UNIVERSITY OF ENGINEERING AND TECHNOLOGY PESHAWAR

SIGNALS & SYSTEMS LAB REPOT

SALAAR KHAN

17PWELE5087

SECTION A

SIR KIFAYAT

ELECTRICAL ENGINEERING DEPARTMENT

LAB NO # 1

INTRODUCTION TO MATRIX , PROGRAMMING IN MATLAB

MATRIX IN MATLAB

DEFINING MATRIX:

```
>> A=[1 2 3 4]


A =

   1    2    3    4     %Define me a matrix A
```

For going to the second row we will always use semicolen.

```
>> A=[1 2 3 4;5 6 7 8]


A =

   1    2    3    4
   5    6    7    8
```

If we want to excess the number 5 then we write:

```
>> A(2,1)


ans =

   5
```

ACCESSING ELEMENTS IN MATRIX

If we want to access the number 8 then

```
>> A(2,4)

ans =

   8
```

SIZE COMMAND

Size command is used for find row and colum size.eg.

```
>> size(A)


ans =

   2    4
```

ROW AND COLUMN VECTORS

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


ROWMATRIX =

   1    2    3
>> columnmatrix=[1;2;3]
columnmatrix =

   1

   2

   3
```

COLON OPERATION

```
>> %t=intial valve:step size:final valve
>> t=1:2:9


t =

   1    3    5    7    9


>> t=1:2:10


t =

   1    3    5    7    9


>> t=1:10
```

```
t =
   1    2    3    4    5    6    7    8    9   10

>> t=9:-2:1

t =
   9    7    5    3    1

>> t=4:2:1

t =
  1×0 empty double row vector
```

BUILT-IN MATRIXES

1. ZEROS MATRIX:

```
>> zeros(3)

ans =
   0    0    0
   0    0    0
   0    0    0

>> zeros(2,3)
ans =
   0    0    0
   0    0    0
```

2. ONES MATRIX:

```
>> ones(3)


ans =

    1    1    1

    1    1    1

    1    1    1

>> ones(2,3)


ans =

    1    1    1

    1    1    1
```

3. EYE MATRIX

Eye matrix is a identity matrix.

```
>> eye(3)


ans =

    1    0    0

    0    1    0

    0    0    1


>> eye(2,3)


ans =

    1    0    0

    0    1    0
```

END KEYWORD:

Its helpful when we don't know what the last index is.

```
>> x=1:10

x =
   1    2    3    4    5    6    7    8    9  10

>> x(3:end)

ans =
   3    4    5    6    7    8    9  10
>> x(3:2:end)
ans =
   3    5    7    9
>> b=[1 2 3;4 5 6];
>> b(end,:)
ans =
   4    5    6
>> b(end,1:2)
ans =
   4    5
>> b(end-1,:)
ans =
   1    2    3
>> b(end,end)
ans =
   6
```

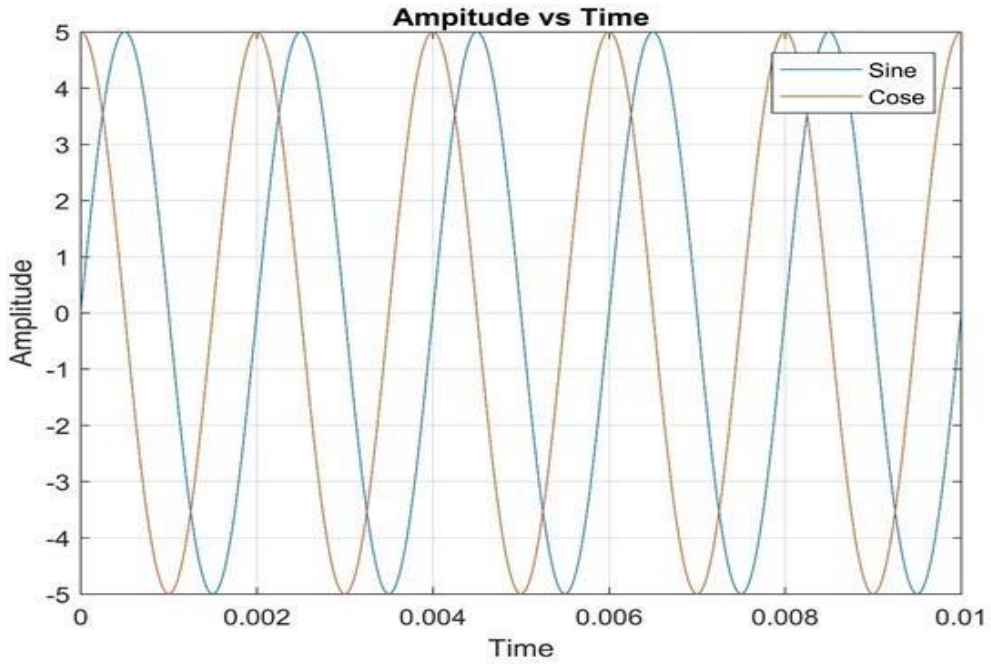
LAB NO # 2

TO PLOT A CONTINOUS TIME SINOCIDAL SIGNAL IN MATLAB

PROGRAM

```
F=500;
T=1/F;
A=5;
t=0:T/100:5*T;
x=A*sin(2*pi*f*t);
plot(t,x);
hold on
y=A*cos(2*pi*f*t);
plot(t,y);
xlabel('time');
ylabel('amplitude');
title('amplitde vs time');
legend('sine','cose');
grid on
```

GRAPH:



HOME TASK

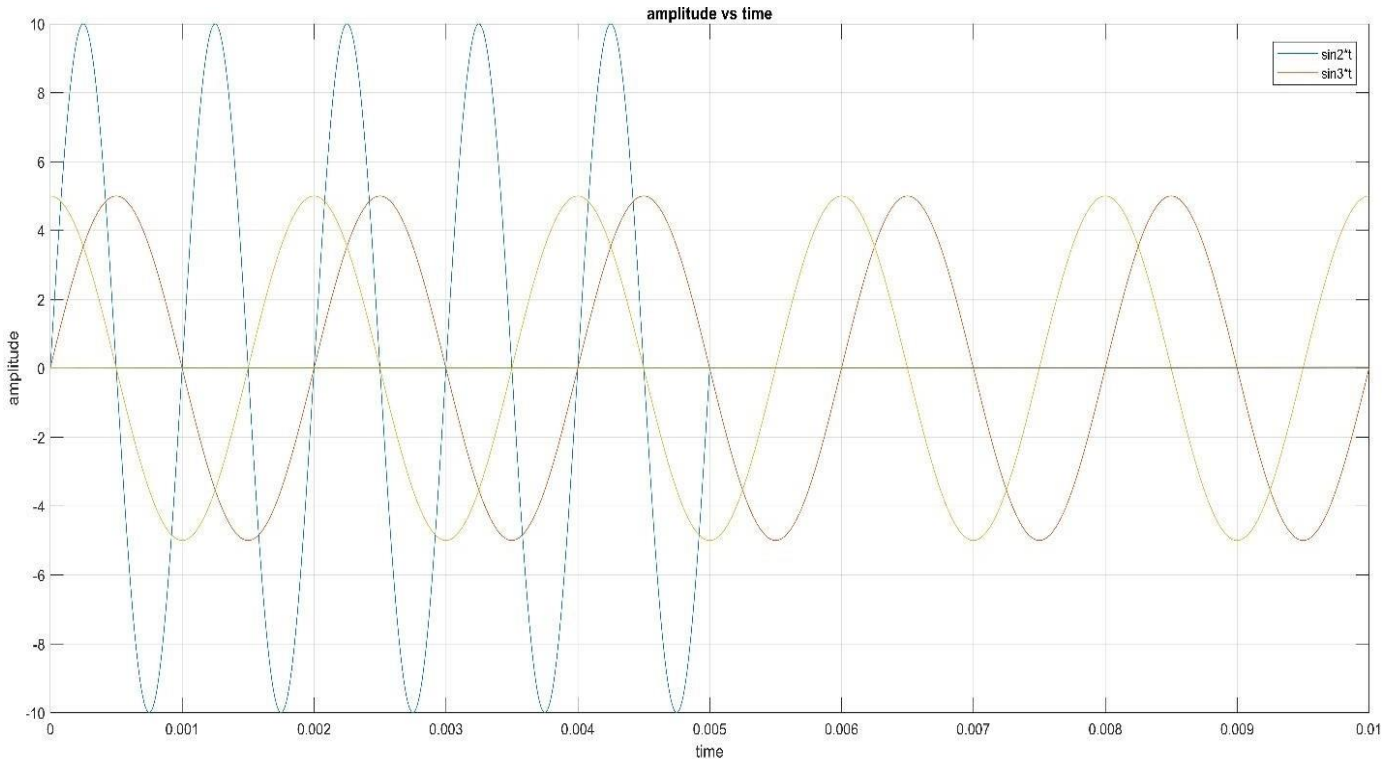
Sin(2*t) & sin(3*t) for t=[-5:0.01:5]

- t=[-5:0.0.1;5];
- x=sin(2*t);
- y=sin(3*t);
- plot(t,x,t,y);
- xlabel('time');
- ylabel('amplitude');
- title('home work');
- legend('sine2t curve & sine3t curve').

These graphs have difference in frequency.

- Sin2t has low frequency and high wavelength as compared to sin3t which ash high frequency and low wavelength in comparison.

GRAPH



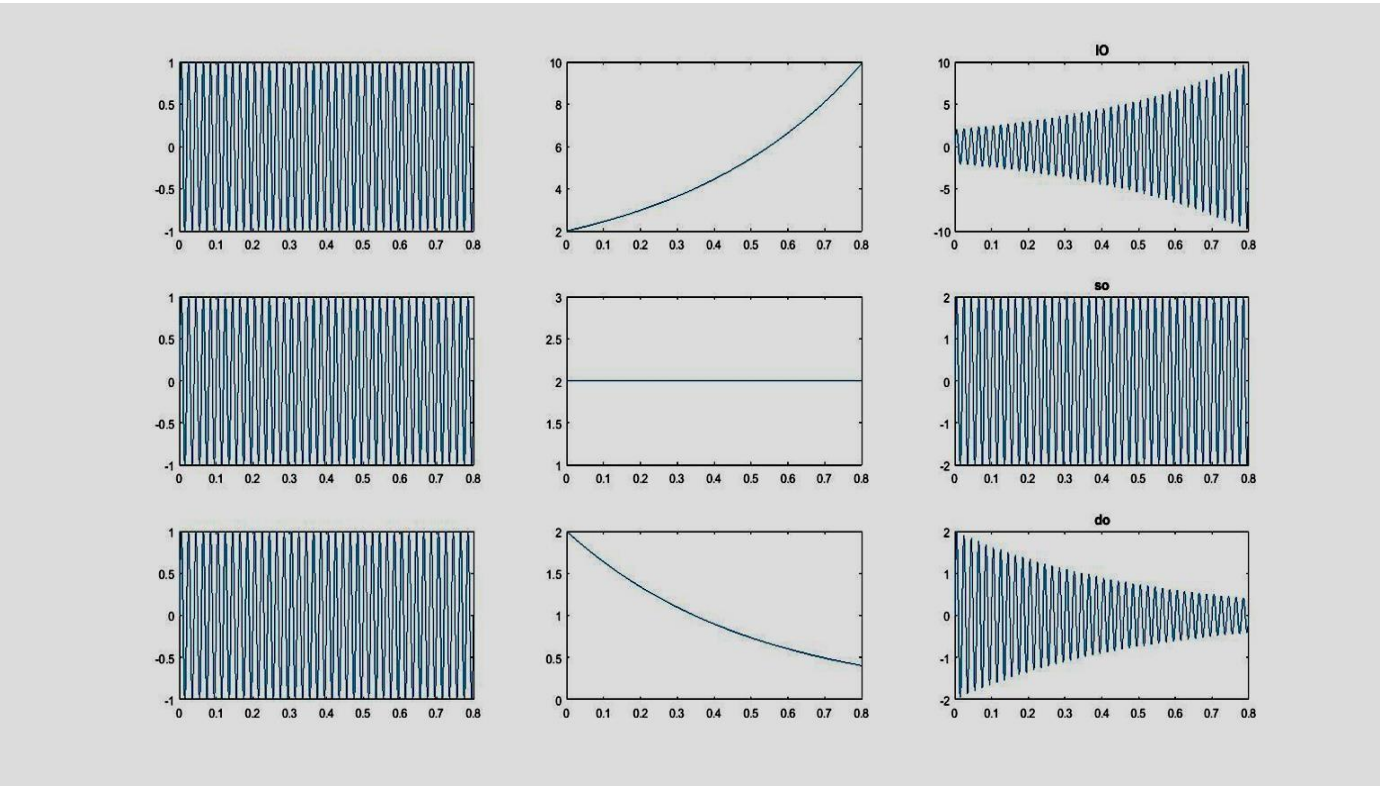
LAB NO # 3

TITLE: TO PLOT AN OSCILLATORY SIGNAL IN MATLAB

PROGRAM:

```
f=50;
T=1/f;
c=2;
t=0:T/20:40*T;
a=2;
x1=sin(2*pi*f*t);
x2=c*exp(a*t);
x3=x1.*x2;
subplot(3,3,1);
plot(t,x1);
subplot(3,3,2);
plot(t,x2);
subplot(3,3,3);
plot(t,x3);
title('IO');
a=0;
y1=sin(2*pi*f*t);
y2=c*exp(a*t);
y3=y1.*y2;
subplot(3,3,4);
plot(t,y1);
subplot(3,3,5);
plot(t,y2);
subplot(3,3,6);
plot(t,y3);
title('so');
a=-2;
z1=sin(2*pi*f*t);
z2=c*exp(a*t);
z3=z1.*z2;
subplot(3,3,7);
plot(t,z1);
subplot(3,3,8);
plot(t,z2);
subplot(3,3,9);
plot(t,z3);
title('do');
```

GRAPH:



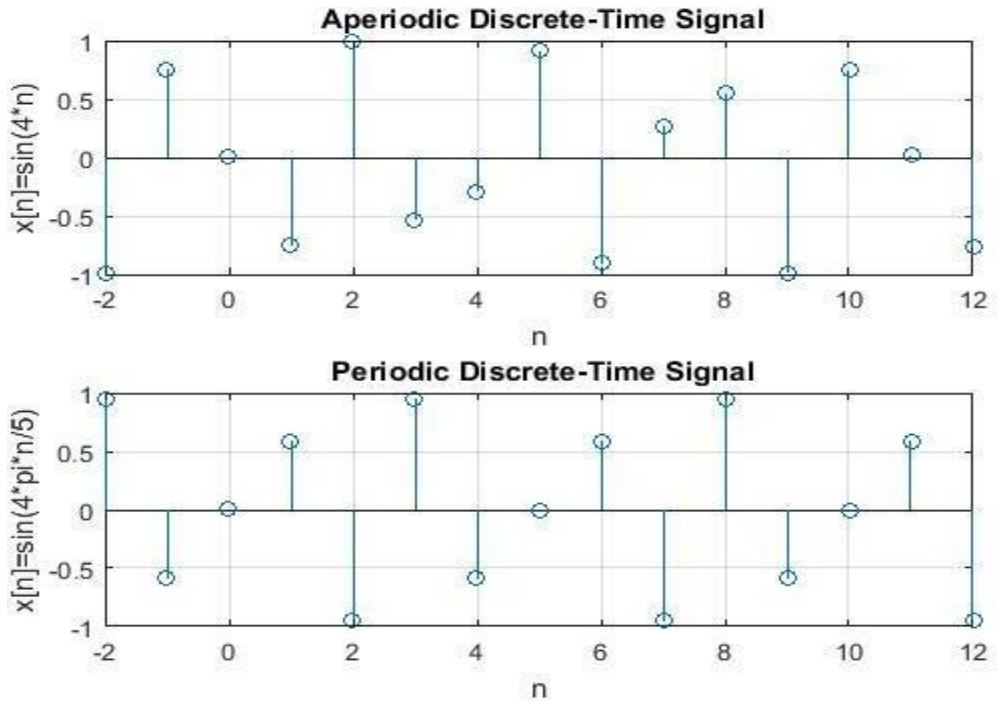
LAB NO # 4

TO PLOT DISCRETE TIME SIGNALS IN MATLAB

PROGRAM:

```
n = -2:12;            % define n values in a vector
x1 = sin(4*n);        % define x1[n]=sin(4*n)
x2 = sin(4*pi*n/5); % define x2[n]=sin(4*pi*n/5)
subplot(2,1,1)        % plot and label first signal
stem(n,x1)
grid
xlabel('n')
ylabel('x[n]=sin(4*n)')
title(' Aperiodic Discrete-Time Signal')
subplot(2,1,2)        % plot and label second signal
stem(n,x2)
grid
xlabel('n')
ylabel('x[n]=sin(4*pi*n/5)')
title(' Periodic Discrete-Time Signal')
```

RESULT:



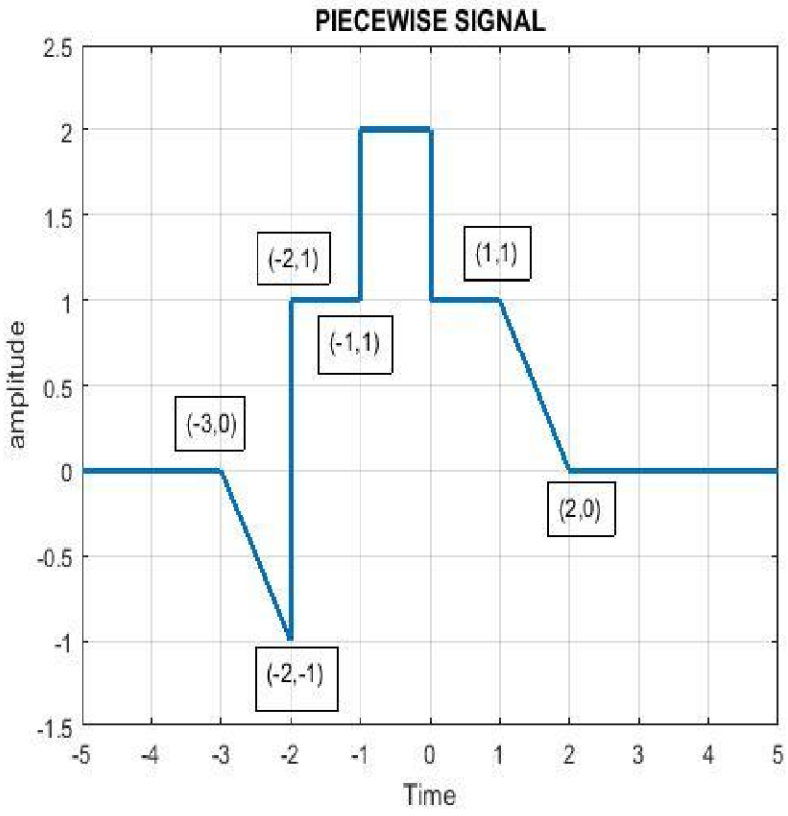
LAB NO # 5

TO PLOT A PIECEWISE SIGNAL IN MATLAB

PROGRAM:

```
t0=-5:0.01:-3;
x0=zeros(size(t0));
t1=-3:0.01:-2;
x1=-t1-3;
t2=-2:0.01:-1;
x2=ones(size(t2));
t3=-1:0.01:0;
x3=2*ones(size(t3));
t4=0:0.01:1;
x4=ones(size(t4));
t5=1:0.01:2;
x5=-t5+2;
t6=2:0.01:5;
x6=zeros(size(t6));
x=[x0 x1 x2 x3 x4 x5 x6]
t=[t0 t1 t2 t3 t4 t5 t6]
plot(t,x,'linewidth',2)
xlabel('Time')
ylabel('amplitude')
title('PIECEWISE SIGNAL')
axis([-5 5 -1.5 2.5])
grid on
```

GRAPH:



LAB NO # 6

TO STUDY AND PLOT TIME TRANSFORMATION OF SIGNAL IN MATLAB

TIME TRANSFORMATION:-

- Time Shift
- Time Scaling
- Time Reversal

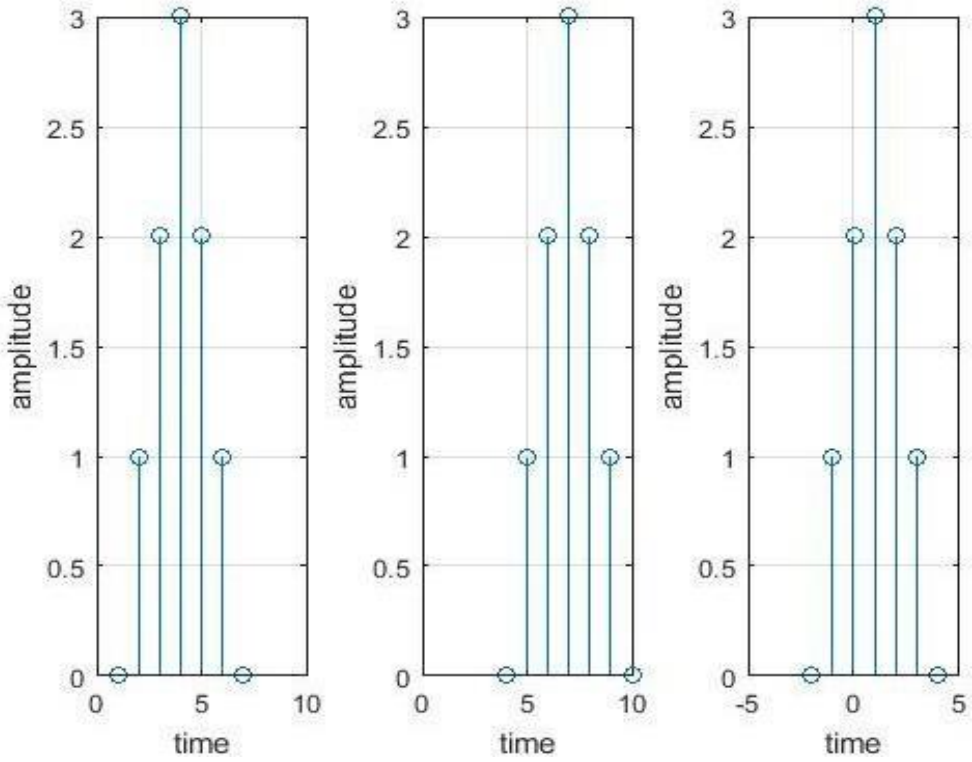
1.TIME SHIFT:

a) DISCRETE TIME SHIFT:

PROGRAM:

```
t=[1 2 3 4 5 6 7];
x=[0 1 2 3 2 1 0];
subplot(1,3,1)
stem(t,x) %plotting the original signal
xlabel( 'time' )
ylabel( 'amplitude')
grid on
a=t+3; % delay the signal
subplot(1,3,2)
stem(a,x) %plotting delay signal
xlabel( 'time' )
ylabel( 'amplitude' )
grid on
b=t-3; % advance the signal
subplot(1,3,3)
stem(b,x) %plotting advanced signal
xlabel('time')
ylabel('amplitude')
grid on
```

RESULT:

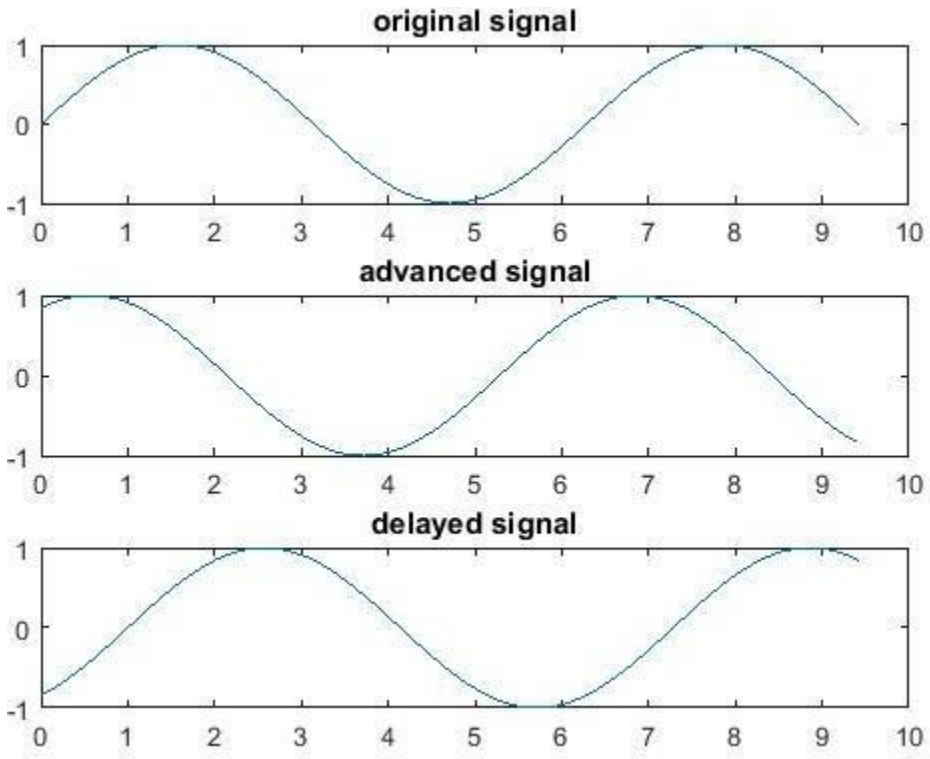


b). CONTINUOUS TIME SHIFT:

PROGRAM:

```
t=0:0.01:3*pi;
y=sin(t);%orignal signal
subplot(3,1,1)
plot(t,y)
title('original signal')
y1=sin(t+1) %advanced signal
subplot(3,1,2)
plot(t,y1)
title('advanced signal')
y2=sin(t-1)%delayed signal
subplot(3,1,3)
plot(t,y2)
title('delayed signal')
```

RESULT:

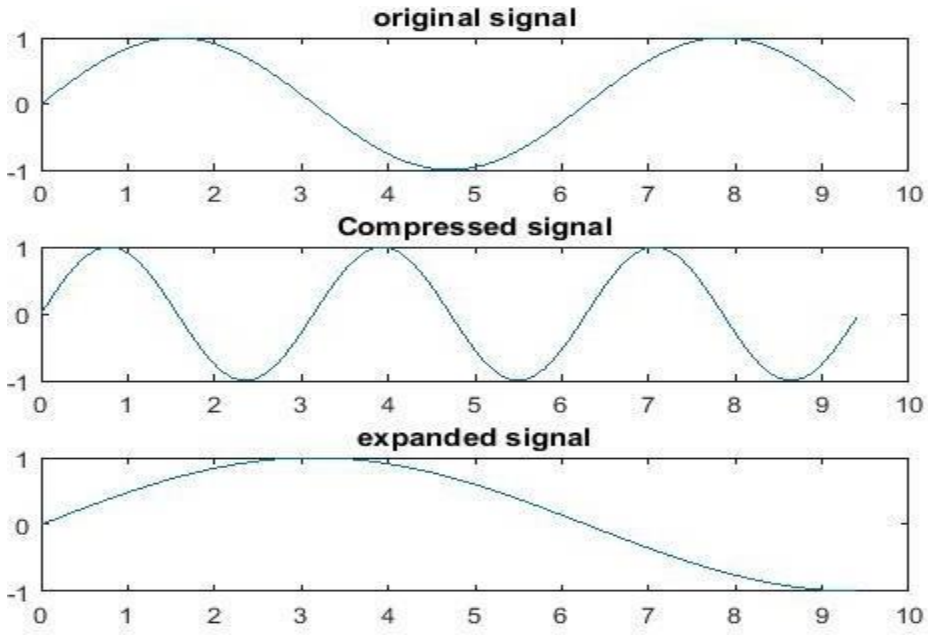


2. TIME SCALING:

PROGRAM:

```
t=0:0.1:3*pi;
X=sin(t)
subplot(3,1,1)
plot(t,X)
title('original signal')
Y=sin(2*t) %compressed signal
subplot(3,1,2)
plot(t,Y)
title('Compressed signal')
Z=sin(t/2) %expanded signal
subplot(3,1,3)
plot(t,Z)
title('expanded signal')
```

RESULT:

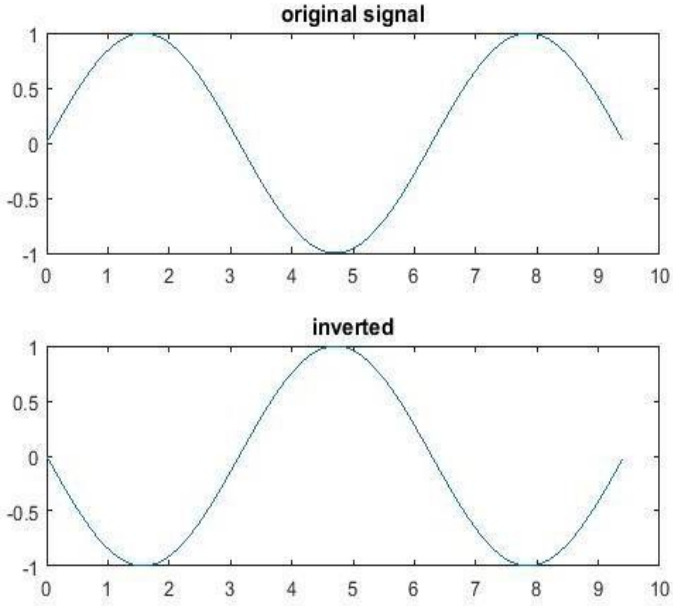


3.TIME REVESAL:

PROGRAM:

- t= 0:0.1:3*pi;
- y=sin(t)
- subplot(2,1,1)
- plot(t,y)
- title('original signal')
- y1=sin(-t)
- subplot(2,1,2)
- plot(t,y1)
- title('inverted')

RESULT:



LAB # 7(A)

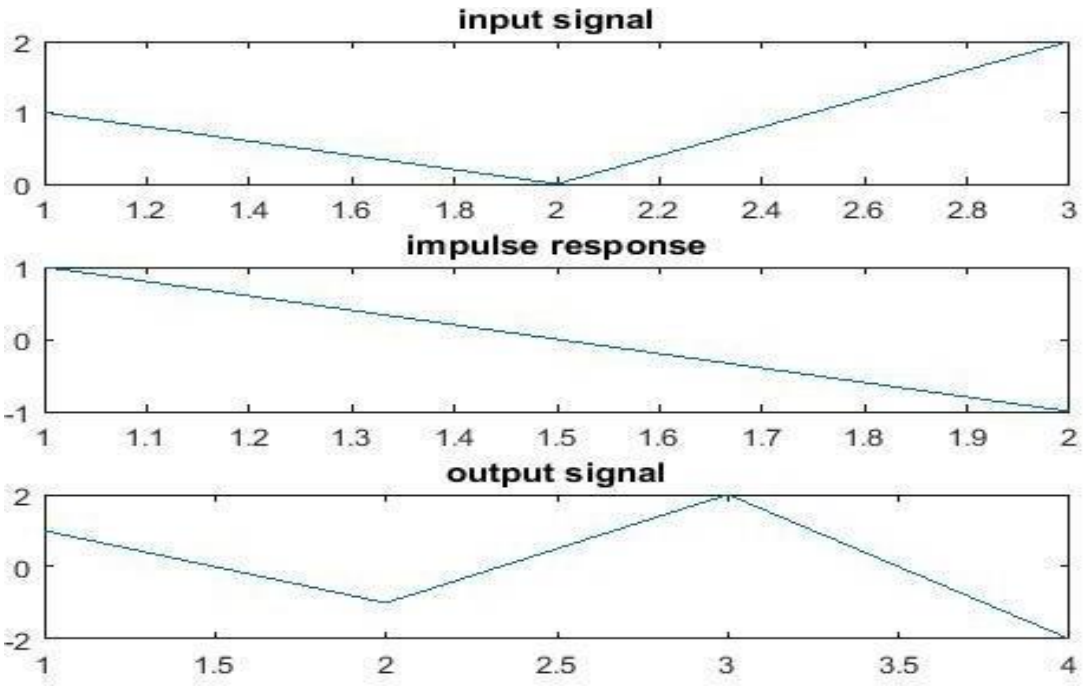
CONTINUOUS TIME CONVOLUTION IN MATLAB

PROGRAM:

```
x=[1 0 2];
h=[1 -1];
y=conv(x,h)
subplot(3,1,1)
plot(x)
title('input signal')
subplot(3,1,2)
plot(h)
title('impulse response')
subplot(3,1,3)
plot(y)
title('output signal')
```

RESULT

```
y =

   1   -1    2   -2
```



LAB # 7(B)

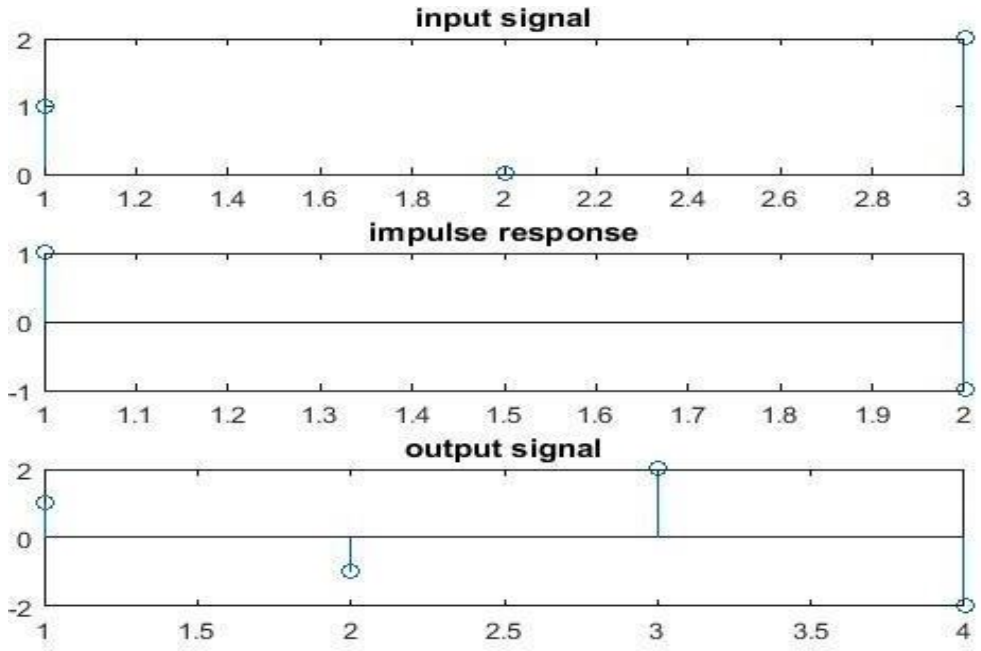
TITLE: DISCRETE TIME CONVOLUTION IN MATLAB

PROGRAM:

```
x=[1 0 2];
h=[1 -1];
y=conv(x,h)
subplot(3,1,1)
stem(x)
title('input signal')
subplot(3,1,2)
stem(h)
title('impulse response')
subplot(3,1,3)
stem(Y)
title('output signal')
```

RESULT:

```
y =

   1   -1    2   -2
```



LAB # 8

TO SOLVE DIFFERENTIAL EQUATIONS IN MATLAB

PROGRAM:

```
syms x(t)
ode=diff(x,t,2)==30
xsol=dsolve(ode)
```

RESULT:

ode(t) =

diff(x(t), t, t) == 30

xsol =

C2 + t*(C1 + 15*t)

- PUT INTIAL VALVE IN THIS DIFFERENTIAL EQUATIONS

PROGRAM:

```
syms x(t)
xprime=diff(x);
ode=diff(x,t,2)==30;
cond1=x(0)==0;
cond2=xprime(0)==1;
conds=[cond1 cond2];
xsol=dsolve(ode,conds)
```

RESULT:

xsol =

t*(15*t + 1

LAB # 9

IMPLEMENTATION OF LAPLACE TRANSFORM
IMPLEMENTATION OF Z TRANSFORM.

LAPLACE TRANSFOM IN MATLAB:-

PROGRAM:

```
syms imp stp rmp x y t ex
imp=dirac(t)
stp=heaviside(t)
rmp=t
parb=t^2
x=sin(2*t)
y=cos(2*t)
ex=exp(-3*t)
fprintf('the laplace transform of unit impulse signal is\n')
laplace(imp)
fprintf('the laplace transform of unit step signal is\n')
laplace(stp)
fprintf('the laplace transform of ramp signal is\n')
laplace(rmp)
fprintf('the laplace transform of parabolic signal is\n')
laplace(parb)
laplace(x)
laplace(y)
laplace(ex)
```

RESULT:

```
imp =dirac(t)

stp =heaviside(t)

rmp =t

parb =t^2

x =sin(2*t)

y =cos(2*t)
```

- ex =exp(-3*t)
- the laplace transform of unit impulse signal is

  ans =1
- the laplace transform of unit step signal is
- ans =1/s
- the laplace transform of ramp signal is

  ans =1/s^2
- the laplace transform of parabolic signal is

  ans =2/s^3
- ans =2/(s^2 + 4)
- ans =s/(s^2 + 4)
- ans =1/(s + 3)

TO SOLVE Z TRANSFORM IN MATLAB

PROGRAM:

```
syms n w
fprintf('the z transfom of unit impulse signal is\n')
a=ztrans(kroneckerDelta(n))
fprintf('the z transfom of shifted impulse signal is\n')
b=ztrans(kroneckerDelta(n-1))
fprintf('the z transfom of unit step signal is\n')
c=ztrans(heaviside(n))
fprintf('the z transfom of cosine signal is\n')
d=ztrans(cos(w*n))
fprintf('the z transfom of sine signal is\n')
e=ztrans(sin(w*n))
f=ztrans(n)
g=ztrans(n^2)
h=ztrans((2^n)*heaviside(n))
```

RESULT:

- the z transfom of unit impulse signal is

  a =

  1

- the z transfom of shifted impulse signal is

  b =

  1/z

- the z transfom of unit step signal is

  c =

  1/(z - 1) + 1/2

- the z transfom of cosine signal is

  d =

  (z*(z - cos(w)))/(z^2 - 2*cos(w)*z + 1)

- the z transfom of sine signal is

  e =

  (z*sin(w))/(z^2 - 2*cos(w)*z + 1)

- f =

  z/(z - 1)^2

- g =

  (z*(z + 1))/(z - 1)^3

- h =

  2/(z - 2) + 1/2

LAB # 10 (A)

TO STUDY TRANSFER FUNCTION USING MATLAB

CONTINUOUS-TIME TRANSFER FUNCTION.

PROGRAM:

- num=[10 0]
- den=[1 5 6]
- h=tf(num,den)

RESULT:

- num =

  10 0

- den =

  1 5 6

- h =

```
        10 s
  -------------
  s^2 + 5 s + 6
```

- Continuous-time transfer function.

LAB # 10(B)

TO FIND TRANSFER FUNCTION FROM ZEROS AND POLES IN MATLAB

PROGRAM:

```
z=input('enter zeros')
p=input('enter poles')
k=input('enter gain')
[num,den]=zp2tf(z,p,k)
tf(num,den)
```

RESULT:

```
enter zeros0

z =

     0

enter poles[-2 -3]

p =

    -2    -3

enter gain10

k =

    10

num =

     0    10     0

den =

     1     5     6

ans =

          10 s
  -------------
  s^2 + 5 s + 6

Continuous-time transfer function.
```

LAB # 10(C)

FIND POLES-ZEROS FROM TRANSFER FUNCTION IN MATLAB

PROGRAM:

```
num=[10 0]
den=[1 5 6]
[z p k]=tf2zp(num,den)
```

RESULT:

```
num =

    10     0


den =

     1     5     6


z =

     0


p =

    -3.0000
    -2.0000


k =

    10
```

LAB # 10(D)

TITLE: TO FIND STEP RESPONSE AND IMPULSE RESPONSE OF A SYSTEM IN MATLAB

PROGRAM:

```
syms s c
c=(2*s+2)/(s^3+11*s^2+30*s)
ilaplace(c)
```

RESULT:

```
c =

(2*s + 2)/(s^3 + 11*s^2 + 30*s)
```

```
ans =

(8*exp(-5*t))/5 - (5*exp(-6*t))/3 + 1/15
```

LAB # 11

INTRODUCTION TO SIMULINK

SIMULINK:

MATLAB and simulink are two very different pieces of software with radically different approaches to modeling of signals and systems. MATLAB is an imperative programming language, whereas simulink is a block diagram language. Simulink is used for a variety of purposes, but mainly for the simulation of real time systems. The function of simulink is very similar to that of electronic work bench or circuit designer.

USING SIMULINK:

To create a simulation in simulink, a new file is created first. In this file, objects from the simulink library browser are inserted. If we are not sure in which tool box the object would lie, we can use the search command.

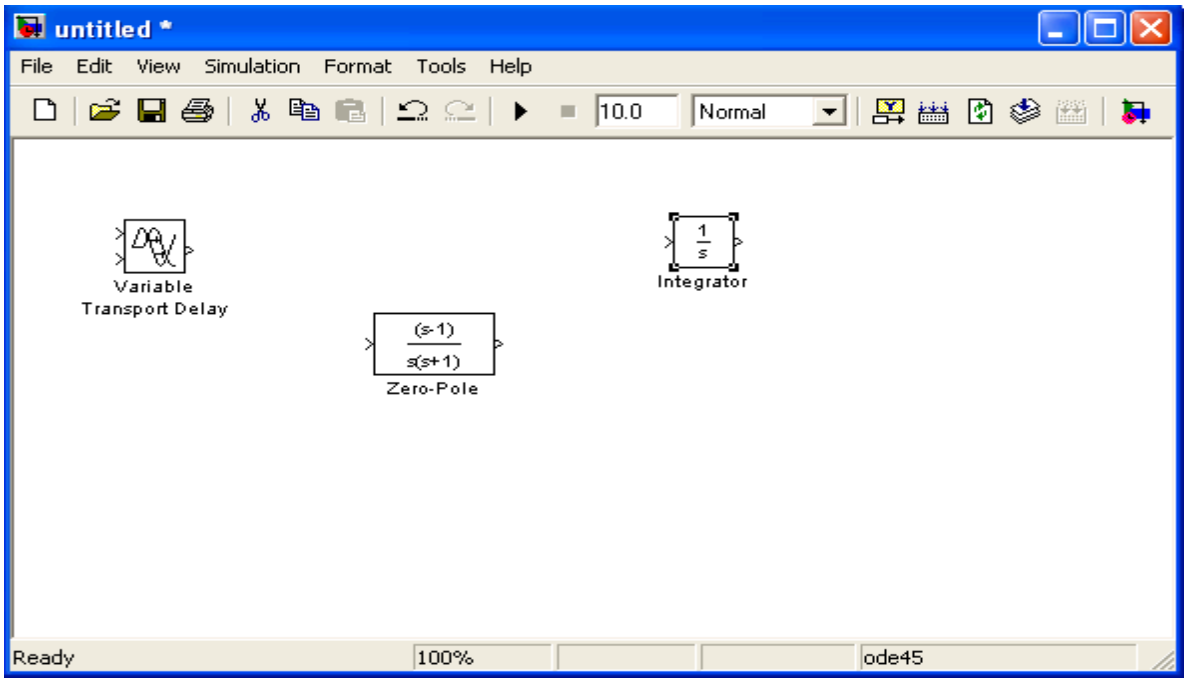
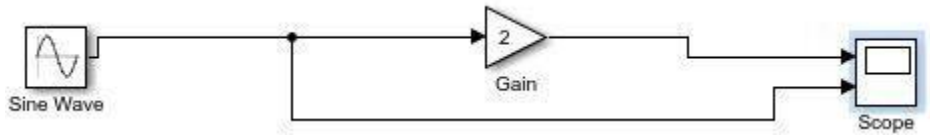
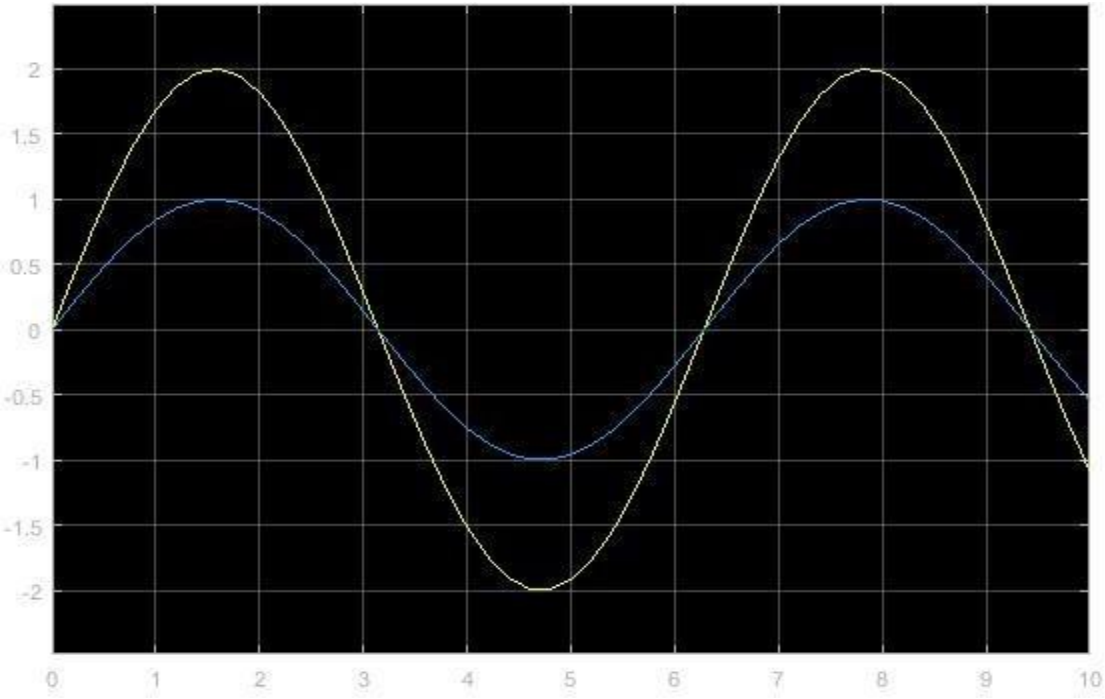


Figure 2: Simulation files in Simulink

FIND SIMPLE MODEL IN SIMULINK



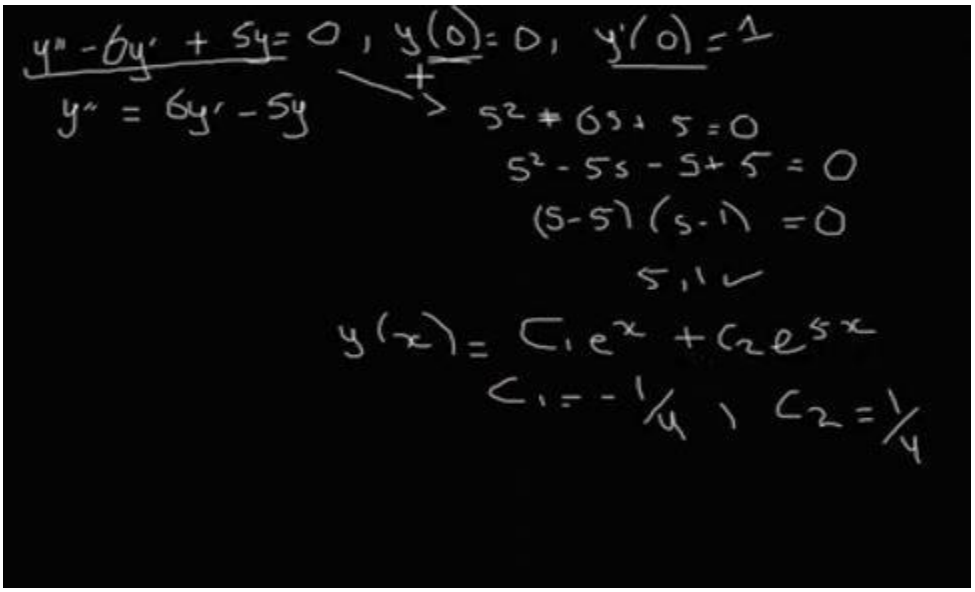
RESULT:



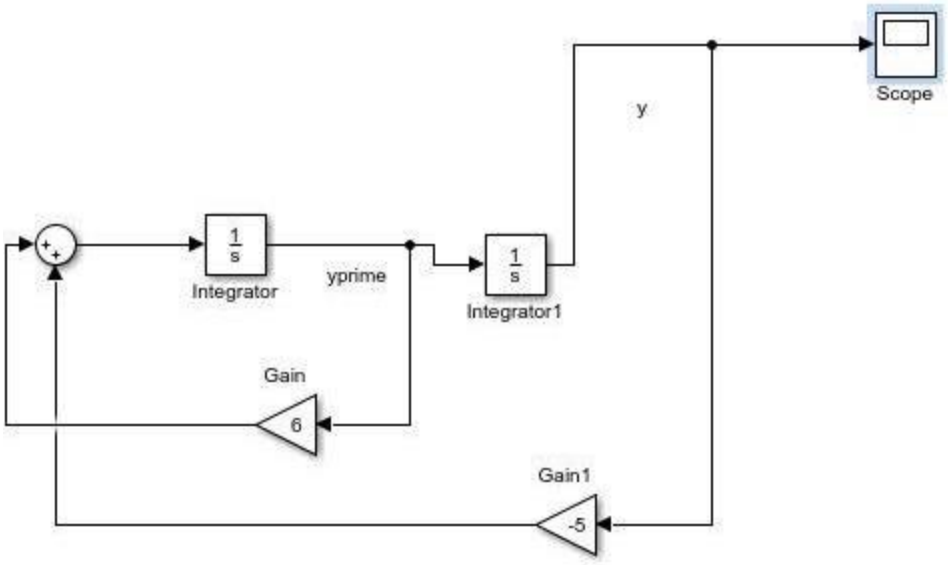
LAB # 12

TITLE: TO SOLVE DIFFERENTIAL EQUATION IN SIMULINK

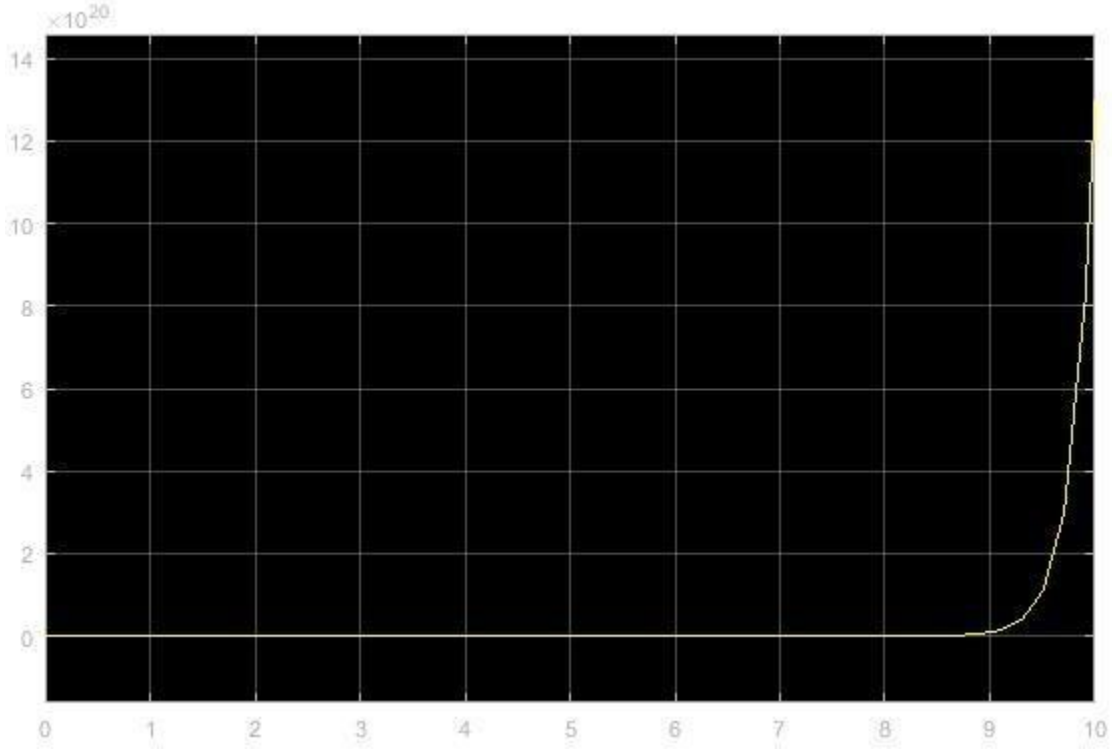
FOR EXAMPLE



SIMUKINK DIAGRAM



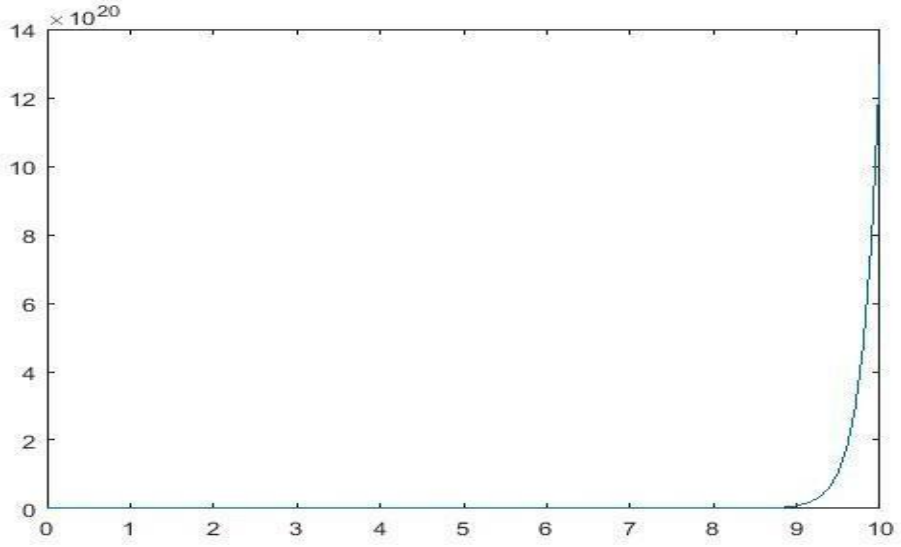
SCOPE RESULT:



PROGRAM IN MATLAB:

- t=0:0.1:10
- x=-0.25*exp(t)+0.25*exp(5*t)
- plot(t,x)

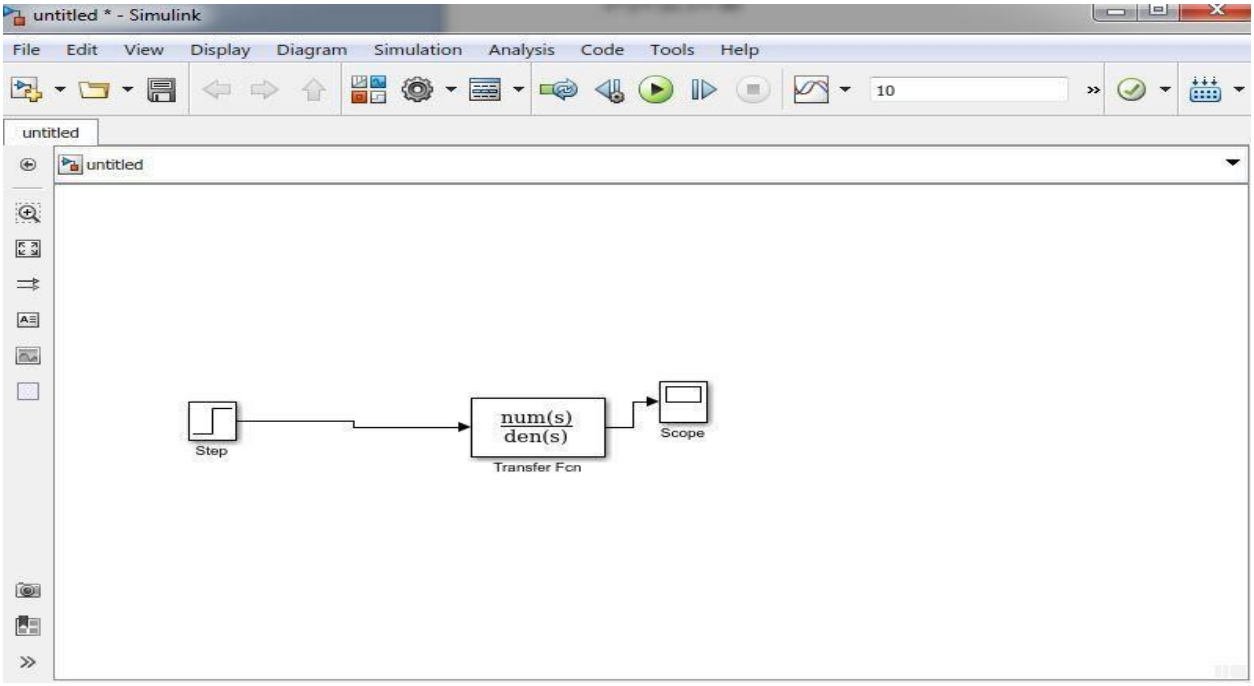
RESULTANT GRAPH:



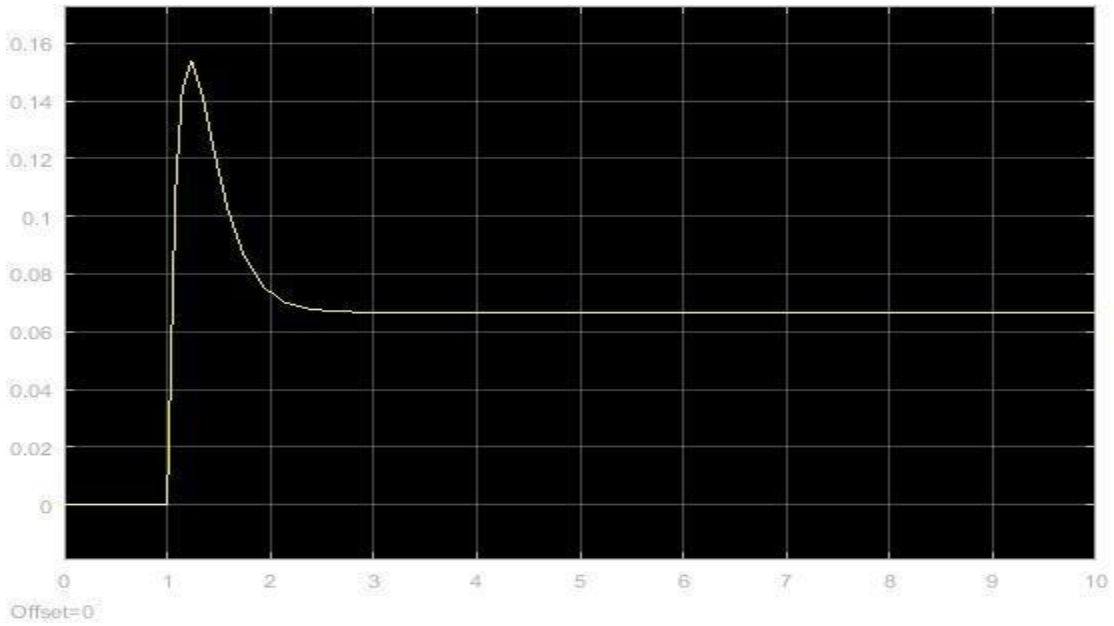
LAB # 13

TO FIND STEP RESPONSE AND IMPULSE RESPONSE OF TRANSFER FUNCTION USING SIMULINK

STEP RESPONSE IN SIMULINK:



RESULTANT GRAPH:



FIND IN MATLAB

PROGRAM:

- num=[2 2]
- den=[1 11 30]
- c=tf(num,den)
- step(c)

RESULT:

- num =

  2 2

- den =

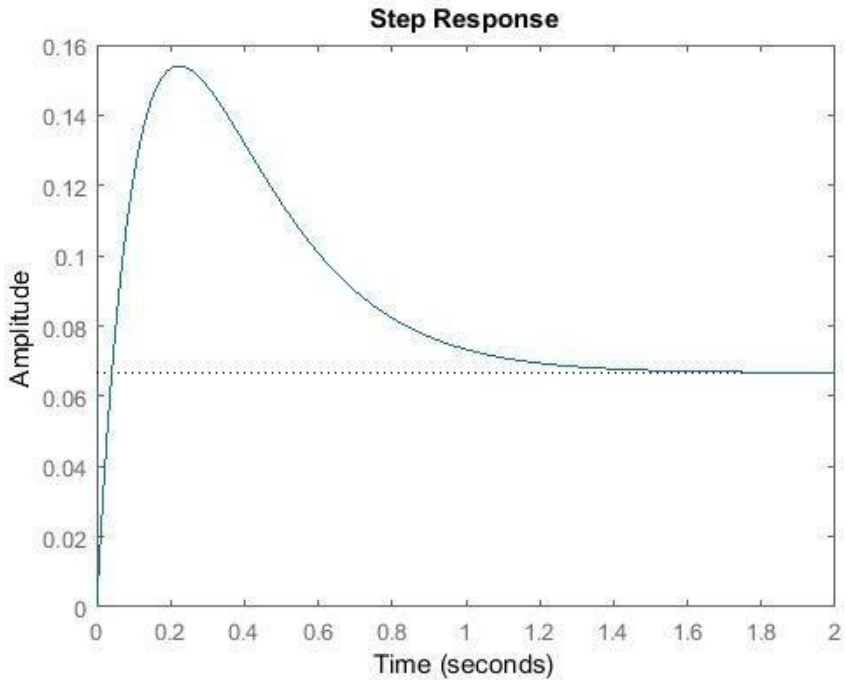
  1 11 30

- c =

  2 s + 2
  ----
  s^2 + 11 s + 30

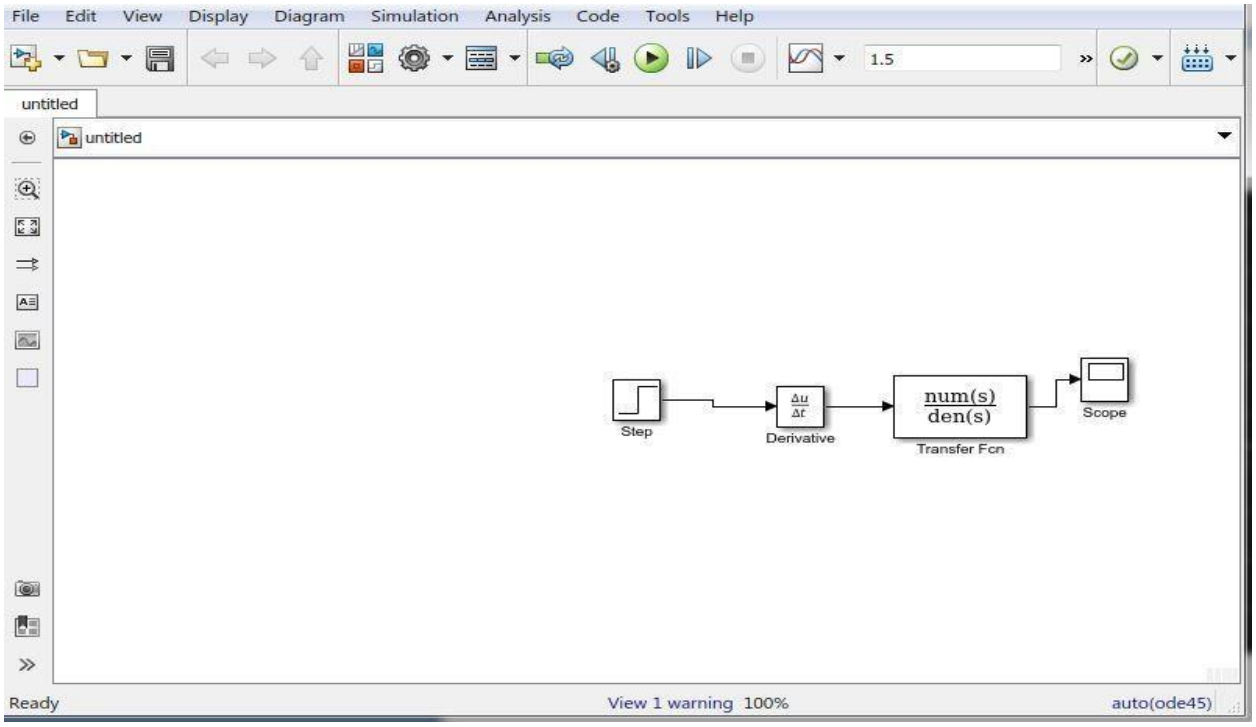
➢ Continuous-time transfer function.

GRAPH

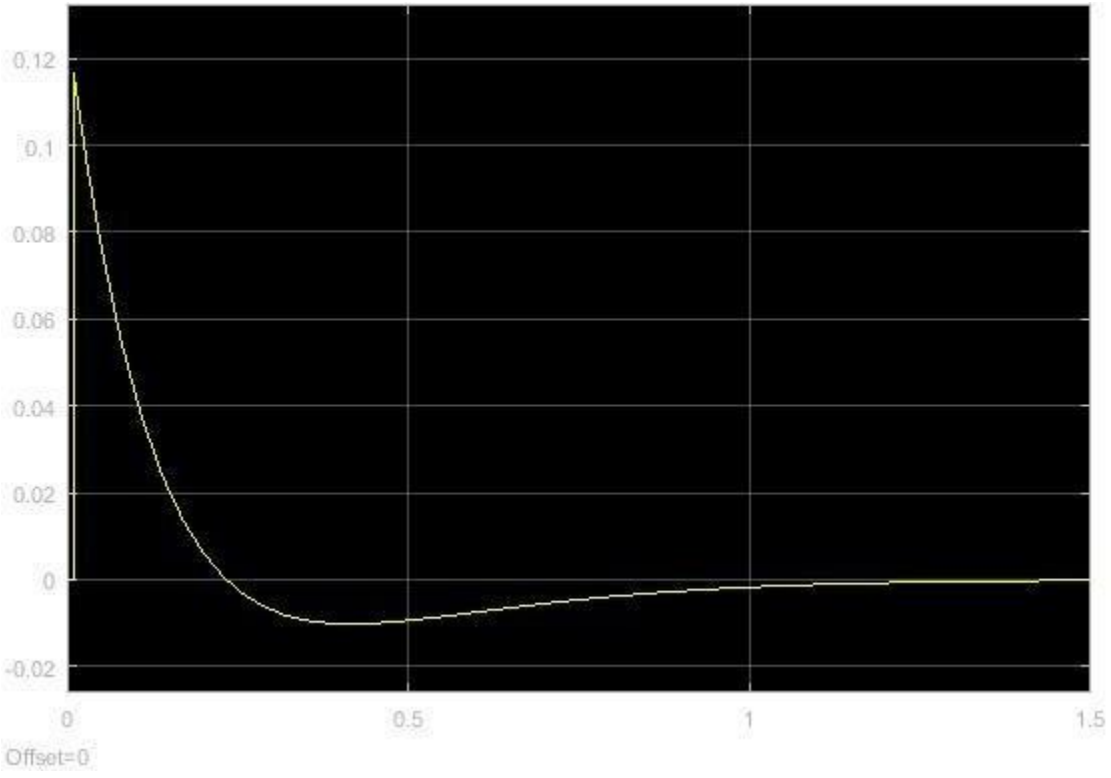


IMPULSE PRESONSE

IN SIMULINK:



GRAPH:



IMPULSE RESPONSE IN MATLAB

PROGRAM:

- num=[2 2]
- den=[1 11 30]
- c=tf(num,den)
- impulse(c)

RESULT:

- num =

  2 2

- den =

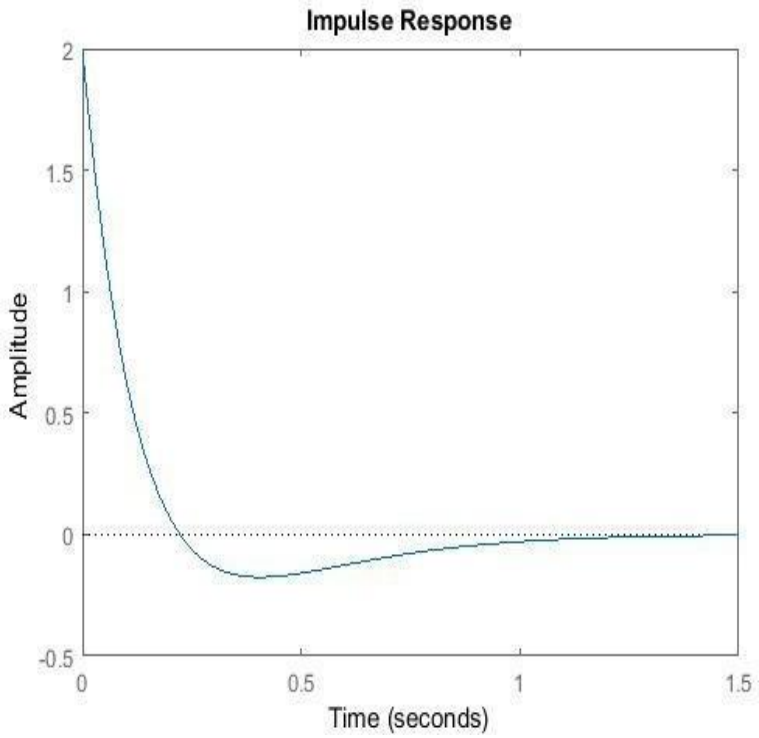
  1 11 30

- c =

  2 s + 2
  ---
  s^2 + 11 s + 30

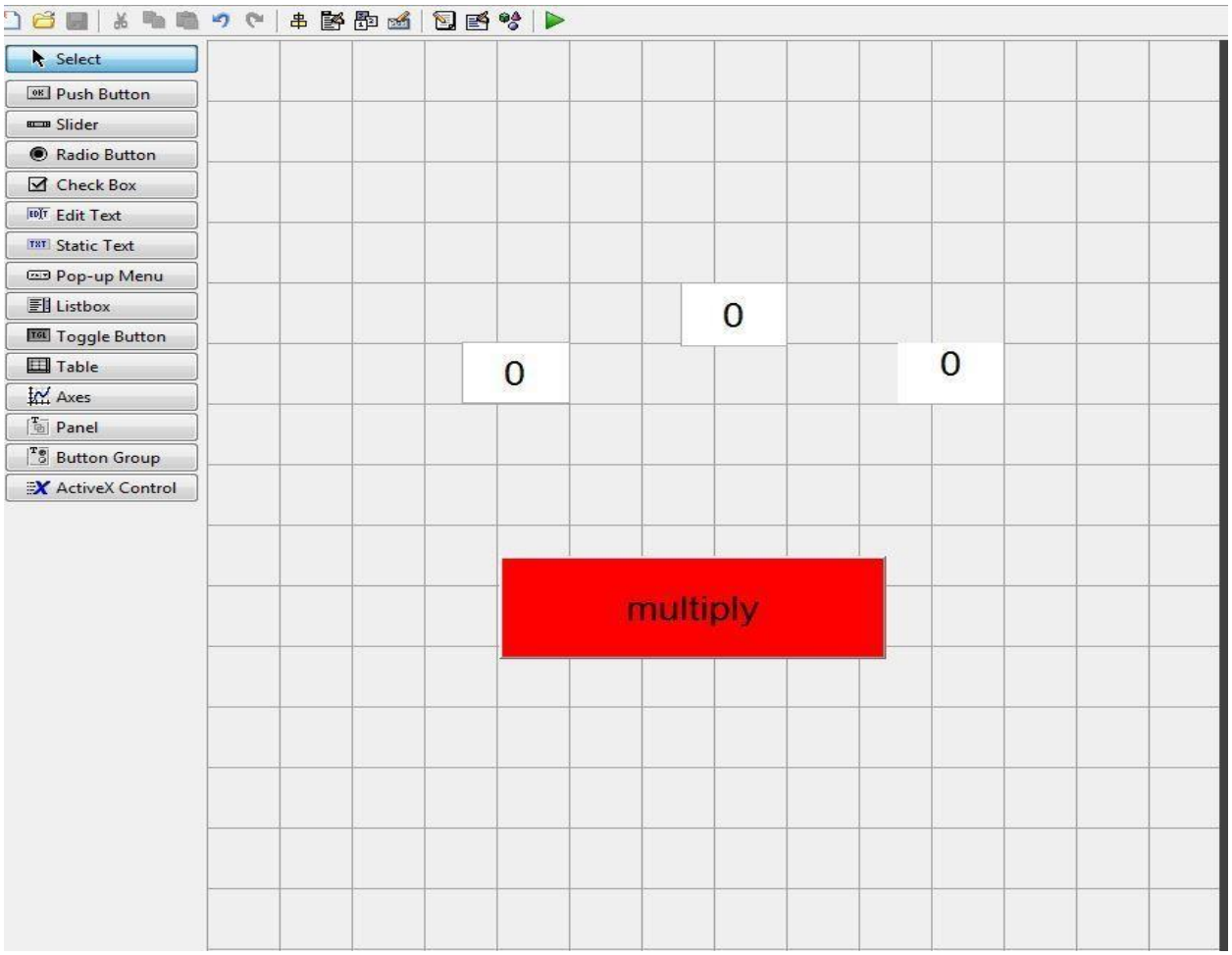
- Continuous-time transfer function.

GRAPH:



LAB # 14

INTRODUCTION TO GUI



PROGRAM OUT:

```
function varargout = lab14(varargin)
% LAB14 MATLAB code for lab14.fig
%      LAB14, by itself, creates a new LAB14 or raises the existing
%      singleton*.
%
%      H = LAB14 returns the handle to a new LAB14 or the handle to
%      the existing singleton*.
%
%      LAB14('CALLBACK',hObject,eventData,handles,...) calls the local
```

```
%     function named CALLBACK in LAB14.M with the given input arguments.
%
%     LAB14('Property','Value',...) creates a new LAB14 or raises the
%     existing singleton*. Starting from the left, property value pairs are
%     applied to the GUI before lab14_OpeningFcn gets called.  An
%     unrecognized property name or invalid value makes property application
%     stop. All inputs are passed to lab14_OpeningFcn via varargin.
%
%     *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%     instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help lab14

% Last Modified by GUIDE v2.5 02-Sep-2020 13:21:34

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton', gui_Singleton, ...
                   'gui_OpeningFcn', @lab14_OpeningFcn, ...
                   'gui_OutputFcn', @lab14_OutputFcn, ...
                   'gui_LayoutFcn', [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before lab14 is made visible.
function lab14_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)
% varargin command line arguments to lab14 (see VARARGIN)

% Choose default command line output for lab14
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);
```

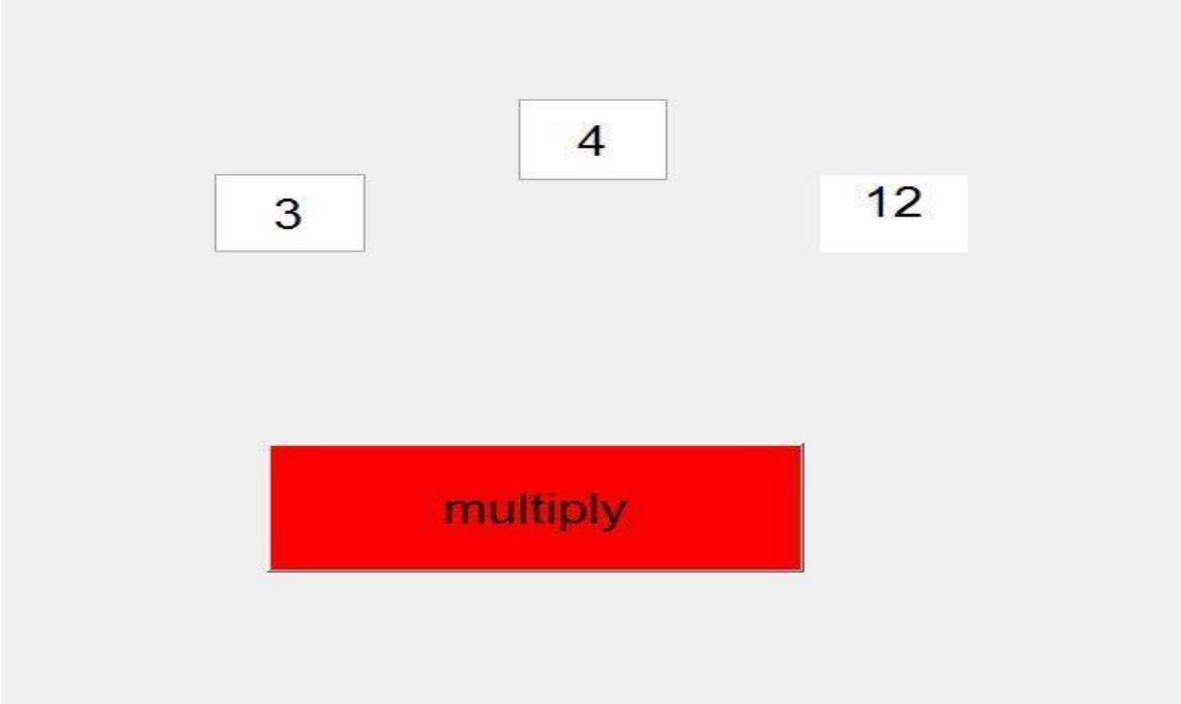
```
% UIWAIT makes lab14 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = lab14_OutputFcn(hObject, eventdata, handles)
% varargout cell array for returning output args (see VARARGOUT);
% hObject handle to figure
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


function N2_Callback(hObject, eventdata, handles)
% hObject handle to N2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N2 as text
%        str2double(get(hObject,'String')) returns contents of N2 as a double

% --- Executes during object creation, after setting all properties.
function N2_CreateFcn(hObject, eventdata, handles)
% hObject handle to N2 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function N1_Callback(hObject, eventdata, handles)
% hObject handle to N1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N1 as text
%        str2double(get(hObject,'String')) returns contents of N1 as a double

% --- Executes during object creation, after setting all properties.
function N1_CreateFcn(hObject, eventdata, handles)
% hObject handle to N1 (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
```

```
%      See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in multiplypushbutton.
function multiplypushbutton_Callback(hObject, eventdata, handles)
% hObject handle to multiplypushbutton (see GCBO)
% eventdata reserved - to be defined in a future version of MATLAB
% handles structure with handles and user data (see GUIDATA)

x=str2double(get(handles.N1,'String'));
y=str2double(get(handles.N2,'String'));
product=x*y;
set(handles.result,'String',product);
```

RESULT:

- 3*4

  =12



LAB # 15

USING FVTOOL TO DETERMINE

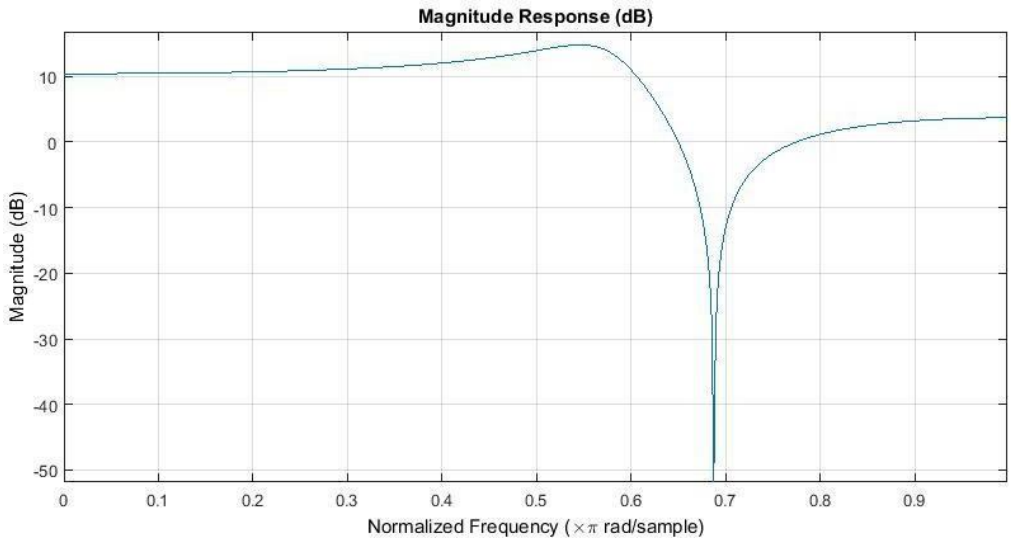
- Magnitude response
- Phase response
- Poles-zero plot
- Impulse response
- Step response

PROGRAM:

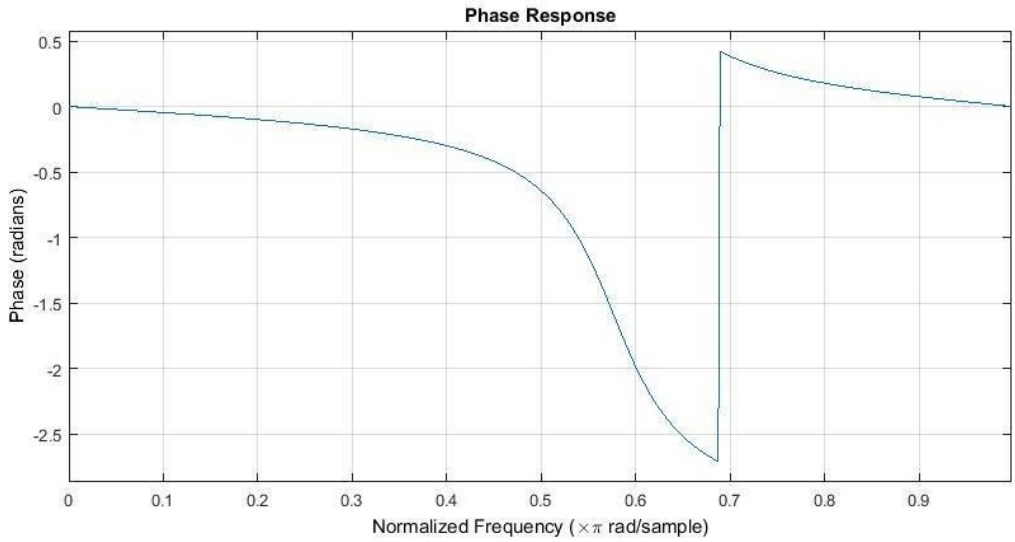
```
num=[2.2403 2.4908 2.2403];
den=[1 0.4 0.7];
fvtool(num,den)
```

RESULT

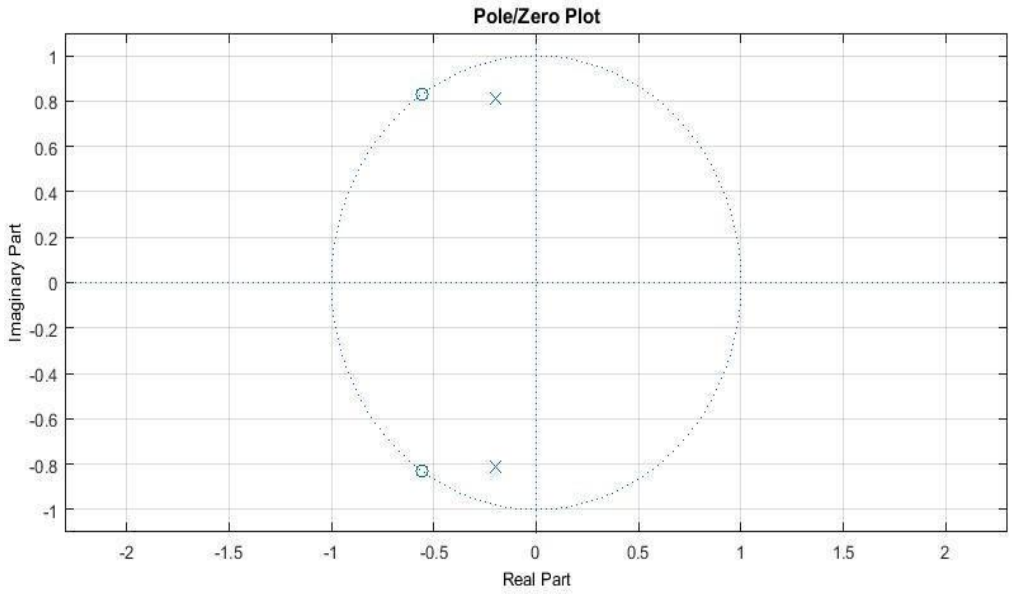
A) MAGNITUDE RESONSE:



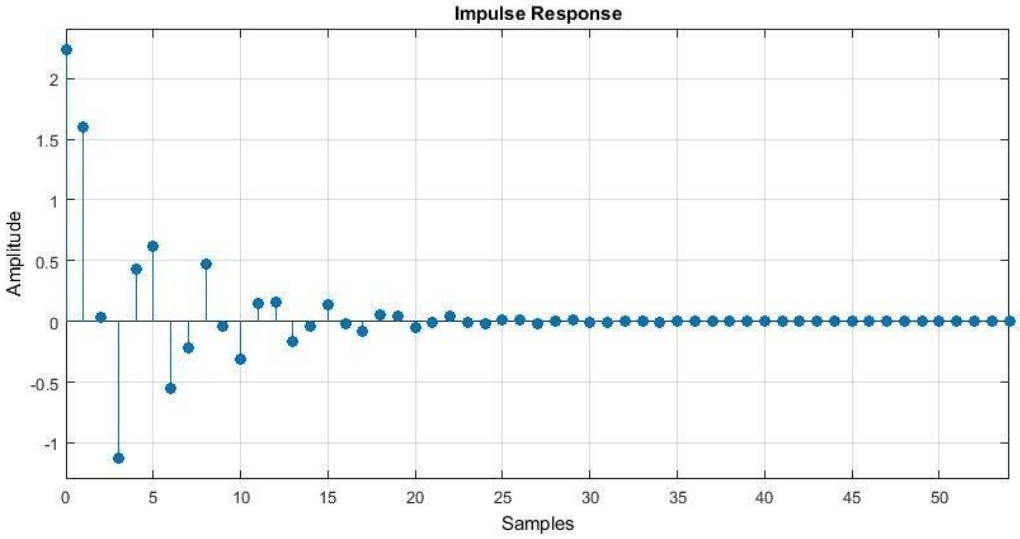
B)PHASE RESPONSE:



C) POLES-ZERO PLOT:



D) IMPULSE RESPONSE:



E) STEP RESPONSE: