SIMATS - ENGINEERING
SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

Engineer to Excel

ECA03 SIGNALS AND SYSTEMS
LAB MANUAL

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

SAVEETHA SCHOOL OF ENGINEERING
CHENNAI – 602 105

INDEX

18	**POLE ZERO PLOT**	
19	**Z-TRANSFORM OF THE GIVEN SIGNAL**	
20	**CONVOLUTION OF DISCRETE TIME SIGNALS**	
21	**DISCRETE FOURIER TRANSFORM**	
22	**CIRCULAR CONVOLUTION**	
23	**ANALYSIS OF DISCRETE-TIME SYSTEM**	
24	**STEP RESPONSE OF THE SYSTEM**	

EXP. NO. 1

GENERATION OF CONTINUOUS TIME SIGNALS

AIM:- To generate the given continuous time signals.

APPARATUS REQUIRED:- PC and SCILAB Software.

ALGORITHM:-

Assume t

(a)Unit step signal:

$u(t)= 1$ for $t \geq 0$

0 for $t< 0$

(b)Ramp signal :

$r(t) = t$ for $t \geq 0$

0 for $t < 0$

(C) Unit impulse function:

$\delta(t)= 0$ for $t \neq 0$

1 for $t = 0$

(d) exponential sequence :

x(t) =Ae^at

If a + ve→ growing exponential

If a - ve→decaying exponential

If a = 0→constant

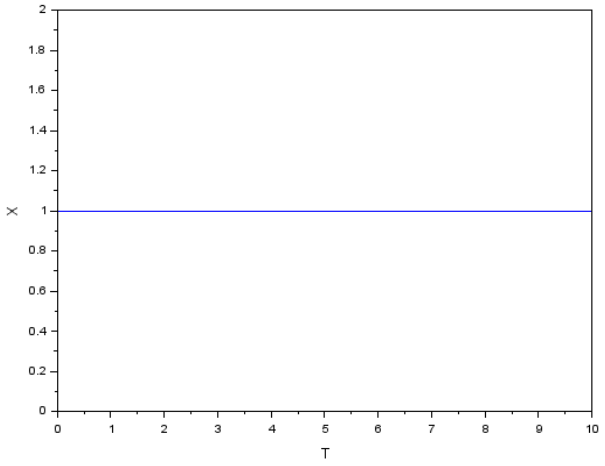
(e) sinusoidal :

y=Y sin wt

PROGRAM:

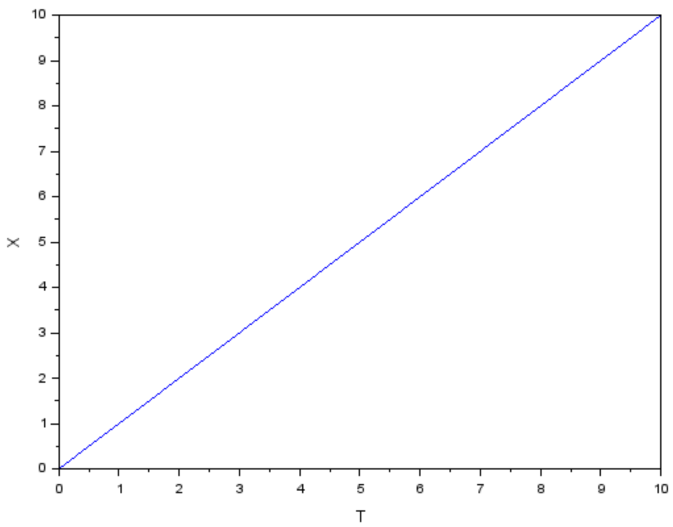
(a) Unit step signal x(t)=u(t)

```
clear;
clc;
t=0:1/100:10;
x=ones(length(t),1);
plot(t,x)
xlabel("T")
ylabel("X")
```



(b) Ramp signal x(t)=tu(t)

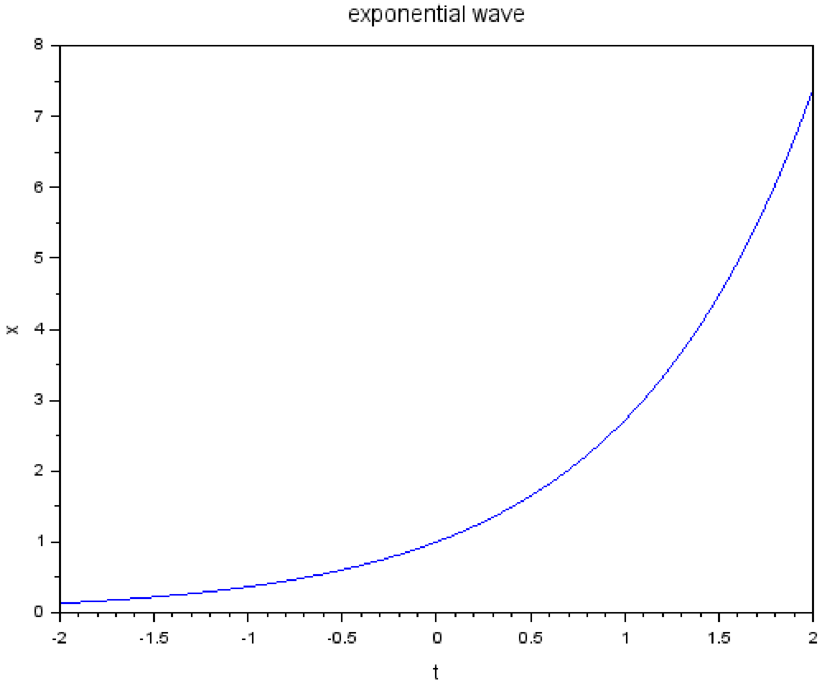
```
clc;
t=0:1/100:10
x=t
plot(t,x)
xlabel("T")
ylabel("X")
```



(c) Exponential function

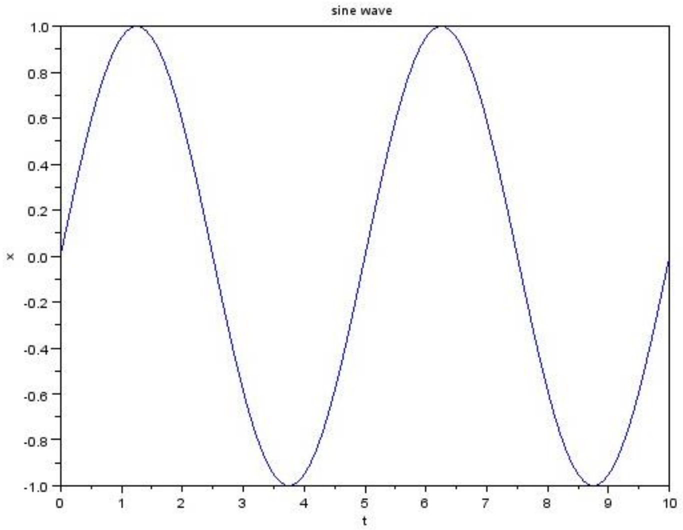
```
clc ;
clf ;
clear all;
// Caption : generation of exponential wave
t = -2:0.1:2;
x = exp (t) ;
plot(t,x) ;
```

```
title("exponential wave");
xlabel("t")
ylabel("x")
```



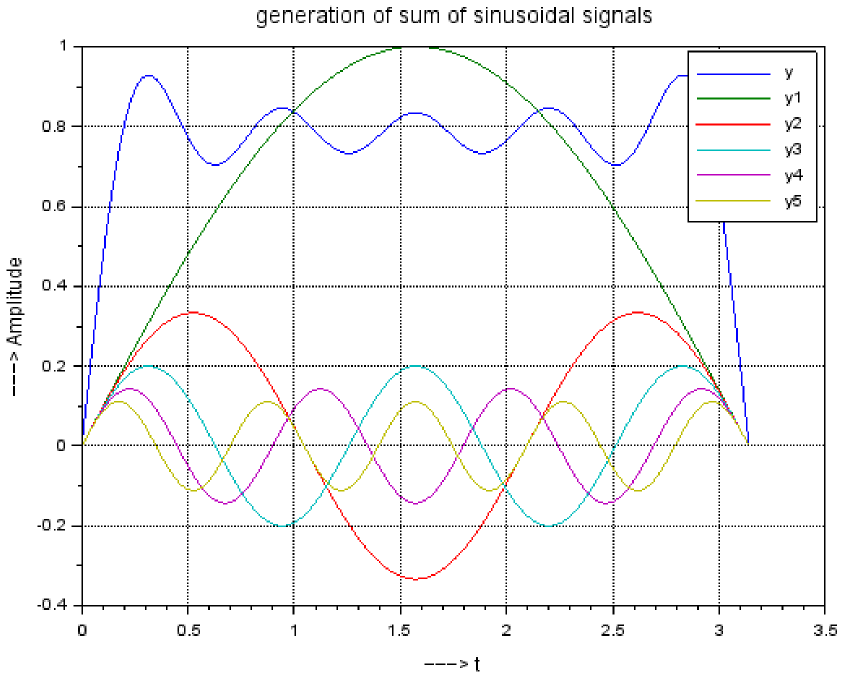
(d) Sinusoidal function

```
clc ;
clf ;
clear all;
// g e n e r a t i o n o f s i n e wave
f =0.2;
t =0:0.1:10;
x= sin (2* %pi *t*f);
plot (t,x);
title ( ' s i n e wave ' );
xlabel ( ' t ' );
ylabel ( ' x ' );
```



e. Sum of sinusoidal signals

```
clc ;
clear all;
tic ;
t =0:.01: %pi ;
// generation of sine signals
y1=sin(t) ;
y2=sin (3* t) /3;
y3=sin (5* t) /5;
y4=sin (7*t) /7;
y5=sin (9* t) /9;
y = sin (t) + sin (3*t)/3 + sin (5*t)/5 + sin (7*t) /7 + sin (9*t) /9;
plot (t ,y ,t , y1 ,t , y2 ,t , y3 ,t , y4 ,t , y5) ;
legend ( "y", "y1" , "y2" ,"y3" ,"y4" , "y5") ;
title ("generation of sum of sinusoidal signals") ;
xgrid (1) ;
ylabel ( "−−−> Amplitude") ;
xlabel ( "−−−> t ") ;
toc ;
```



<u>RESULT</u>:

Thus the given continous time signal has been generated.

EXP. NO. - 2

GENERATION OF DISCRETE TIME SIGNALS

AIM:- To generate the given discrete time signals.

APPARATUS REQUIRED:- PC and SCILAB Software.

ALGORITHM:-

a) Unit Step Signal :
   u(n)=1 for n $\geq$ 0
   0 for n< 0

b) Ramp signal :
   r(n) = n for n $\geq$ 0
   0 for n < 0

c) Unit impulse function:
   δ(n)= 0 for n $\neq$ 0
   1 for n = 0

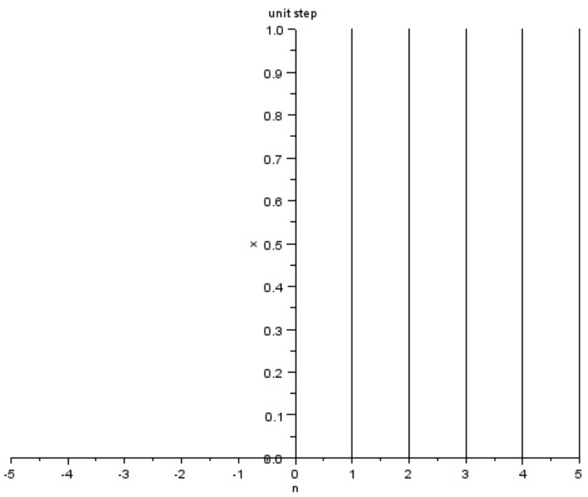
d) exponential sequence :
   x(n) =Ae^at
   If a + ve$\rightarrow$ growing exponential
   If a - ve$\rightarrow$decaying exponential
   If a = 0$\rightarrow$constant

e)sinusoidal :
   y=Y sin wt

PROGRAM

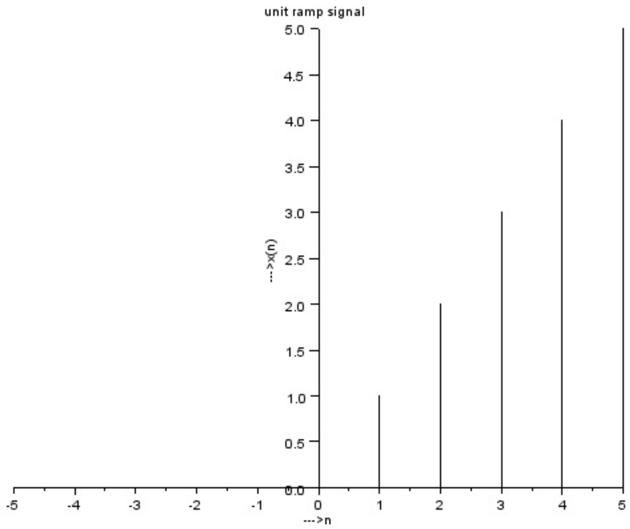
(a)

```
 n = -10:10;
// Range of time indices
x = sign(n >= 0);
// Unit step signal
stem(n, x)
xlabel('n')
ylabel('Amplitude')
title('Unit Step Signal')
grid on
```



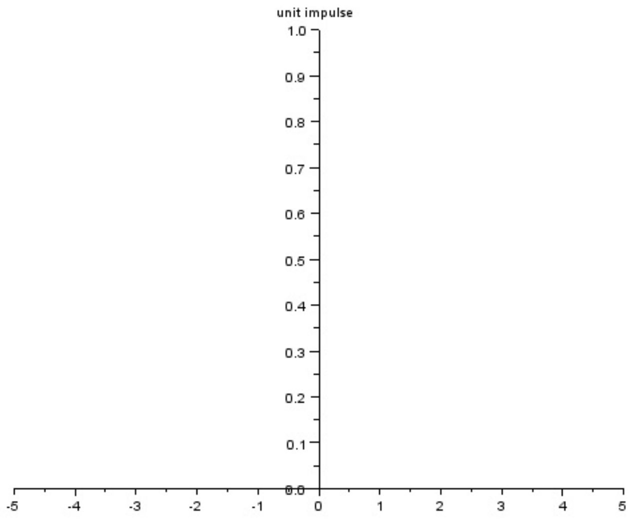
(b) Ramp signal;

```
  // unit ramp
  clc ;
  clf ;
  clear all;
  L =5;
  n = - L : L;
  x =[ zeros (1 , L ) ,0: L ];
  a = gca () ;
 . y_location = ’ middle ’ ;
 plot2d3 (n ,x ) ;
 xtitle ( ’ unit ramp signal ’ ) ;
 xlabel ( ’−−−>n ’ ) ; 13 ylabel ( ’−−−>x ( n ) ’ ) ;
```



(c) unit impulse function:

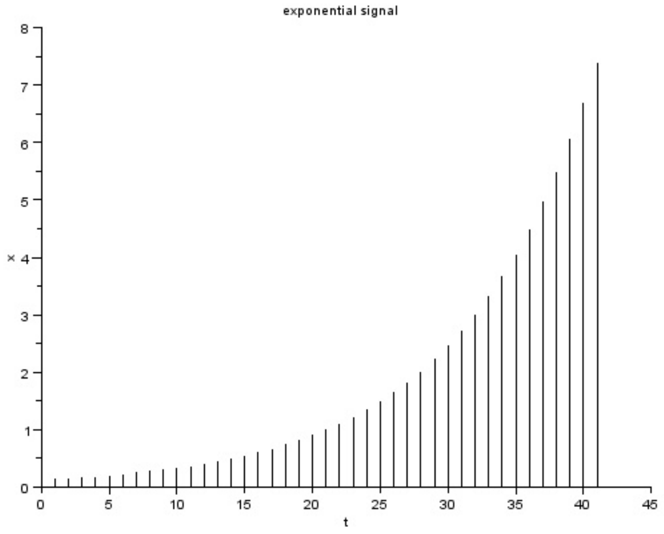
```
clc ;
clf ;
clear all;
 // unit impulse
 L =5;
 n = - L : L;
 x =[ zeros (1 , L) ,ones (1 ,1) ,zeros (1 , L) ];
 a = gca () ;
 a . y_location =” middle ”
 plot2d3 (n ,x ) ;
 title ( ’ unit impulse ’ ) ;
```



(d) Exponential sequence:

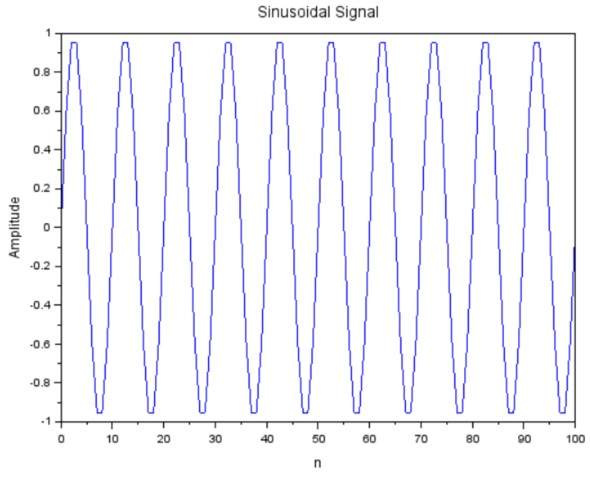
```
// u n i t e x p o n e n t i a l
clc ;
clear all;
a =1;
x= exp (a*t);
```

```
plot2d3 (x);
title ( 'e x p o n e n t i a l s i g n a l ' );
xlabel ( ' t ' );
ylabel ( ' x ' );
```



(e) sinusoidal

```
n = 0:100;  // Range of time indices
f = 0.1;   // Frequency of the sinusoidal signal
A = 1;     // Amplitude of the sinusoidal signal
phi = 0;   // Phase of the sinusoidal signal
x = A * sin(2*%pi*f*n + phi);  // Sinusoidal signal
plot(n, x)
xlabel('n')
ylabel('Amplitude')
title('Sinusoidal Signal')
grid on
```



RESULT:

Thus the given discrete time signal has been GENERATED.

EXP. NO. 3

PERIODIC AND APERIODIC SIGNALS

AIM: Determine whether the given signal is periodic or not.

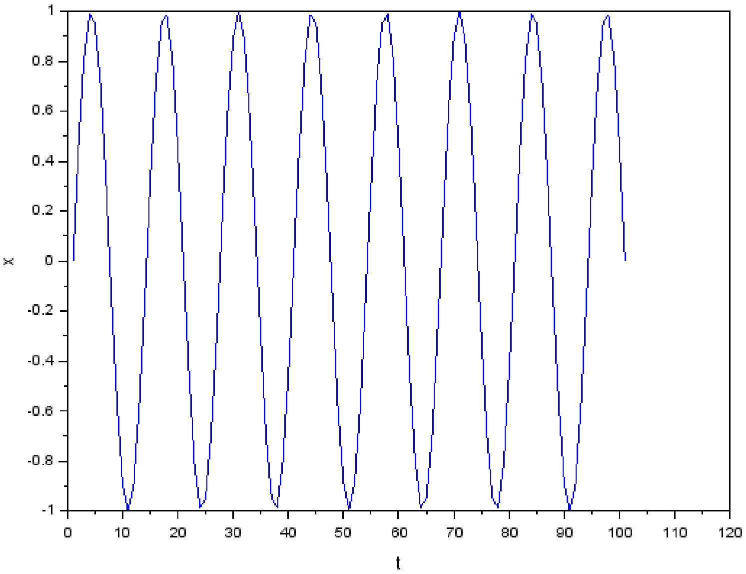
APPARATUS REQUIRED: PC and SCILAB Software.

ALGORITHM:

a) continuous time :
   x(t) = x(t+T) for all t
   T→ fundamental period
b) Discrete time
   x(n)=(n+N) for all n
   N→ fundamental period

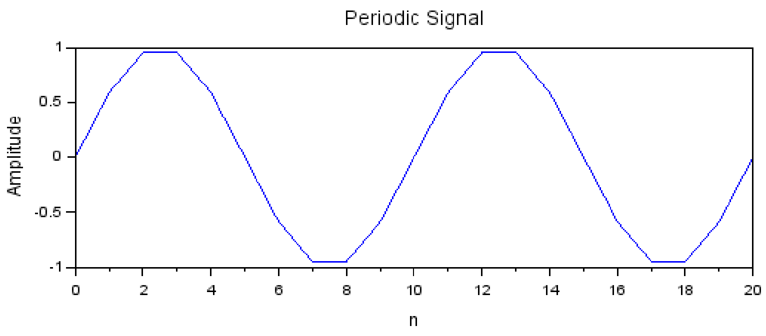
PROGRAM:

a)Determine whether the given signal is periodic or not
```
clc;
t=0:1/100:1
x=sin(15*%pi*t);
plot(x);
disp('plotting the signal and showing that it is periodic with period=2pi/15pi');
xlabel("t");
ylabel("x");
```



b)
```
// Periodic Signal
n_periodic = 0:20;  // Range of time indices for the periodic signal
f_periodic = 0.1;   // Frequency of the periodic signal
A_periodic = 1;     // Amplitude of the periodic signal
phi_periodic = 0;   // Phase of the periodic signal
```

```
x_periodic = A_periodic * sin(2*%pi*f_periodic*n_periodic + phi_periodic);  // Periodic
signal
subplot(2, 1, 1)
plot(n_periodic, x_periodic)
xlabel('n')
ylabel('Amplitude')
title('Periodic Signal')
grid on
```



RESULT: Thus the given signal is verified by periodic or aperiodic signal.

EXP. NO. 4

ADDITION AND MULTIPLICATION OF TWO SEQUENCES

AIM:To find Addition and Multiplication of two sequences.

APPARATUS REQUIRED: PC and SCILAB software.

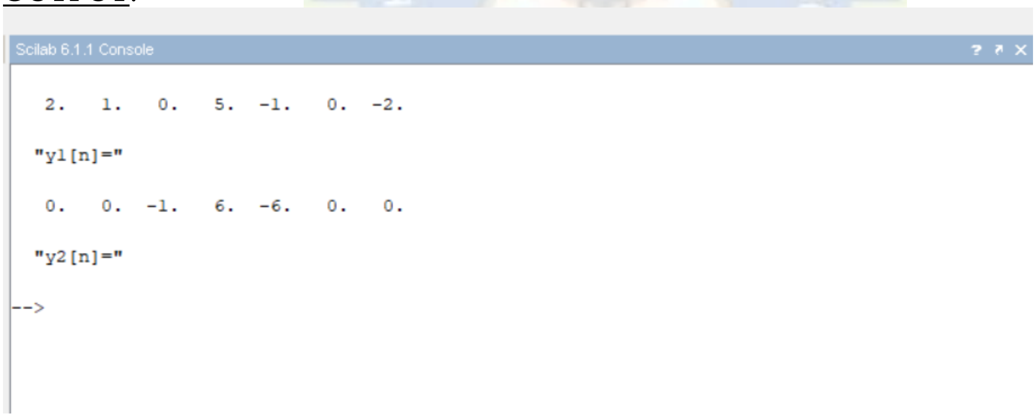
ALGORITHM:

X1 and x2 are the given sequences
y1=x1*x2 (MULTIPLICATION)
Y2= X1+X2 (ADDITION)

PROGRAM:

```
clear;
clc;
n=-1:5;
x1=[0,0,1,2,-3,0,-2];
x2=[2,1,-1,3,2,0,0];
y1=x1+x2;
disp(y1,'y1[n]=');
y2=x1.*x2;
disp(y2,'y2[n]=');
```

OUTPUT:



RESULT:
Thus the given signal is verified by multiplication and addition of two sequences.

EXP. NO. 5

DISCRETE-TIME SYSTEM

AIM: To Check if y[n] = x[n-1] is memoryless, causal, linear, time variant.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

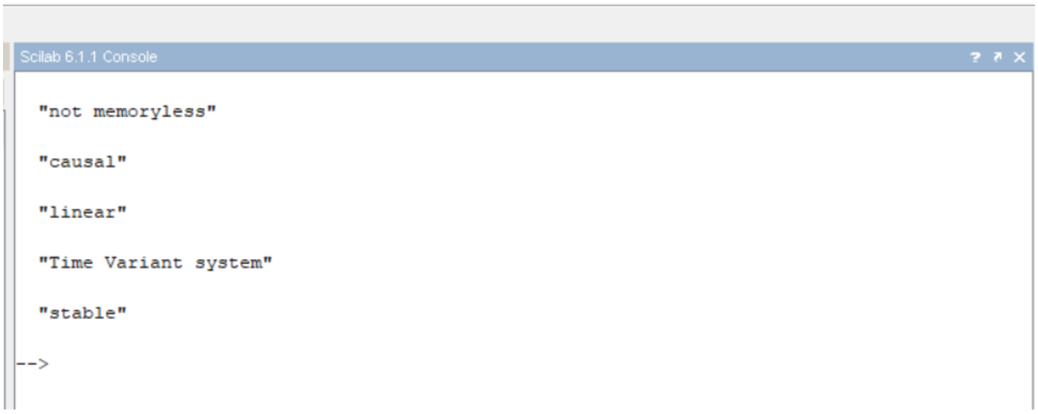
*Linear and nonlinear
need to satisfy superposition principle [a*x_{1}(t) + b*x_{2}(t)] = aT[x(t)] + bT(x_{2}(t)]
[axi(t) + bx_2 (t)] @ If the Condition satisfied it is linear..
*Time invariant and Time Variant -
Input and Output do not chang, therefore it is time invariant.
*Causal and Non causal
Causal present input and past output
Non-causal present input and future output.
*Static and Dynamic.
Static [Memory less] = Present input
Dynamic [Memory] =Past or future input
*stable - produces bounded output for bounded input.

PROGRAM:

```
clear;
clc;
s = 2;//shift
T = 20; //length of signal
x(1)=1;
for n = 2:T
  x(n) = n;
  y(n) = x(n-1);
end
if y(2)==x(2) then
   disp("memoryless")
else
   disp("not memoryless")
end
//causal if it doesn't depend on future
if y(2)==x(2) | y(2)==x(1) then
   disp('causal')
else
        disp('non casual')
end
x1=x;
y1=y;
x2(1)=2;
```

```
for n = 2:T
  x2(n) = 2;
  y2(n) = x2(n-1);
end
z=y1+y2;
for n = 2:T
  y3(n) = (x2(n-1)+x1(n-1));
end
if z==y3 then
    disp('linear')
else
    disp("nonlinear")
end
Ip = x(T-s);
Op = y(T-s);
if(Ip == Op) n
  disp(' Time In-variant system');
else
  disp('Time Variant system');
end
Max =20;
dd=1;
for n=2:T
            if y(n)>Max then
            dd=0
            end
end
if dd==0
  disp('unstable')
 else
  disp('stable');
end
```

<u>output:</u>



<u>RESULT</u>: The discrete time system is not memoryless, causal, linear, time variant system and stable.

EXP. NO. 6

CAUSAL/NON-CAUSAL SYSTEM

AIM: To verify whether the given system is causal or non-causal.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

The system is causal or non-causal based on the time-invariance of its impulse response.

A system is causal if its impulse response h(t) is zero for t < 0, meaning that its output at any time t depends only on the input at that time and previous times

.A system is non-causal if its impulse response h(t) is non-zero for t < 0, meaning that its output at any time t depends on future inputs.

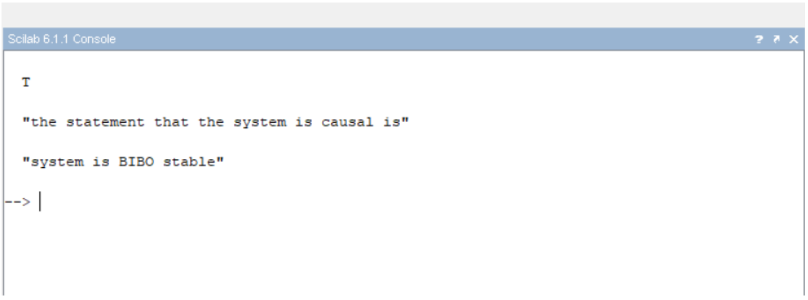
Assume n= -5 to 5 and α=0.5.

PROGRAM:

```
clear;
clc;
n=-5:5;
alpha=.6;
for i=1:length(n)
        if(n(i)>=0)
        h(i)=alpha^n(i);
else
        h(i)=0;
end
end
        causal=%t;
        for i=1:length(n)
                if n(i)<0 & h(i)~=0 then
                causal=%f;
                end
        end
        disp(causal,"the statement that the system is causal is");
        n=0:100000;
        for i=1:length(n)
                if(n(i)>=0)
                h(i)=alpha^n(i);
                else
                h(i)=0;
                end
         end
         bibo=sum(h);
         if (bibo<%inf) then
                disp("system is BIBO stable");
         else
```

```
                disp("system is not stable");
        end
   xlabel('y')
   ylabel('x')
```

output:



RESULT: The system has been verified as causal.

EXP. NO. 7

AUTOCORRELATION

AIM: To Compute the autocorrelation function Rxx(τ) as a function of time lag τ.

APPARATUS REQUIRED: PC, SCILAB software.

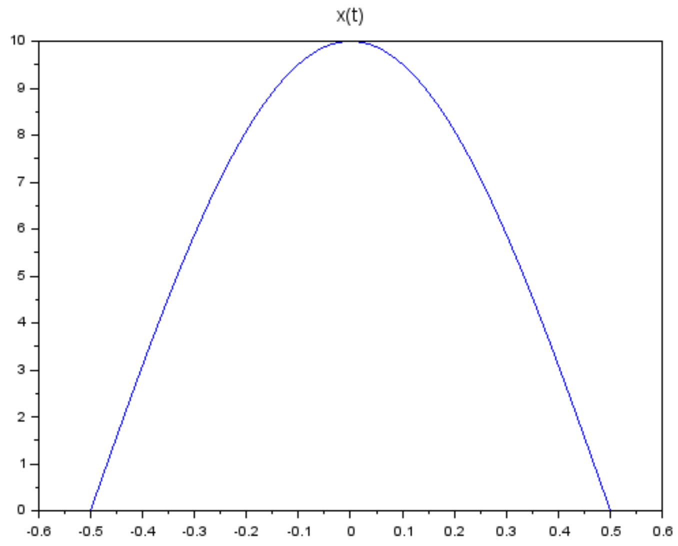
ALGORITHM:

Define the input signal x(t) that you want to autocorrelate. The signal should be of finite duration and have a known sampling rate.
Compute the length of the signal N, which is the number of samples in the signal.
Define the lag variable τ, which represents the time delay between the signal and its time-delayed version. The range of τ should cover the expected time delay of the signal.
Compute the autocorrelation function Rxx(τ) for each time lag τ using the following formula:
$Rxx(\tau) = 1/N * \sum[x(n) * x(n-\tau)]$, where n ranges from 0 to N-1.
In other words, for each time lag τ, multiply the samples of x(t) and its time-delayed version x(t-τ) and sum them over all the samples in the signal. Then divide by the total number of samples N.
Normalise the autocorrelation function if necessary to obtain a normalised autocorrelation coefficient, which varies between -1 and 1. This can be done by subtracting the mean and dividing by the standard deviation of the signal.

PROGRAM:

```
clc;
clear;
T=1;
n=1;
for t=-T/2:0.01:T/2;
   x(n)=10*cos(%pi*t/T);
         n=n+1;
end
t=-T/2:0.01:T/2;
plot(t,x);
title('x(t)');
disp('Rxx(0)=Energy of signal');
Rxx=integrate('50*(1+cos(2*%pi*t/T))','t',-T/2,T/2);
disp(Rxx,'Rxx(0)=');
Xlabel('y')
ylabel('x')
```

output:



<u>RESULT</u>: The function of autocorrelation has been determined by the formula.

EXP.NO. 8

CROSS CORRELATION

AIM:Plot the cross-correlation function Rxy(τ) as a function of time lag τ.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Define the two input signals x(t) and y(t) that you want to cross-correlate. The signals should be of finite duration and have the same sampling rate.

Compute the length of the signals N, which is the number of samples in each signal.

The range of τ should cover the expected time delay between the signals.

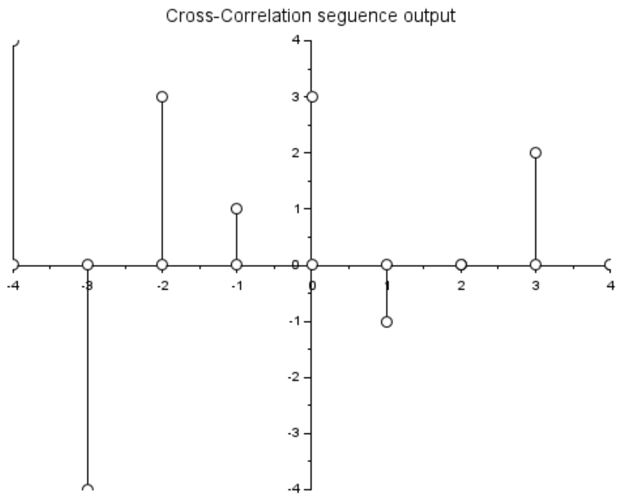
Compute the cross-correlation function Rxy(τ) for each time lag τ using the following formula:
$Rxy(\tau) = 1/N * \sum[x(n) * y(n+\tau)]$, where n ranges from 0 to N-1.

For each time lag τ, multiply the samples of x(t) and y(t+τ) and sum them over all the samples in the signal. Then divide by the total number of samples N.
Plot the cross-correlation function Rxy(τ) as a function of time lag τ.

PROGRAM:

```
clear;
clc;
x=[2,-1,1,0,2];
y=[0,1,0,-1,2];
//computation of cross correlation sequence;
n1 = max(size(y))-1;
n2 = max(size(x))-1;
r = xcorr(x,y,n1);
n=-4:4;
a=gca();
a.x_location="origin";
a.y_location="origin";
plot2d3(n,r,-9);
title('Cross-Correlation seguence output');
Xlabel('y')
ylabel('x')
```

output:



<u>RESULT</u>: The cross correlation function has been determined.

EXP.NO. 9

CONVOLUTION OF CONTINUOUS TIME SIGNALS

<u>AIM</u>: To find the convolution of two continuous time signals
<u>APPARATUS REQUIRED</u>: PC and SCILAB software
<u>ALGORITHM</u>:

⇒ x(t) and h(t) are given two sequences

Where,

$y=\int (x)t\ h(t-z)\ \partial t$

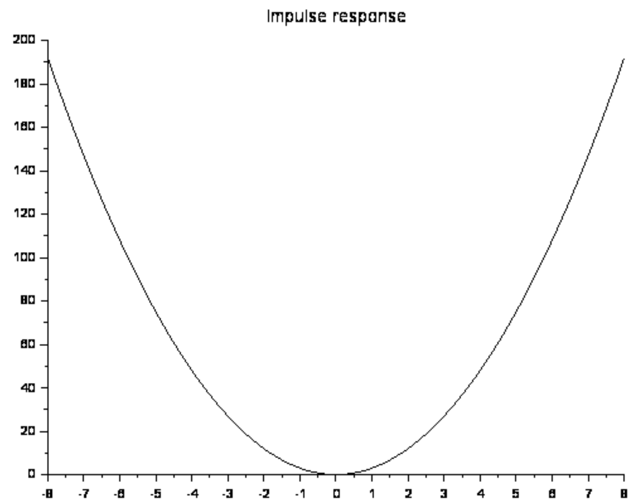
$y=x(t)\ .h(t)$

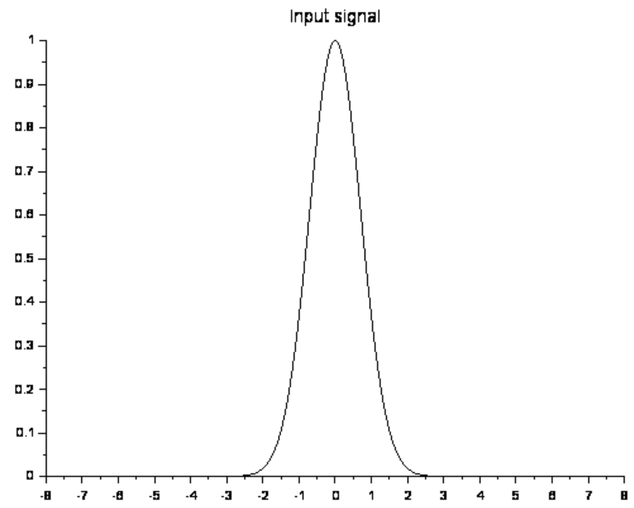
⇒ obtain h(t) by folding the h(t)
⇒do the time shifting of the signal
⇒then do the multiplication of both signals i.e, y=x(t) .h(t)

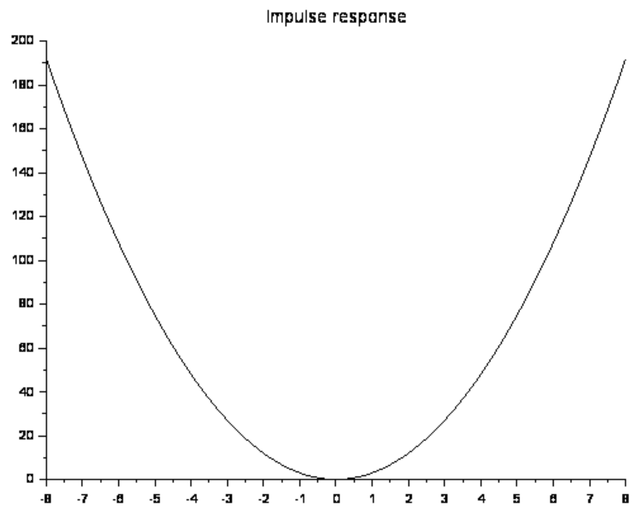
<u>PROGRAM</u>:

```
clc;
t=-8:1/100:8;
for i=1:length(t)
x(i)=exp(-t(i)^2);
h(i)=3*t(i)^2;
end
y=convol(x,h);
figure
plot2d(t,h);
title('Impulse response');
figure
plot2d(t,x);
title('Input signal');
figure
t2=-16:1/100:16
plot2d(t2,y);
title('Output signal');
```

<u>output</u>:







RESULT: The convolution of two given continuous time signals has been performed.

EXP. NO. 10

FOURIER SERIES OF HALF-WAVE RECTIFIER OUTPUT

AIM: To determine fourier series of Half -Wave rectifier output.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

1. Assume the period of the signal T=1sec.
2. Define the input voltage waveform v_i(t) as a sinusoidal signal with amplitude V_m and angular frequency ω.
3. Define the output voltage waveform v_o(t) as the rectified output of v_i(t), which is given by:

   v_o(t) = |v_i(t)| = v_i(t) for v_i(t) > 0, and v_o(t) = 0 for v_i(t) < 0.
3. Define the period of the waveform T = 2π/ω.
4. Compute the DC component of the Fourier series, a_0/2, as:

a_0/2 = V_m/π

5. For each odd harmonic number n (starting with n=1), compute the Fourier coefficient of the cosine term, a_n, as:

a_n = (2V_m/π) [(1/n) - cos(nπ)/nπ]

6. Compute the Fourier series of the half-wave rectifier output waveform, v_o(t), as:

v_o(t) = a_0/2 + ∑(n=1,3,5,...) ∞ a_n cos(nωt)

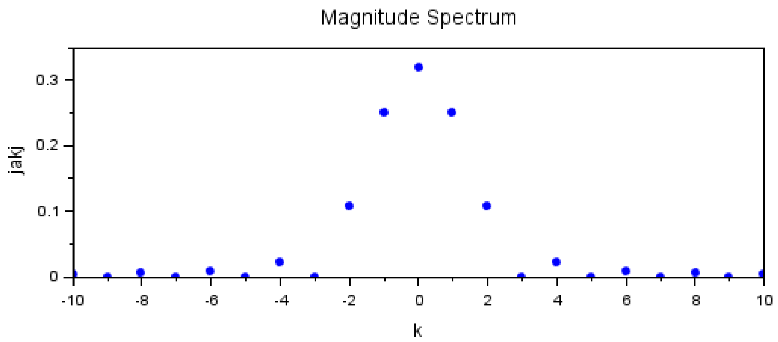
7. Plot the Fourier series as a function of time to visualise the waveform.

PROGRAM:

```
clc;
t=-0.5:0.01:1;
T=1;
for i=1:length(t)
if t(i)<T/2 then
x(i)=sin(2*%pi*t(i));
else
x(i)=0;
end
end
k=-10:10;
for i=1:length(k)
if k(i)==1 then
ak(i)=1/(4*%i);
elseif k(i)==-1
ak(i)=-1/(4*%i);
else
ak(i)=(cos(k(i)*%pi/2)*exp(-k(i)*%pi/2*-%i))/(%pi-(%pi*k(i)*k(i)));
end
end
disp("The fourier series coefficients are")
disp(ak)
disp("magnitude of Fourier series coefficient")
disp(abs(ak))
```

```
//Plotting frequency spectrum
subplot(2,1,1)
plot(k,abs(ak),'.');
xtitle("Magnitude Spectrum","k","jakj");
for i=1:length(k)
if k(i)==0|k(i)==3|k(i)==-3|k(i)==-5|k(i)==5 then
phase(i)=0;
elseif k(i)==-1 then
phase(i)=%pi/2;
elseif k(i)==1 then
phase(i)=-%pi/2;
elseif k(i)== -2|k(i)==-4
phase(i)=%pi;
elseif k(i)==2|k(i)==4
phase(i)=-%pi;
end
end
xlabel('y')
ylabel('x')
```

output:



RESULT: The fourier transform of a rectangular waveform has been generated

EXP.NO. 11

FOURIER TRANSFORM OF A RECTANGULAR WAVEFORM

<u>AIM</u>: To find Fourier transform of a rectangular waveform

<u>APPARATUS REQUIRED</u>: PC and SCILAB software.

<u>ALGORITHM</u>:

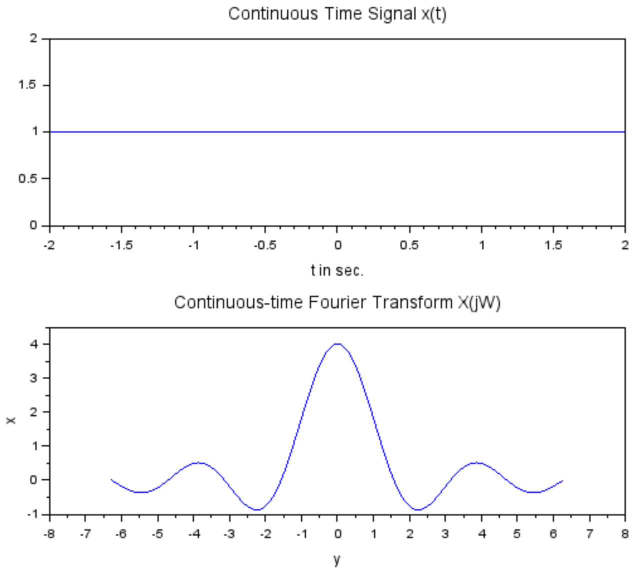
$f(t) = A$ for $0 <= t < T/2$
$f(t) = -A$ for $T/2 <= t < T$
where A is the amplitude of the waveform.
The Fourier transform of this waveform can be calculated using the formula:
$F(\omega) = \int[0,T]\ f(t)\ e\hat{}(-j\omega t)\ dt$
where ω is the angular frequency, and j is the imaginary unit.
By substituting the rectangular waveform into this equation and integrating, we get:
$F(\omega) = 2A\ [(\sin(\omega T/2))/(\omega T/2)]\ e\hat{}(-j\omega T/2)$
This is the Fourier transform of a rectangular waveform. It is a sinc function multiplied by a complex exponential term.
The sinc function is defined as:
$\mathrm{sinc}(x) = (\sin(x))/x$
where x is a real number. It has a central lobe that extends from $-\pi$ to π, and side lobes that decay as $1/x$ away from the central lobe.

<u>PROGRAM</u>:

```
clear;
clc;
A=1;
Dt=0.005;
T0=4;
t=-T0/2:Dt:T0/2;
for i=1:length(t)
xt(i)=A;
end
Wmax=2*%pi*1;
K=4;
k=0:(K/1000):K;
W=k*Wmax/K;
xt=xt';
XW=xt*exp(-sqrt(-1)*t'*W)*Dt;
XW_Mag=real(XW);
W=[-mtlb_fliplr(W),W(2:1001)];
XW_Mag=[mtlb_fliplr(XW_Mag),XW_Mag(2:1001)];
subplot(2,1,1);
plot(t,xt);
xlabel('t in sec.');
title('Continuous Time Signal x(t)')
```

```
subplot(2,1,2);
plot(W,XW_Mag );
xlabel('Frequency in Radians/Seconds');
title('Continuous-time Fourier Transform X(jW)')
xlabel("y")
ylabel("x")
```

Output:



RESULT: The given signal of fourier transform of periodic impulse train has been sketched

EXP. NO. 12

FOURIER TRANSFORM OF PERIODIC IMPULSE TRAIN

AIM: To sketch the fourier transform of periodic impulse train.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Define the impulse train as a periodic function of time with a period T, such that it can be written as:

1. $x(t) = \sum_n \delta(t - nT)$, where $\delta(t)$ is the delta function.Apply the Fourier series expansion of the impulse train, which is given by:
2. $x(t) = \sum_k c_k e^{(j 2\pi kt/T)}$, where $c_k = 1/T \int_0^T x(t) e^{(-j 2\pi kt/T)} dt$ is the k-th Fourier coefficient.

Substitute the impulse train expression into the Fourier series expansion and evaluate the Fourier coefficients.

3. $c_k = 1/T \int_0^T \delta(t) e^{(-j 2\pi kt/T)} dt = 1/T$.

Substitute the Fourier coefficients into the Fourier series expansion and simplify

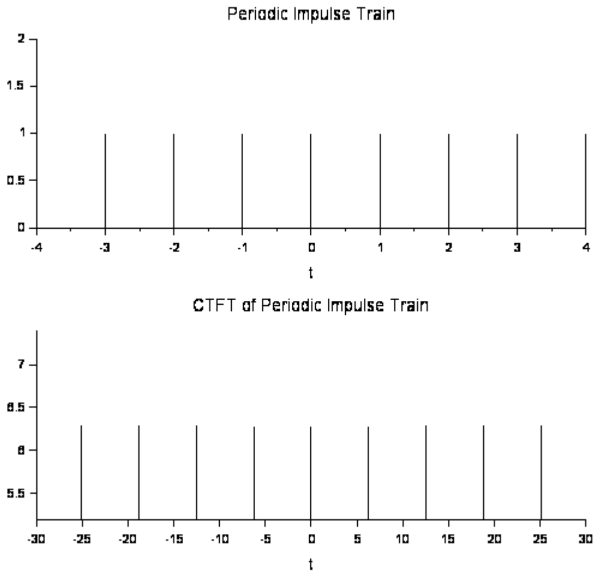
4. $X(f) = \sum_k c_k \delta(f - k/T) = (1/T) \sum_k \delta(f - k/T)$.

This result indicates that the Fourier transform of a periodic impulse train.

PROGRAM:

```
clear;
clc;
T=-4:4;
T1=1;
xt=ones(1,length(T));
ak=1/T1;
XW=2*%pi*ak*ones(1,length(T));
Wo=2*%pi/T1;
W=Wo*T;
figure
subplot(2,1,1)
plot2d3('gnn',T,xt);
xlabel('t');
title('Periodic Impulse Train')
subplot(2,1,2)
plot2d3('gnn',W,XW);
xlabel('t');
title('CTFT of Periodic Impulse Train')
xlabel('y')
ylabel('x')
```

output:



<u>RESULT</u>: The given signal has been analysed

EXP. NO. 13

ANALYSIS OF SYSTEM

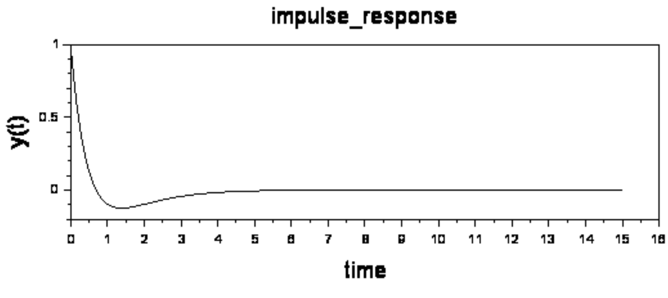
AIM: To determine the response of the system.

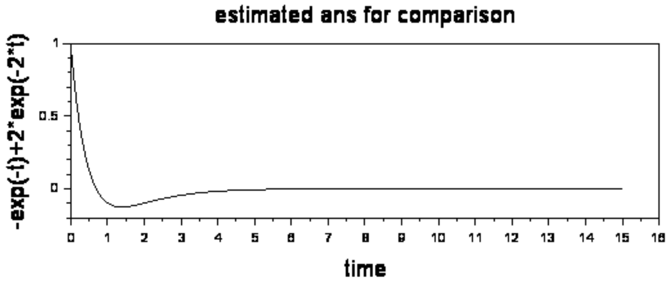
APPARATUS REQUIRED: PC and SCILAB software.

PROGRAM:

```
clc
clear
close;

function dy= f(t,y)
    dy=zeros(2,1);
    dy(1)=y(2);
    dy(2)=-3*y(2)-2*y(1);
endfunction

y0=[1;-3];//initial conditions for impulse response calculated from given system equation
t0=0;
t=linspace(0,15,500);
y=ode(y0,t0,t,f);   //solving the 2nd order ode system equation

figure(1)
subplot(2,1,1);plot(t,y(1,:));     //plotting the obtained result
title("impulse_response","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(t)","fontsize",4);

subplot(2,1,2);plot(t,-exp(-t)+2*exp(-2*t));//plotting the estimated answer for comparision
title("estimated ans for comparison","fontsize",4);
xlabel("time","fontsize",4);
ylabel("-exp(-t)+2*exp(-2*t)","fontsize",4);
```

output:





RESULT: The response of the system has been determined.

Exp. No. 14

Analysis of Continuous-time system

AIM: To determine and sketch the output of this system to the given input x(t).

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Obtain the Input Signal: Identify or generate the continuous-time signal that serves as the input to your system. The input signal could be a pre-recorded waveform or a synthesised signal, depending on your specific requirements.

Simulate the System: Apply the input signal to the system model to simulate its behaviour. Calculate the output of the system as a function of time using the system's mathematical equations or algorithms.

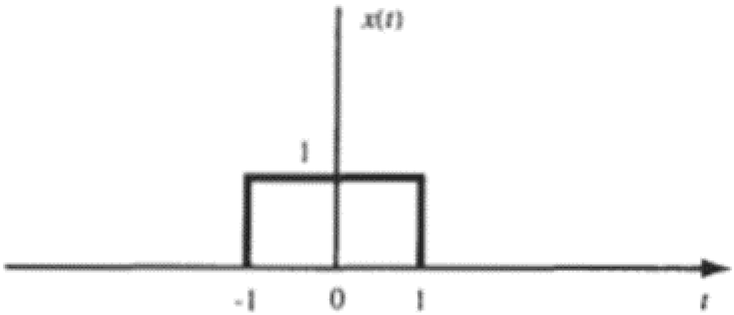
Perform Time-Domain Analysis: Analyse the output signal in the time domain. This can involve examining properties such as signal amplitude, duration, periodicity, and any time-varying behaviour.

Perform Frequency-Domain Analysis: Convert the time-domain signal into the frequency domain using techniques such as the Fourier Transform. Analyse the spectrum of the signal, which reveals information about its frequency content, harmonics, and energy distribution.

Consider a continuous-time LTI system whose step response is given by

$$s(t) = e^{-t}u(t)$$

shown in Fig

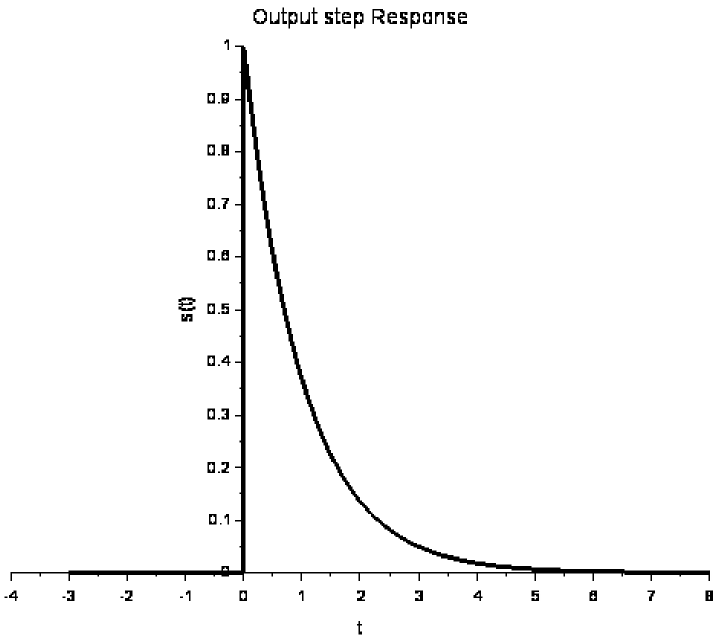


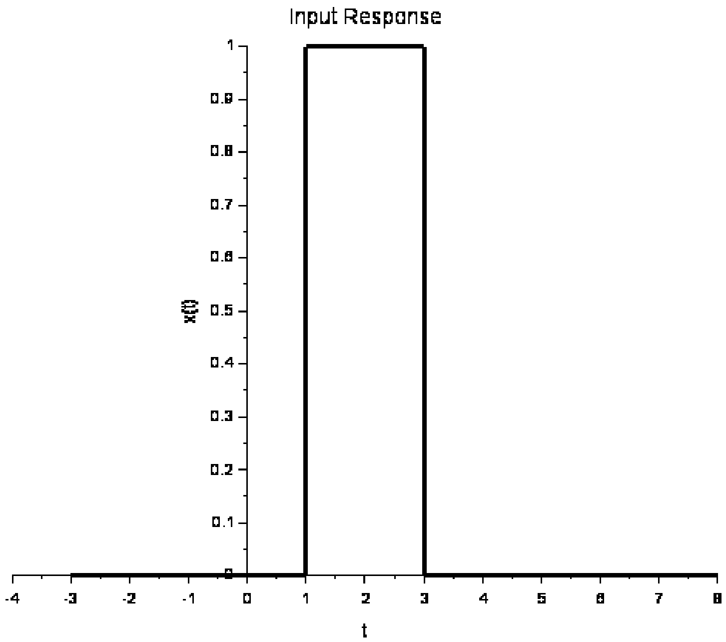
PROGRAM:

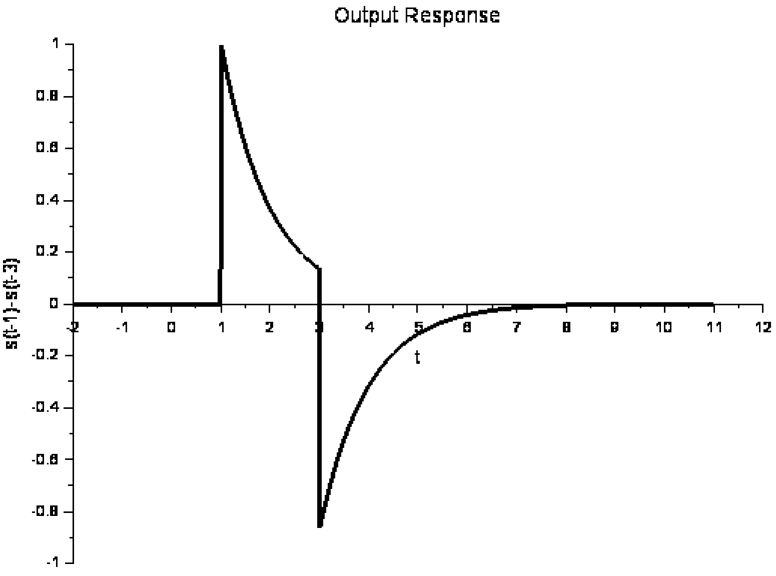
```
clear;
close;
clc;
t =-3:1/100:8;
s=[];
ss=[];
for i=1:length(t)
        if t(i)<1|t(i)>3 then
        x(i)=0;
        else
        x(i)=1;
```

```
            end
            if t(i)<0 then
            s(i)=0;
            else
            s(i)=exp(-t(i));
            end
end
figure
a=gca();
a.y_location="origin";
a.x_location="origin";
plot2d(t,s)
xtitle('Output step Response','t','s(t)');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
figure
a=gca();
a.x_location="origin";
a.y_location="origin";
plot2d(t,x)
xtitle('Input Response','t','x(t)');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
t1=t+1;
t2=t+3;
s=s';
tt=min(min(t1,t2)):1/100:max(max(t1,t2));
ee=zeros(tt);
x=find(tt==-2);
y=find(tt==0);
z=find(tt==9);
for i=1:1:length(tt)
            if i<y then
        ee(i)=s(i);
            elseif i<z
        ee(i)=s(i)-s(i-y+1);
            else
        ee(i)=-s(i-y+1);
            end
end
figure
a=gca();
a.y_location="left";
a.x_location="origin";
plot2d(tt,ee)
xtitle('Output Response','t','s(t-1)-s(t-3)');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
```

Output:


Output step Response
s(t)
t
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8


Input Response
x(t)
t
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-3
-2
-1
0
1
2
3
4
5
6
7
8



RESULT: The *given Continuous-time system has been analysed*

EXP NO. 15

IMPULSE RESPONSE OF THE SYSTEM

AIM: To determine Impulse response of the system.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

It can be in the form of a transfer function, a set of differential equations, or a difference equation.

If the system has initial conditions, ensure that they are properly set to zero.

Create an input signal, known as the impulse signal. The impulse signal is a discrete signal that has a single sample with a value of 1 at time index zero and all other samples set to zero.

Apply the impulse signal as the input to the system.

Simulate or solve the system equations using the impulse input. This can involve numerical methods, analytical solutions, or simulation tools depending on the complexity of the system.

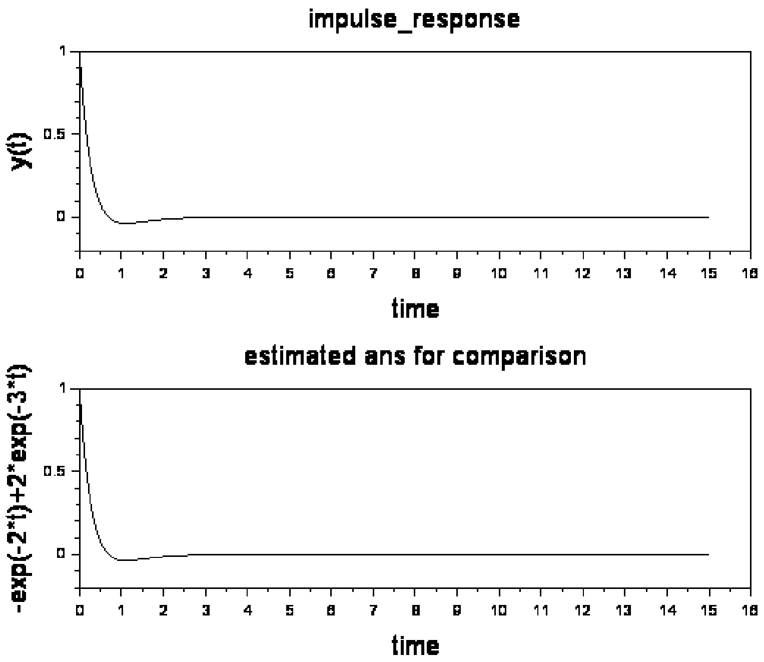
Record or calculate the output of the system as it responds to the impulse input. This represents the impulse response of the system.

PROGRAM:

```
clc
clear
close;

function dy= f(t,y)
   dy=zeros(2,1);
   dy(1)=y(2);
   dy(2)=-5*y(2)-6*y(1);
endfunction

y0=[1;-4];//initial conditions for impulse response
t0=0;
t=linspace(0,15,500);
y=ode(y0,t0,t,f);   //solving the 2nd order ode system equation

figure(1)
subplot(2,1,1);plot(t,y(1,:));     //plotting the obtained result
title("impulse_response","fontsize",4);
xlabel("time","fontsize",4);
ylabel("y(t)","fontsize",4);

subplot(2,1,2);plot(t,-exp(-2*t)+2*exp(-3*t));//plotting the estimated answer for comparision
title("estimated ans for comparison","fontsize",4);
```

xlabel("time","fontsize",4);
ylabel("-exp(-2*t)+2*exp(-3*t)","fontsize",4);

<u>OUTPUT</u>:



<u>RESULT</u>: The impulse response of the system is determined.

EXP. NO. 16

DISCRETE TIME FOURIER TRANSFORM

AIM:To find Discrete time Fourier Transform of the given signal

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Write the expression for the DTFT of x[n], which is defined as:

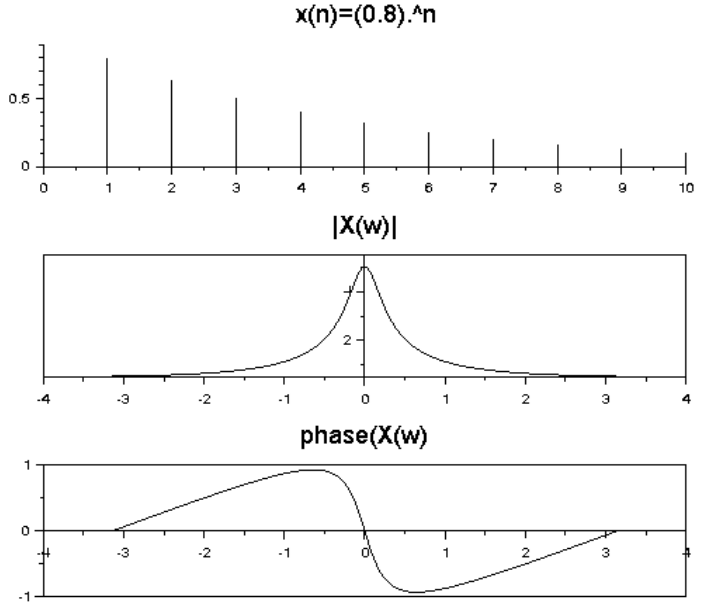
1. X(e^(jω)) = ∑n x[n] e^(-jωn).
2. Identify the range of frequencies for which the DTFT is defined. Therefore, it is sufficient to evaluate the DTFT in the range $-\pi \le \omega < \pi$.
3. Evaluating the DTFT usingThis involves computing the complex exponential e^(-jωn) for each value of n and multiplying it by the corresponding sample x[n], and then summing these products over all values of n.
4. Simplify the expression for X(e^(jω)) using algebraic manipulations and the properties of complex exponentials and sums.
5. The DTFT is periodic with a period of 2π, and its magnitude and phase responses can reveal important information about the signal x[n].
6. Optionally, plot the magnitude and phase responses of the DTFT as a function of frequency ω. This can provide a visualisation of the spectral content of the signal x[n].
7. Specific properties of the signal x[n], such as its symmetry, periodicity, or causality.

Gn: $x=(0.8)^n$

PROGRAM:

```
clc
clear
close;
N=1024; //samples number
n=0:N-1;
x=(0.8).^n;
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
X_angle=fftshift(phasemag(X)*%pi/180);

figure(1)
subplot(3,1,1);plot2d3(n,x,[2]);
title("x(n)=(0.8).^n","fontsize",4);
set(gca(),"zoom_box",[0 0 10 0.9]);
subplot(3,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);
subplot(3,1,3);plot(omega,X_angle,'r');
set(gca(),"x_location","middle");
title("phase(X(w)","fontsize",4);
```

OUTPUT:



RESULT: The Discrete time Fourier Transform of the given signal is been analysed.

EXP. NO. 17

DISCRETE TIME FOURIER TRANSFORM OF RECTANGULAR PULSE

AIM: To find Discrete time Fourier Transform of rectangular pulse.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

The x[n], where n represents the discrete time index.

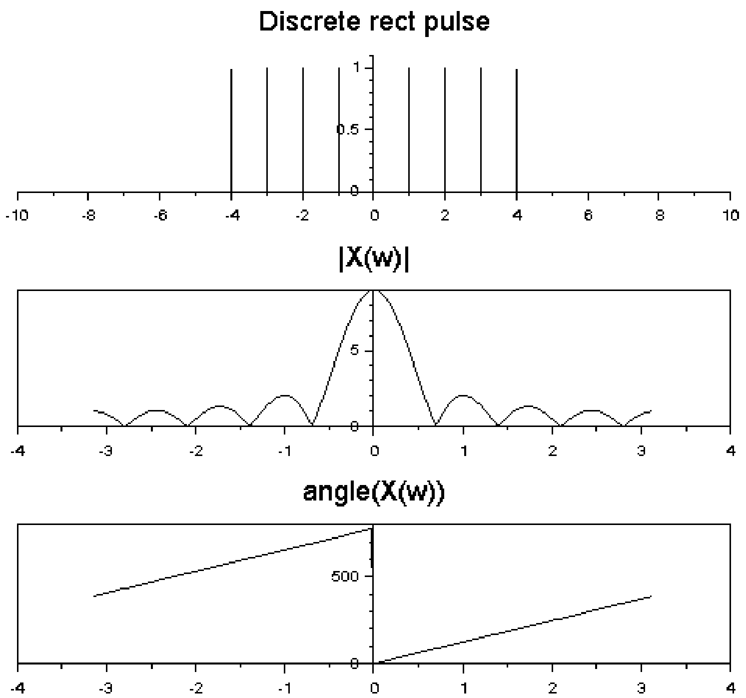
Determine the length of the rectangular pulse signal. Let's denote it as N.

Define the frequency variable for the DTFT. Let's call it ω, where ω ranges from -π to π.
Compute the DTFT coefficient X(ω) for each frequency value ω using the following formula:
X(ω) = Σ(x[n] * e^(-j * ω * n)) for n = 0 to N-1
In this formula, j represents the imaginary unit (√(-1)), and e^(-j * ω * n) represents the complex exponential term.

PROGRAM:

```
clc
clear
close;
N=256; //samples number
n=-(N/2):N/2-1;
x=1.*((n>=-4)&(n<=4));
omega=[-N/2:(N/2)-1]*2*%pi/N;
X=fft(x);
X_mode=fftshift(abs(X));
X=round(X*10000)/10000;
X_angle=fftshift(phasemag(X)*%pi/180);
figure(1)
subplot(3,1,1);plot2d3(n,x,[2]);
title("Discrete rect pulse","fontsize",4);
set(gca(),"zoom_box",[-10 0 10 1.1],"y_location","middle");
subplot(3,1,2);plot(omega,X_mode,'r');
set(gca(),"y_location","middle");
title("|X(w)|","fontsize",4);
subplot(3,1,3);plot(omega,X_angle,'r');
set(gca(),"y_location","middle");
title("angle(X(w))","fontsize",4)
```

output:



RESULT: The Discrete time Fourier Transform of rectangular pulse has been determined.

EXP.NO. 18

POLE ZERO PLOT

AIM: To Analyse the frequency response of the system based on the location of its zeros and poles in the complex plane.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Write the transfer function H(s) or H(z) in factored form, which separates the numerator and denominator into their individual poles and zeros.

For example, H(s) can be written as H(s) = K(s - z1)(s - z2)...(s - zm)/(s - p1)(s - p2)...(s - pn), where K is a constant and z1, z2, ..., zm are the zeros of H(s) and p1, p2, ..., pn are its poles. Similarly, H(z) can be written as H(z) = K(z - z1)(z - z2)...(z - zm)/(z - p1)(z - p2)...(z - pn).

Plot the zeros and poles of H(s) or H(z) as points in the complex plane. The zeros are plotted as circles or crosses, and the poles are plotted as X's or squares.

Determine the stability of the system based on the location of its poles in the complex plane. If all of the poles lie in the left half-plane (Re{s} < 0) or inside the unit circle for H(z), the system is stable and will converge to a steady state response.

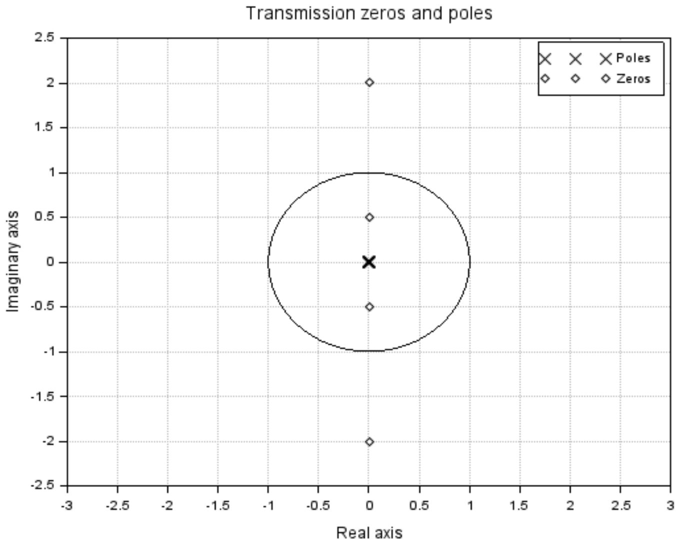
If any poles lie in the right half-plane (Re{s} > 0) or outside the unit circle for H(z), the system is unstable and will exhibit unbounded or oscillatory behaviour.

Analyse the frequency response of the system based on the location of its zeros and poles in the complex plane.

PROGRAM:

```
clear;
z=%z;
az=z^4+4.25*z^2+1;
bz=z^4;
poles=roots(bz)
zeroes=roots(az)
h=az/bz
plzr(h)
```

Output:



RESULT: The system has been analysed by pole zero plot.

EXP.NO. 19

Z-TRANSFORM OF THE GIVEN SIGNAL

AIM: To determine Z transform of the given signal.

APPARATUS REQUIRED: pc, scilab software.

ALGORITHM:

Write the expression for the Z-transform of x[n], which is defined as:

1. X(z) = ∑n x[n] z^(-n).
2. Identify the region of convergence of the Z-transform. The ROC can be expressed in terms of the radius of convergence r and the angle of convergence θ, as {z : |z| > r, or |z| = r and θ_1 < arg(z) < θ_2}.

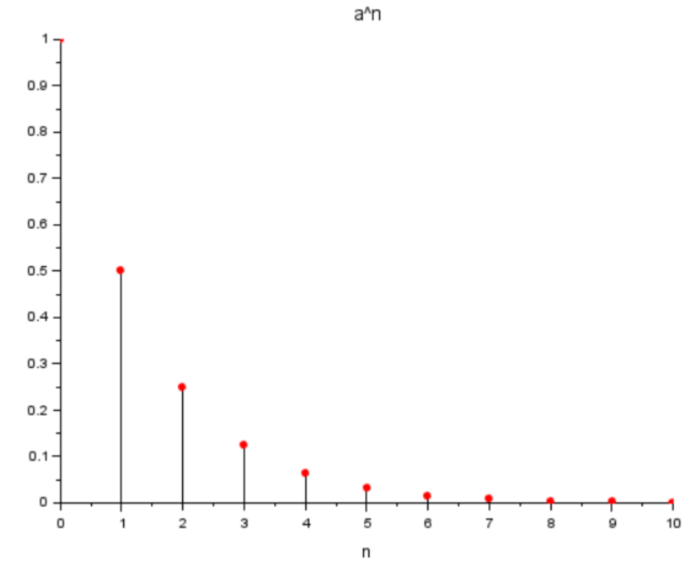
3.Analyse the ROC of X(z) by examining the values of z for which the series converges or diverges

4.Express the final result for X(z) in terms of a pole-zero plot or a partial fraction expansion, if applicable. A pole-zero plot shows the location of the poles and zeros of X(z) in the complex plane, while a partial fraction expansion expresses X(z) as a sum of simple fractions.

PROGRAM:

```
clc
Syms n z;
a = 0.5;
x =(a)^n;
n1=0:10;
plot2d3(n1,a^n1); xtitle('a^n','n');
plot(n1,a^n1,'r.')
X = symsum(x*(z^(-n)),n,0,%inf)
disp(X,"ans=")
(II)
clc
Syms n z;
Wo =%pi/4;
a = (0.33)^n;
x1=%e^(sqrt(-1)*Wo*n);
X1=symsum(a*x1*(z^(-n)),n,0,%inf)
x2=%e^(-sqrt(-1)*Wo*n)
X2=symsum(a*x2*(z^(-n)),n,0,%inf)
X =(1/(2*sqrt(-1)))*(X1+X2)
disp(X,"ans1=")
```

OUTPUT:

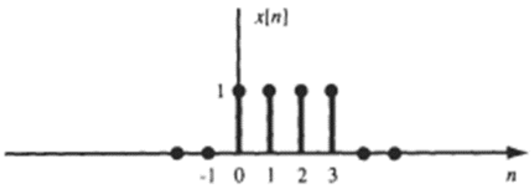


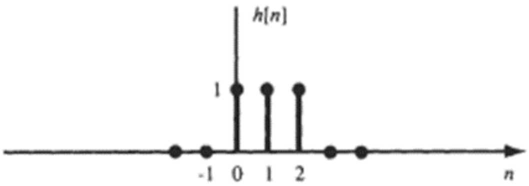
RESULT: The Z transform of the given signal has been determined.

EXP. NO. 20

CONVOLUTION OF DISCRETE TIME SIGNALS

AIM: To Evaluate y[n] = x[n] * h[n], for the given signals.





APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Given two discrete-time signals, let's call them x[n] and h[n], where n represents the discrete time index.

Determine the lengths of the signals: Determine the number of samples or the length of both signals, denoted as Nx and Nh, respectively.

Define the output signal: Create an output signal y[n] that will store the result of the convolution. The length of the output signal will be Nx + Nh - 1.

Initialise the output signal: Set all the elements of y[n] to zero.
Perform the convolution: Loop through each sample index of the output signal y[n] from 0 to Nx + Nh - 2.

Inside the loop, calculate the current value of y[n] as follows:
y[n] = 0
for k from 0 to Nh - 1:
if (n - k) >= 0 and (n - k) < Nx:

- y[n] += x[n - k] * h[k]

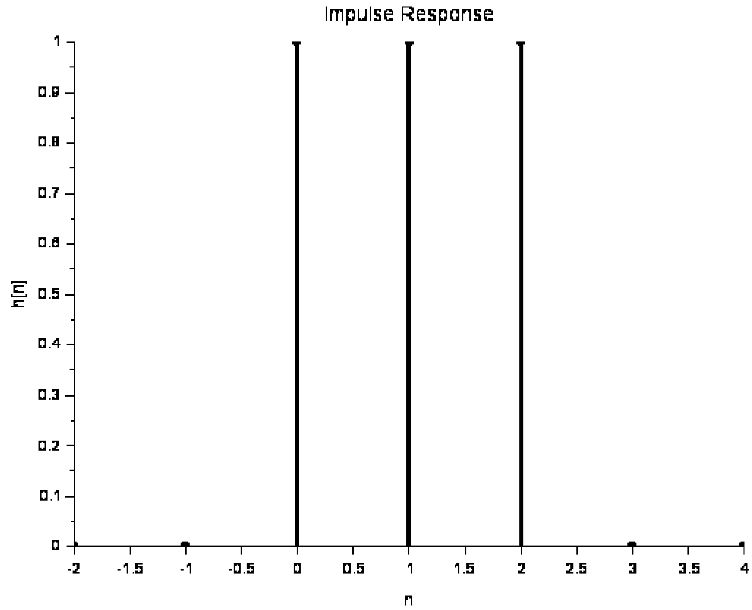
The above calculations involve multiplying the values of x[n] and h[n] at specific time indices and accumulating the result in y[n].

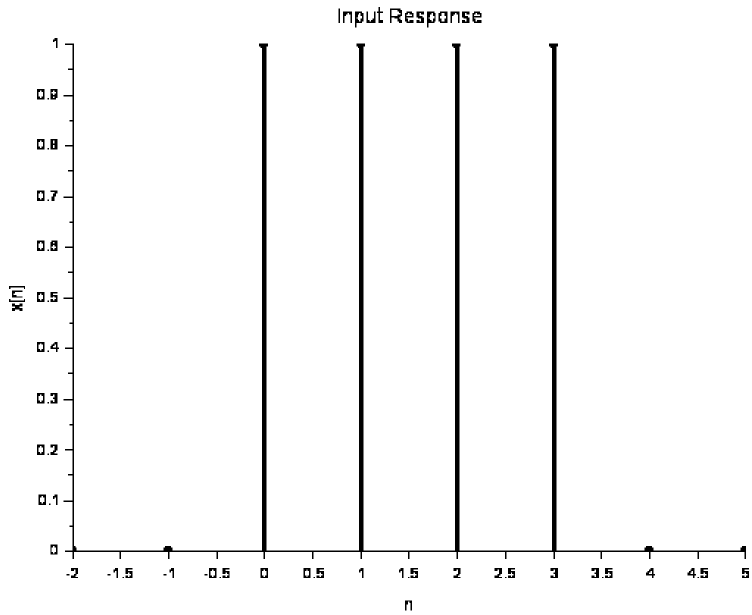
Note: Care must be taken to handle cases where the indices are out of range to avoid accessing invalid memory locations.
The resulting output signal y[n] represents the convolution of the input signals x[n] and h[n].

PROGRAM:

```
clear;
clc;
x=[0 0 1 1 1 1 0 0];
h=[0 0 1 1 1 0 0];
nx=-2:length(x)-3;
nh=-2:length(h)-3;
y=convol(x,h);
ny=min(nx)+min(nh):max(nx)+max(nh);
figure
a=gca();
a.x_location="origin";
plot2d3(nh,h)
plot(nh,h,'r.')
xtitle('Impulse Response','n','h[n]');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
a.y_location="left";
figure
a=gca();
plot2d3(nx,x)
plot(nx,x,'r.')
a.y_location="left";
a.x_location="origin";
xtitle('Input Response','n','x[n]');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
figure
a=gca();
plot2d3(ny,y)
plot(ny,y,'r.')
a.x_location="origin";
a.y_location="left";
xtitle('Output Response','n','y[n]');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
```

output:





<u>RESULT</u>: The Convolution of discrete time signals has been determined.

EXP. NO. 21

DISCRETE FOURIER TRANSFORM

AIM: To find Discrete Fourier Transform of the given signal.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

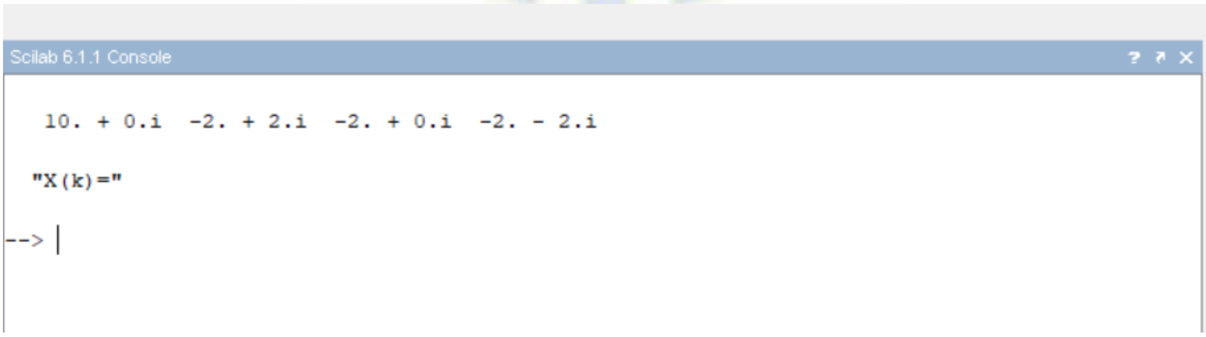
> Given a signal of length N, let's call it signal x[n].
>
> Create an output array or signal of length N, initialised with zeros. Let's call it signal X[k], where k represents the frequency index.
>
> For each frequency index k from 0 to N-1, calculate the DFT coefficient X[k] using the following formula:
> X[k] = sum(x[n] * e^(-j * 2 * pi * k * n / N)) for n = 0 to N-1
>
> After calculating the DFT coefficients for all frequency indices, the resulting array signal X[k] will represent the frequency domain representation of the input signal x[n].

Gn: x[n]=[1,2,3,4]

PROGRAM:

```
clc;
x=[1,2,3,4];
X=fft(x,-1);
disp(X,'X(k)=');
xlabel('y')
ylabel('x')
```

OUTPUT:



RESULT: The Discrete Fourier Transform has been analysed.

EXP. NO. 22

CIRCULAR CONVOLUTION

AIM: To find the circular convolution of the given sequences.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

1. Given two input signals of length N, let's call them signal X and signal Y.
2. Create an output array or signal of length N, initialised with zeros. Let's call it signal Z.
3. Perform the circular shift operation on signal Y. Starting from the first element, move each element one position to the right. The rightmost element wraps around to the first position. Repeat this process N times to generate N circularly shifted versions of signal Y.
4. Multiply each element of the circCIRCULAR CONVOLUTIONularly shifted signal Y with the corresponding element of signal X.
5. Sum up the products obtained in step 4 for each circularly shifted version of signal Y. The resulting sum represents one element of signal Z.
6. Repeat steps 4 and 5 for all N circularly shifted versions of signal Y, updating the corresponding element in signal Z each time.
7. After completing steps 4-6 for all N circularly shifted versions, signal Z will contain the circular convolution result.

Gn: x1=[3,1,3,1] and x2=[1,2,3,4];

PROGRAM:

```
clc;
x1=[3,1,3,1];
x2=[1,2,3,4];
X1=fft(x1,-1);
X2=fft(x2,-1);
X3=X1.*X2;
x3=fft(X3,1);
disp(x3,'x3(n)=x1(n)(N)x2(n)');
xlabel('y')
ylabel('x')
```

Output:

RESULT: The circular convolution has been determined.

EXP. NO. 23

ANALYSIS OF DISCRETE-TIME SYSTEM

AIM: To analyse the discrete time system.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

Obtain the Input Signal: Identify or generate the discrete-time signal that serves as the input to your system.

Calculate the output of the system for each time step using the system's mathematical equations or algorithms.

Perform Time-Domain Analysis: Analyse the output signal in the time domain.

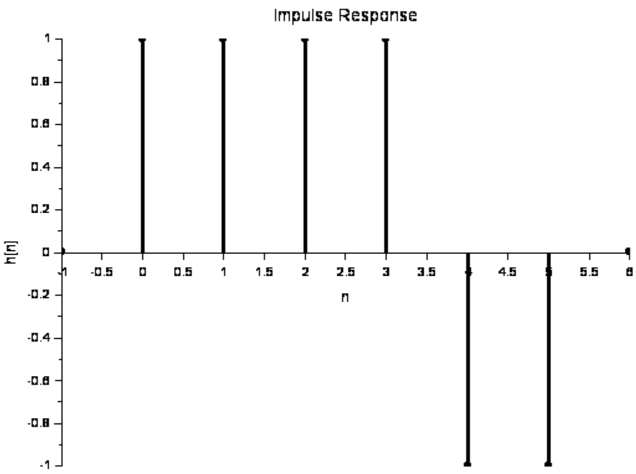
Perform Frequency-Domain Analysis: Convert the time-domain signal into the frequency domain using techniques such as the Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT).

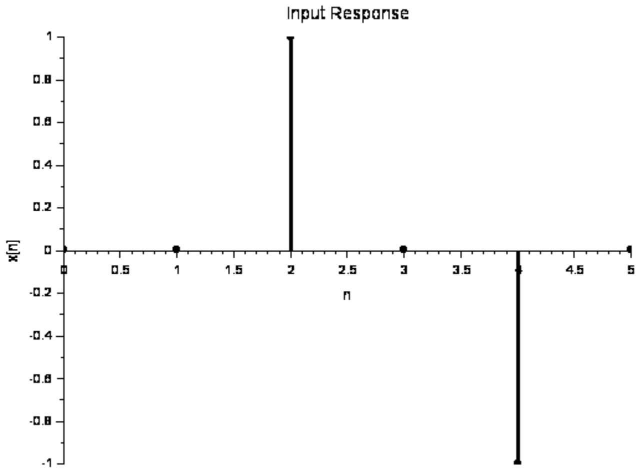
PROGRAM:

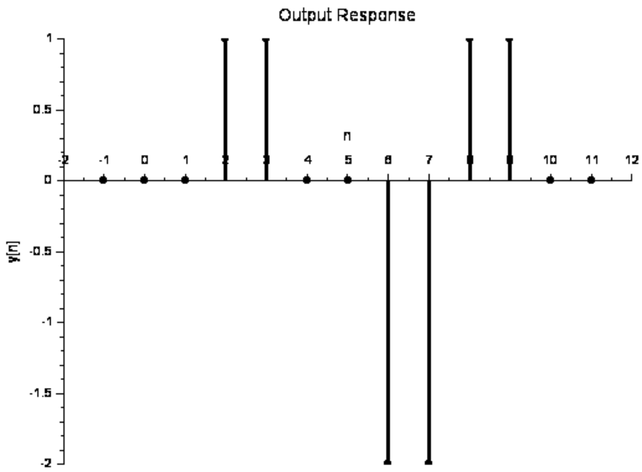
```
clear;
clc;
h=[0 1 1 1 1 -1 -1 0];
x=[0 0 1 0 -1 0];
nx=0:length(x)-1;
nh=-1:length(h)-2;
//y=convol(x,h);
ny=min(nx)+min(nh):max(nx)+max(nh);
//or x[n]=delta[n-2]-delta[n-4] therefore y[n]=h[n-2]-h[n-4]
n1=nh+2;
n2=nh+4;
ny=min(nx)+min(nh):max(nx)+max(nh);
j=1;
k=1;
h2=zeros(ny);
h4=h2;
a=find(ny==n1(1))
for j=1:length(nh)
   h2(a+j-1)=h(j)
        end
a=find(ny==n2(1))
for j=1:length(nh)
   h4(a+j-1)=h(j)
        end
y=h2-h4;
figure
a=gca();
a.x_location="origin";
plot2d3(nh,h)
plot(nh,h,'r.')
```

```
xtitle('Impulse Response','n','h[n]');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
a.y_location="left";
figure
a=gca();
plot2d3(nx,x)
plot(nx,x,'r.')
a.y_location="left";
a.x_location="origin";
xtitle('Input Response','n','x[n]');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
figure
a=gca();
plot2d3(ny,y)
plot(ny,y,'r.')
a.x_location="origin";
a.y_location="left";
xtitle('Output Response','n','y[n]');
a.children.children.thickness = 3;
a.children.children.foreground= 2;
```

<u>OUTPUT</u>:







RESULT: The given discrete time system has been analysed.

EXP. NO. 23

STEP RESPONSE OF THE SYSTEM

AIM: To determine the step response of the system.

APPARATUS REQUIRED: PC and SCILAB software.

ALGORITHM:

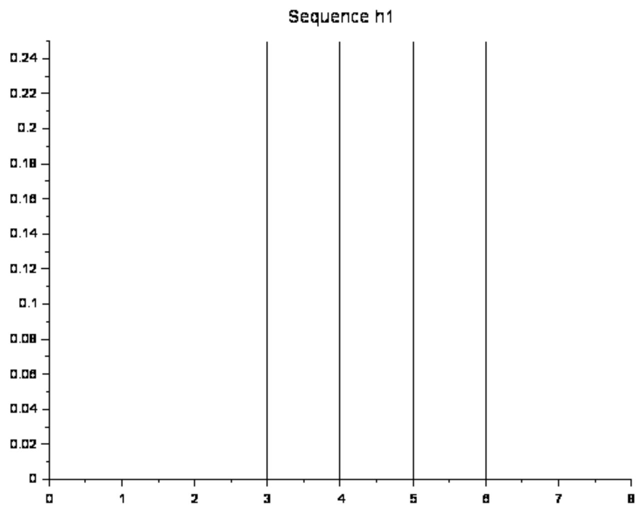
Identify the mathematical representation of the system: Determine the transfer function Apply the step input: The step input signal is a sudden change from 0 to 1 at a specific time.
Represent it as a function that equals 0 for time values less than the step time and 1 for time values greater than or equal to the step time.
Obtain the step response: Once the mathematical solution is obtained, the resulting output represents the step response of the system.

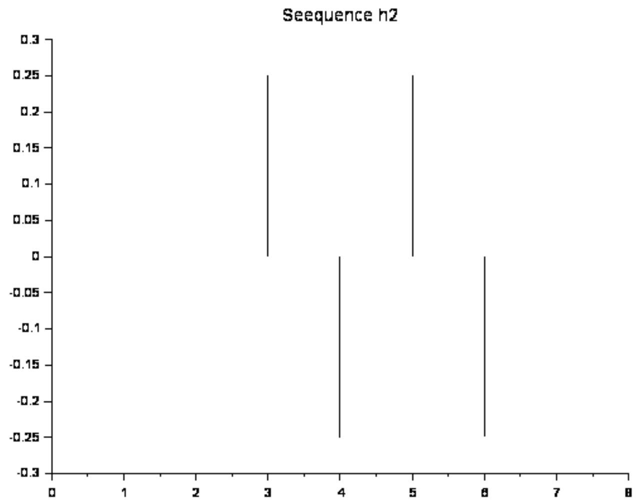
Gn: h1=[0.25, 0.25, 0.25, 0.25] and h2=[0.25, -0.25, 0.25, 0.25]

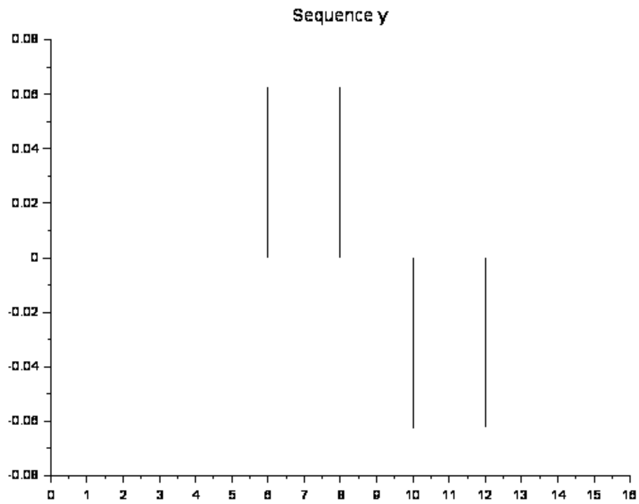
PROGRAM:

```
clc;
n1=0:1:8;
n2=0:1:8;
h1=[0 0 0 .25 .25 .25 .25 0 0]
h2=[0 0 0 .25 -.25 .25 -.25 0 0]
y=convol(h1,h2);
l=length(y);
n3=0:1:l-1;
figure
title('Sequence h1')
plot2d3(n1,h1);
figure
title('Sequence h2')
plot2d3(n2,h2);
figure
title('Sequence y')
plot2d3(n3,y);
```

output:







<u>RESULT</u>: The step response of the system is determined.