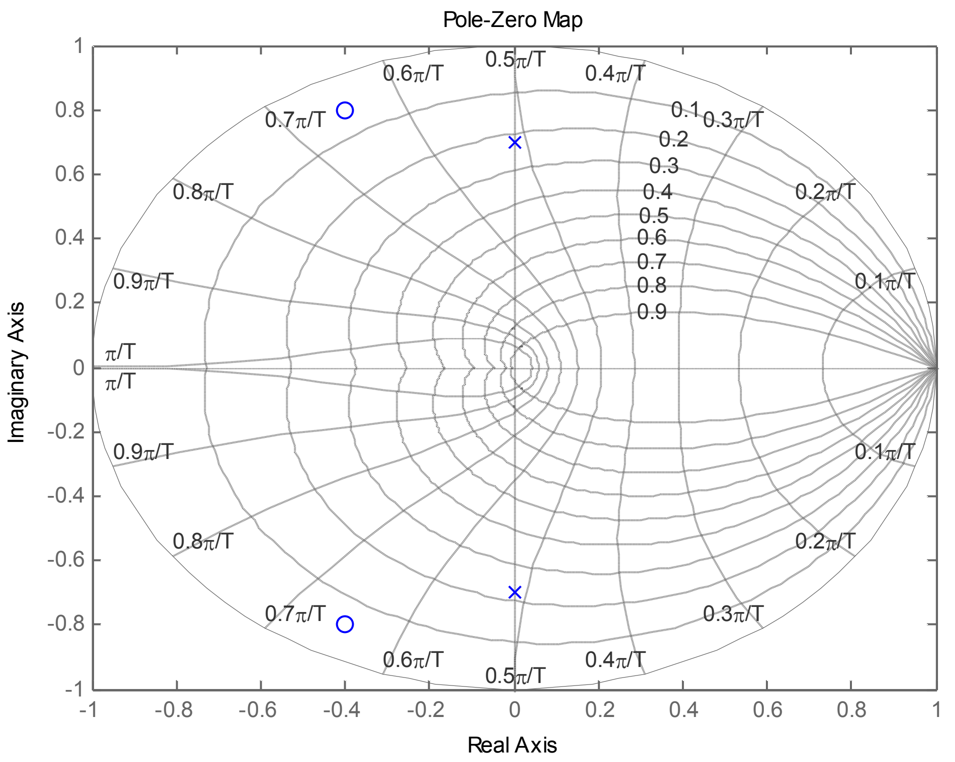
MATLAB CODE:

```
%program to determine poles and zeros of rational function of z and
%to plot the poles and zeros in z-plane
```

```
clear all;
syms z
num_coeff=[1 0.8 0.8];
disp('roots of numerator polynomial z^2+0.8Z+0.8 are zeros');
zeros=roots (num_coeff)

den_coeff=[1 0 0.49];
disp('roots of denominators polynomial z^2+0.49 are poles .');
poles=roots(den_coeff)

H=tf('z');
Ts=0.1;

H=tf([num_coeff],[den_coeff],Ts);
zgrid on;
pzmap(H);
```

RESULT:

```
roots of numerator polynomial z^2+0.8Z+0.8 are zeros


zeros =


  -0.4000 + 0.8000i
  -0.4000 - 0.8000i


roots of denominators polynomial z^2+0.49 are poles .


poles =


        0 + 0.7000i
        0 - 0.7000i
```


Pole-Zero Map
Imaginary Axis
Real Axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
0.5π/T
0.6π/T
0.4π/T
0.7π/T
0.3π/T
0.8π/T
0.2π/T
0.9π/T
0.1π/T
π/T
π/T
0.9π/T
0.1π/T
0.8π/T
0.2π/T
0.7π/T
0.3π/T
0.6π/T
0.4π/T
0.5π/T
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9

Experiment No. 8

MATLAB code to find Laplace transform of Standard causal signal**.**

Theory:

In order to perform a time domain signal x(t) to s-domain,multiply the signal by e^{-st} and then integrate from -∞ to ∞.The transformed signal is represented as X(s) and the transformation is denoted by the script letter $\mathcal{L}$.

By Definition of Laplace Transform,

$$\mathcal{L}\,\{x(t)\} = X(s) = \int_{-\infty}^{+\infty} x(t)\, e^{-st}\, dt.$$

MATLAB CODE:

```
%*****program to find laplace transform of some std signals

clear all;
syms t real;
syms a real;
syms s complex;
syms n real;

%a
x=t;
disp('(a) laplace transform of "t" is');
laplace(x)

%b
x=exp(-a*t);
disp('(b) laplace transform of "exp(-a*t)"is');
laplace(x)

%c
x=t*exp(-a*t);
disp('(c) laplace transfrom of "t*exp(-a*t)" is');
laplace(x)

%d
sg=real(s);
o=imag(s);
s=sg+(i*o);
x=cos(o*t);
disp('(d) laplace transform of "cos(o*t)" is');
laplace(x)
```

```
%e
x=sinh(o*t);
disp('(e) laplace transform of "sinh(o*t)" is ');
laplace(x)

%f
x=exp(-a*t)*sin(o*t);
disp('(f) laplace transform of "exp(-a*t)*sin(o*t" is ');
laplace(x)

%g
n=input('enter the value of n');


x=t^n;
disp('(g) laplace transform of "t^n" is ');
laplace(x)

%h
x=t^(n-1)/factorial (n-1);
disp('(h) laplace transform of "t^(n-1)/factorial (n-1)" is');
laplace(x)

%i
x=(t^n)*exp(-a*t);
disp('(i) laplace transform of "(t^n)*exp(-a*t)" is');
laplace(x)

%j
x=(t^(n-1)*exp(-a*t))/factorial (n-1);
disp('(j) laplace transform of "(t^(n-1)*exp(-a*t))/factorial (n-1" is');
laplace(x)
```

RESULT:

```
(a) laplace transform of "t" is

ans =

1/s^2

(b) laplace transform of "exp(-a*t)"is

ans =

1/(a + s)
```

```
(c) laplace transfrom of "t*exp(-a*t)" is

ans =

1/(a + s)^2

(d) laplace transform of "cos(o*t)" is

ans =

s/(((i*s)/2 - (i*conj(s))/2)^2 + s^2)

(e) laplace transform of "sinh(o*t)" is

ans =

((i*s)/2 - (i*conj(s))/2)/(((i*s)/2 - (i*conj(s))/2)^2 - s^2)

(f) laplace transform of "exp(-a*t)*sin(o*t" is

ans =

-((i*s)/2 - (i*conj(s))/2)/((a + s)^2 + ((i*s)/2 - (i*conj(s))/2)^2)

enter the value of n 3
(g) laplace transform of "t^n" is

ans =

6/s^4

(h) laplace transform of "t^(n-1)/factorial (n-1)" is

ans =

1/s^3

(i) laplace transform of "(t^n)*exp(-a*t)" is

ans =

6/(a + s)^4

(j) laplace transform of "(t^(n-1)*exp(-a*t))/factorial (n-1" is
ans =

1/(a + s)^3
```

Experiment No. 9

MATLAB code for convolution using Fourier transforms.

Theory:

The convolution theorem of Fourier transform says that, Fourier transform of convolution of two signals is given by the product of the Fourier Transform of the individual signals .

i.e. if $F\{x_1(t)\} = X_1(j\Omega)$ and $F\{x_2(t)\} = X_2(j\Omega)$ then,

$$F\{ x_1(t) * x_2(t)\} = X_1(j\Omega)X_2(j\Omega)$$

This equation is also known as convolution property of Fourier transform.

MATLAB CODE:

```
%******program to perform convolution using fourier transform

syms t real;

x1=exp(-2*t).*heaviside(t);

x2=exp(-6*t).*heaviside(t);


disp('Fourier transform of x1(t) is');

x1=fourier(x1)

disp('Fourier transfrom of x2(t) is');

x2=fourier(x2)

y=x1*x2;


disp('let x3 be convolution of x1(t) and x2(t)');

x3=ifourier(y,t)
```

result:

Fourier transform of x1(t) is

```
x1 =

1/(i*w + 2)
```

Fourier transfrom of x2(t) is

```
x2 =

1/(i*w + 6)
```

let x3 be convolution of x1(t) and x2(t)

```
x3 =

((pi*heaviside(t))/(2*exp(2*t)) - (pi*heaviside(t))/(2*exp(6*t)))/(2*pi)
```

Experiment No. 10

MATLAB code for convolution using Laplace-transform

Theory:

The convolution theorem of Laplace transform saya that,Laplace transform of convolution of two signals is given by the product of the Laplace Transform of the individual signals .

i.e. if $L\{x_1(t)\} = X_1(s)$ and $L\{x_2(t)\} = X_2(s)$ then,

$$L\{ x_1(t) * x_2(t)\} = X_1(s)X_2(s)$$

MATLAB CODE:

```
%********program for convolution using laplace transform

clear all;

syms t real;

x1t=(t^2-(2*t));

x2t=t;


x1s=laplace (x1t);

x2s=laplace(x2t);

x3s=x1s*x2s;

conl2=ilaplace(x3s);

disp('convolution of x1(t) and x2(t) is ');

simplify(conl2)
```

result:

```
convolution of x1(t) and x2(t) is


ans =
(t^3*(t - 4))/12
```

MATLAB code:

```
%**************program to perform convolution using Z-transform
```

```
clear all;
syms n z
x1n=0.4^(n);
x2n=0.5^(n);
x1z=ztrans(x1n);
x2z=ztrans(x2n);
x3z=x1z*x2z;
conl2=iztrans(x3z);
disp('convolution of x1(n) and x2(n) is');
simplify(conl2)
```

Result:

```
convolution of x1(n) and x2(n) is


ans =


5*(1/2)^n - 4*(2/5)^n
```

Experiment No. 10

MATLAB code for convolution using Laplace-transform

Theory:

The convolution theorem of Laplace transform saya that,Laplace transform of convolution of two signals is given by the product of the Laplace Transform of the individual signals .

i.e. if $L\{x_1(t)\} = X_1(s)$ and $L\{x_2(t)\} = X_2(s)$ then,

$$L\{ x_1(t) * x_2(t)\} = X_1(s)X_2(s)$$

MATLAB CODE:

```
%********program for convolution using laplace transform

clear all;

syms t real;

x1t=(t^2-(2*t));

x2t=t;


x1s=laplace (x1t);

x2s=laplace(x2t);

x3s=x1s*x2s;

conl2=ilaplace(x3s);

disp('convolution of x1(t) and x2(t) is ');

simplify(conl2)
```

result:

```
convolution of x1(t) and x2(t) is


ans =
(t^3*(t - 4))/12
```

MATLAB code:

```
%**************program to perform convolution using Z-transform
```

```
clear all;

syms n z

x1n=0.4^(n);

x2n=0.5^(n);

x1z=ztrans(x1n);

x2z=ztrans(x2n);

x3z=x1z*x2z;

conl2=iztrans(x3z);

disp('convolution of x1(n) and x2(n) is');

simplify(conl2)
```

Result:

```
convolution of x1(n) and x2(n) is


ans =


5*(1/2)^n - 4*(2/5)^n
```