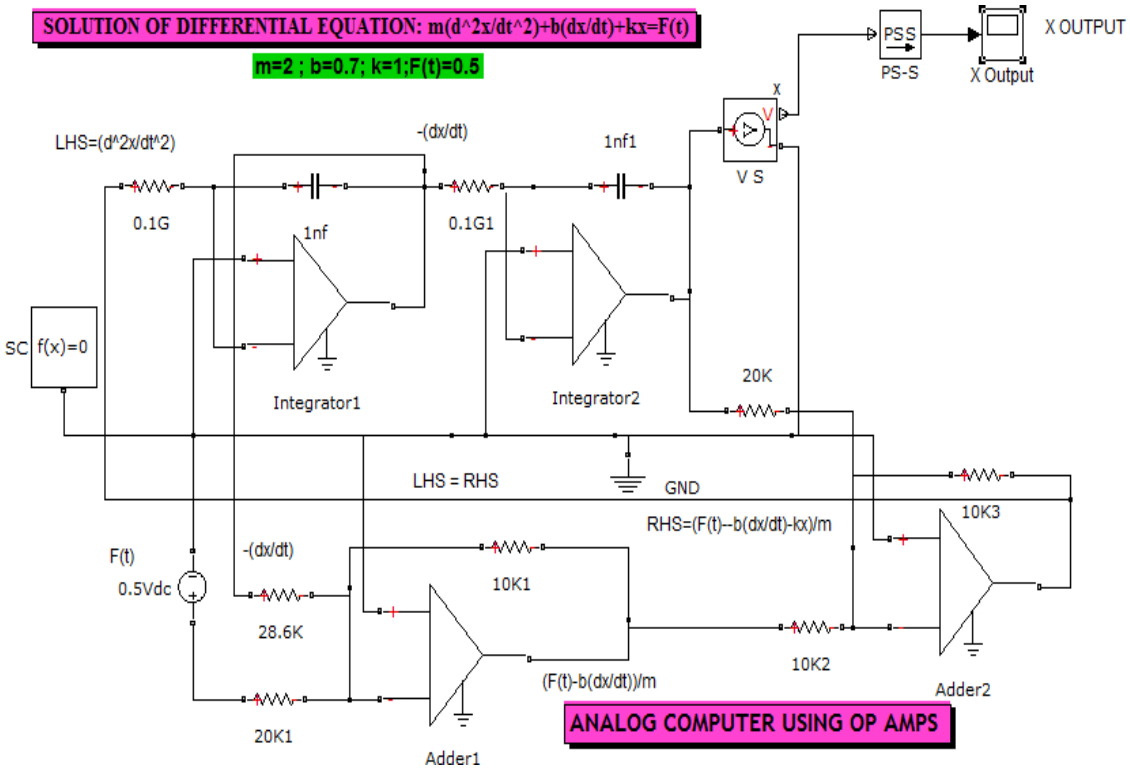
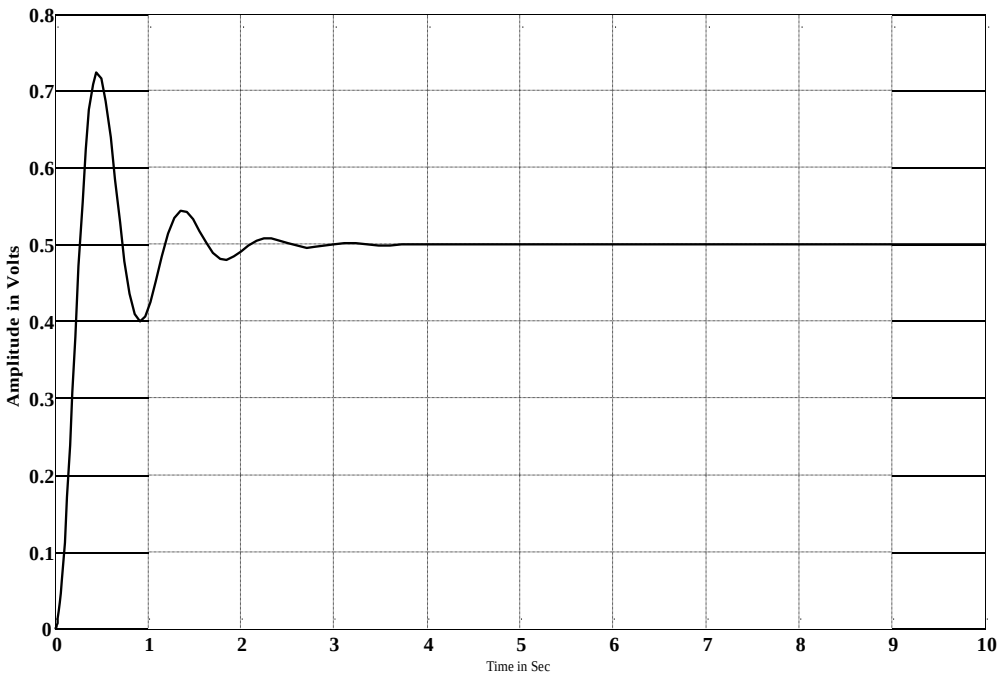
TABLE OF EXPERIMENTS

Sl.No	Date	Name of the Experiments	Page No	Mark Obtained	Staff Signature
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					

SIMULINK BLOCKS



OUTPUT WAVE FORMS



Expt. No: 1	Date:

ANALOG (OP AMP BASED) SIMULATION OF LINEAR DIFFERENTIAL EQUATIONS

AIM

To Study the analog (op amp based) simulation of linear differential equations using Mat lab-Simulink.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

1. To build a SIMULINK model to obtain response of a amplifier, the following procedure is followed:
2. In MATLAB software open a new model in SIMULINK library browser.
3. From the continuous block in the library drag the transfer function block.
4. From the source block in the library drag the step input.
5. From the sink block in the library drag the scope.
6. From the math operations block in the library drag the summing point.
7. From the discrete block in the library drag the amplifier.
8. Connect all to form a system and give unity feedback to the system.
9. For changing the parameters of the blocks connected double click the respective block.
10. Start simulation and observe the results in scope.
11. Compare the simulated and theoretical results.

RESULT

Thus the (op amp based) simulation of linear differential equations has been verified by using Mat-lab Simulink.

PROGRAMME

A.OPEN LOOP RESPONSE (FIRST ORDER SYSTEM)

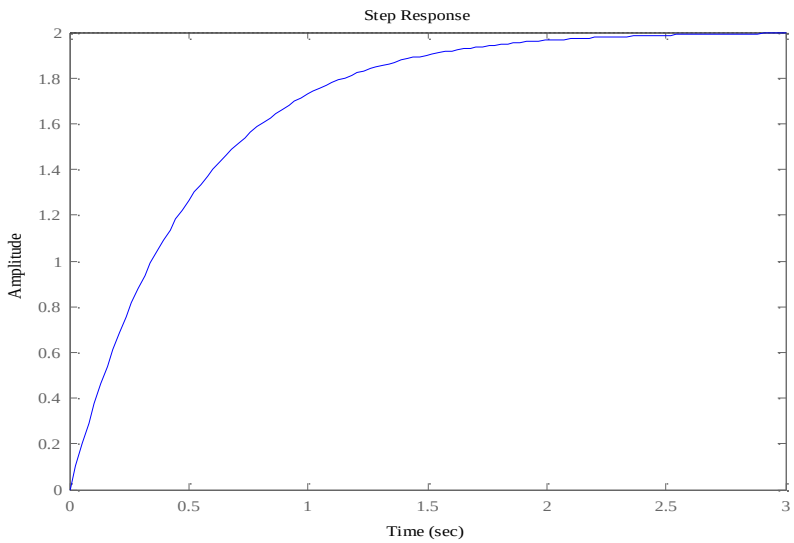
T.F=4/(s+2)

1. Response of system to Step input

```
n=[4];
d=[1 2];
sys=tf(n,d);
step(sys)
```

2. Response of system to Impulse input

```
n=[4];
d=[1 2];
sys=tf(n,d);
step(sys)
impulse(sys)
```



Expt. No:2	Date:

NUMERICAL SIMULATION OF GIVEN NONLINEAR DIFFERENTIAL EQUATIONS.

AIM

To digitally simulate the characteristics of Linear SISO systems using Numerical Simulation of given nonlinear differential equations.

APPARATU REQUIRED

1. A PC with MATLAB package.

THEORY

State Variable approach is a more general mathematical representation of a system, which, along with the output, yields information about the state of the system variables at some predetermined points along the flow of signals. It is a direct time-domain approach, which provides a basis for modern control theory and system optimization. SISO (single input single output) linear systems can be easily defined with transfer function analysis. The transfer function approach can be linked easily with the state variable approach.

The state model of a linear-time invariant system is given by the following equations:

$\dot{X}(t) = A\,X(t) + B\,U(t)$ State equation

$Y(t) = C\,X(t) + D\,U(t)$ Output equation

Where A = n x n system matrix, B = n x m input matrix,

C= p x n output matrix and

D = p x m transmission matrix,

PROGRAM:

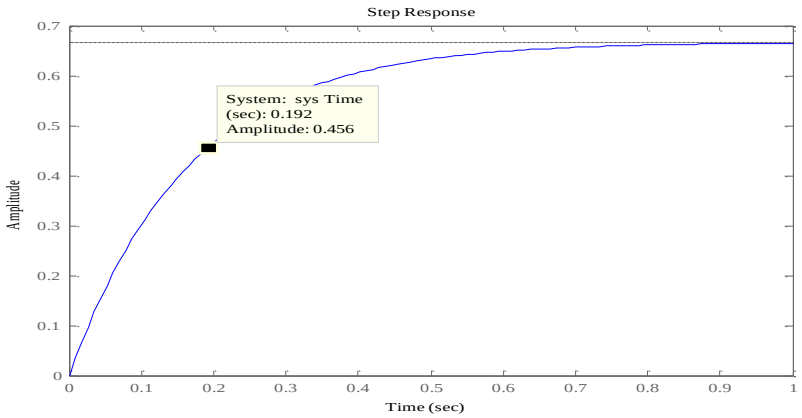
B. Close Loop Response

1. Response of Step input

```
n=[4];
d=[1 2];
sys=tf(n,d);
sys=feedback(sys,1,-1)
step(sys)
```

2. Response of Impulse input

```
n=[4];
d=[1 2];
sys=tf(n,d);
sys=feedback(sys,1,-1)
impulse(sys)
```



Second Order System

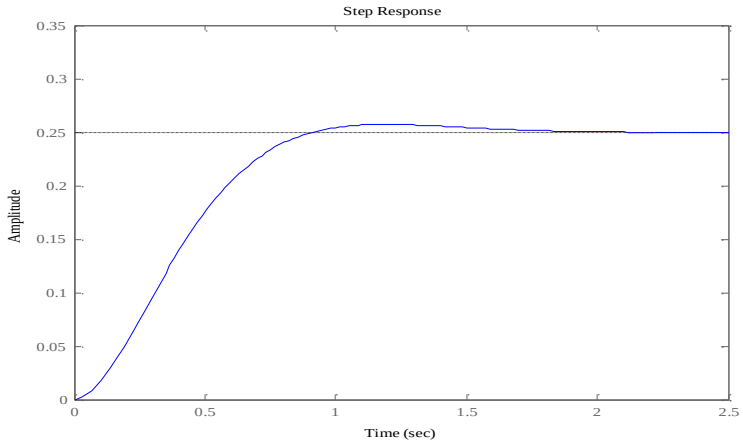
TF= 4/s2+6s+16

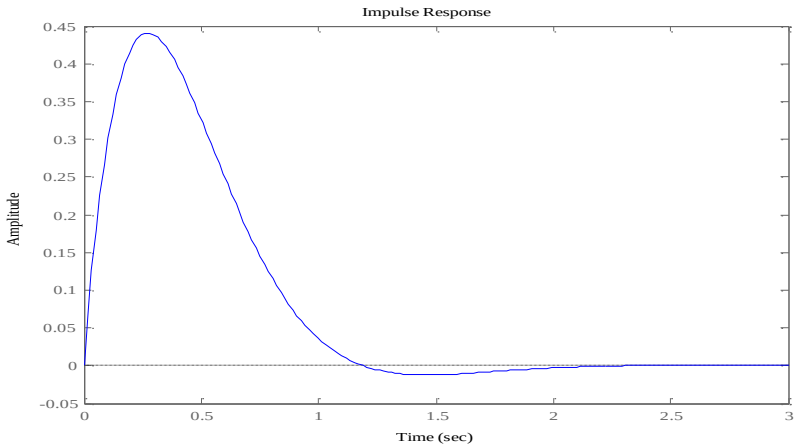
1. Open Loop Response of Step Input

```
n=[4];
d=[1 6 16];
sys=tf(n,d);
step(sys)
```

2. Open Loop Response of impulse Input

```
n=[4];
d=[1 6 16];
sys=tf(n,d);
 impulse(sys)
```





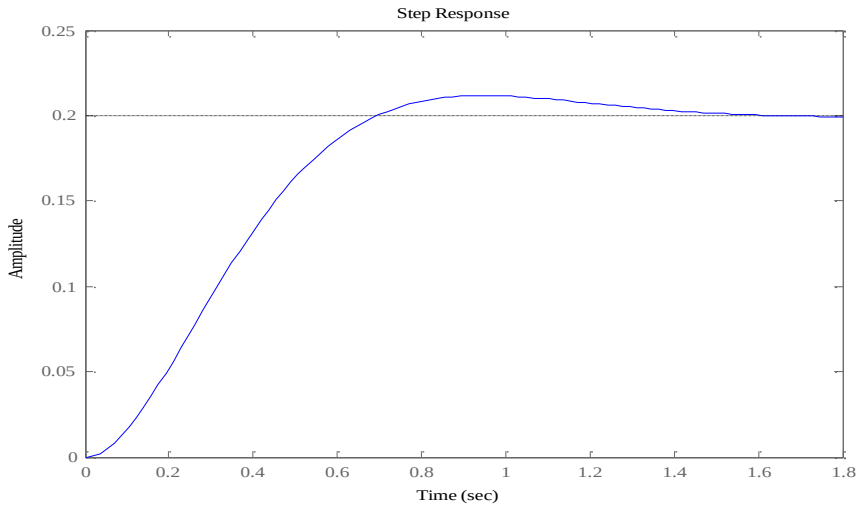
Close Loop Response

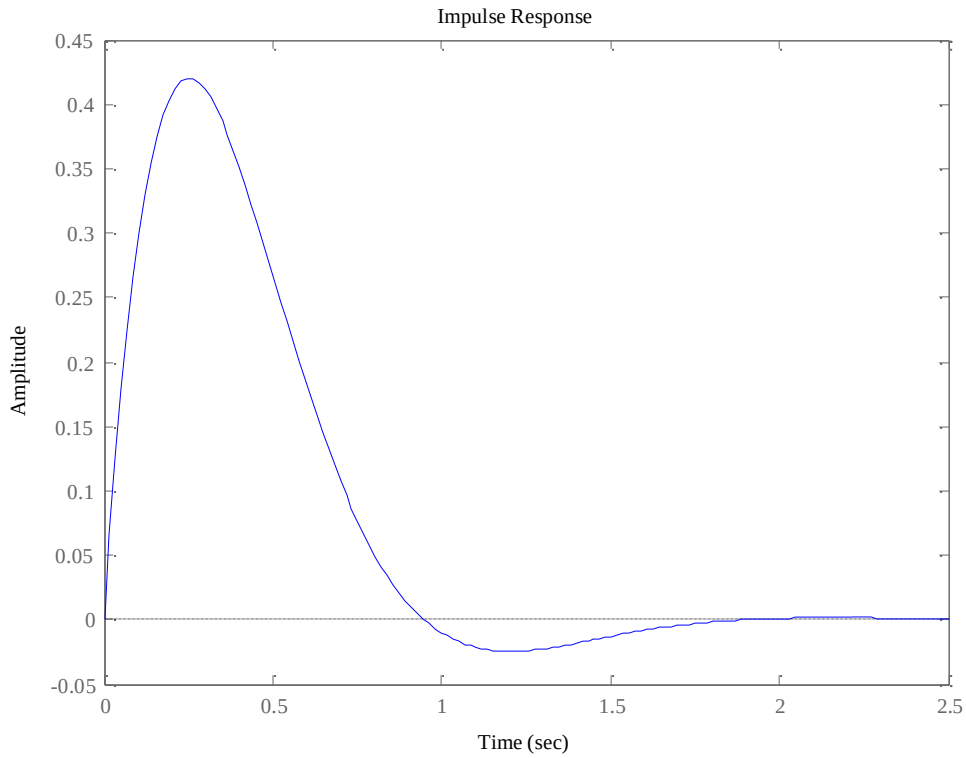
1. Step Input

```
n=[4];
d=[1 6 16];
 sys=tf(n,d);
 sys=feedback(sys,1,-1) ;
 step(sys)
```

2. Impulse Input

```
n=[4];
d=[1 6 16];
sys=tf(n,d);
sys=feedback(sys,1,-1) ;
impulse(sys)
```





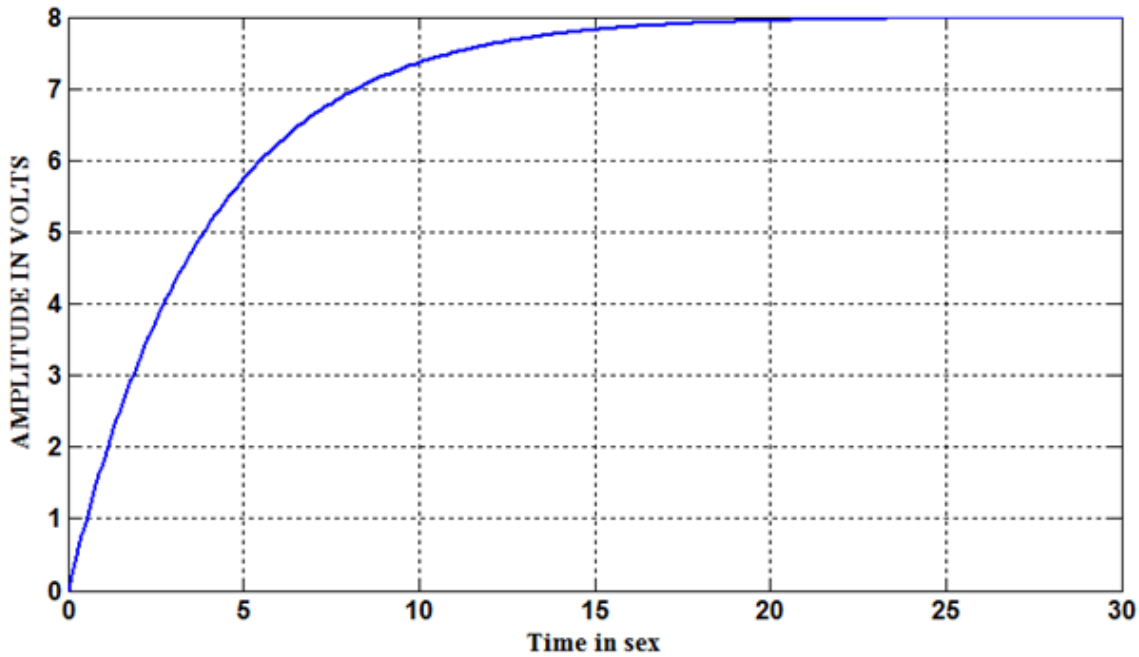
RESULT

Thus the digital simulation of time response characteristics of a first and second order linear system with step and impulse inputs were simulated using MATLAB and outputs are observed for respective inputs.

PROGRAM

```
% Simulation of discrete model
clear,
clc
% Model Parameters
a=0.25;b=2;
% Simulation Parameters
Ts=0.1; %s
Tstop=30; %s
uk=1; % Step Response
x(1)=0;
% Simulation
for k=1:(Tstop/Ts)
x(k+1)=(1-a*Ts).*x(k) + Ts*b*uk;
end
% Plot the Simulation Results
k=0:Ts:Tstop;
plot(k,x)
grid on
```

OUTPUT:



Expt. No: 3	**Date:**

REAL TIME SIMULATION OF DIFFERENTIAL EQUATIONS.

AIM

To check the real time simulation for the given differential equation $\dot{x} = -ax + bu$ using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

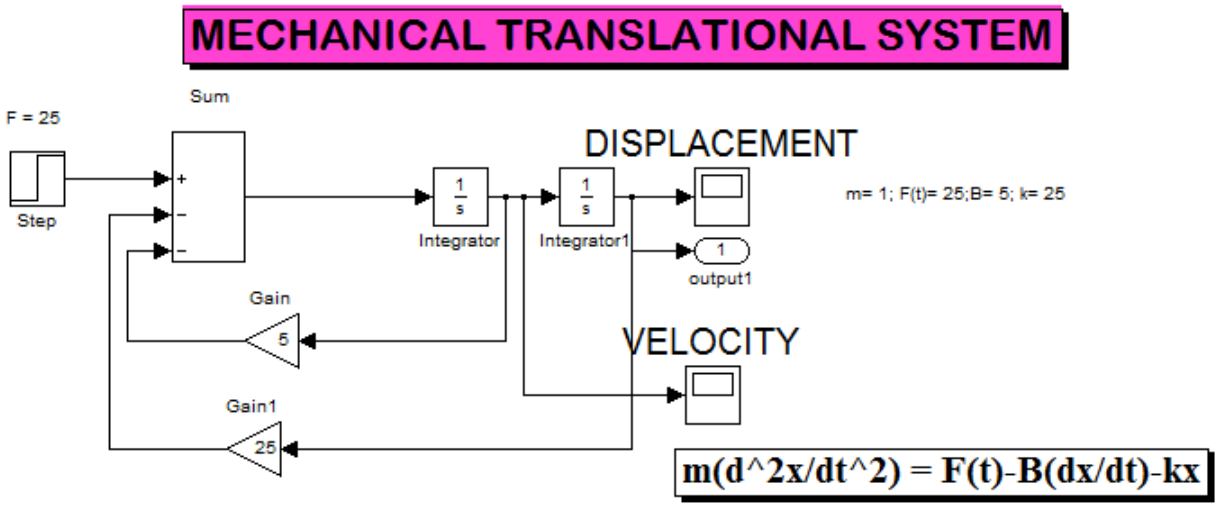
PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

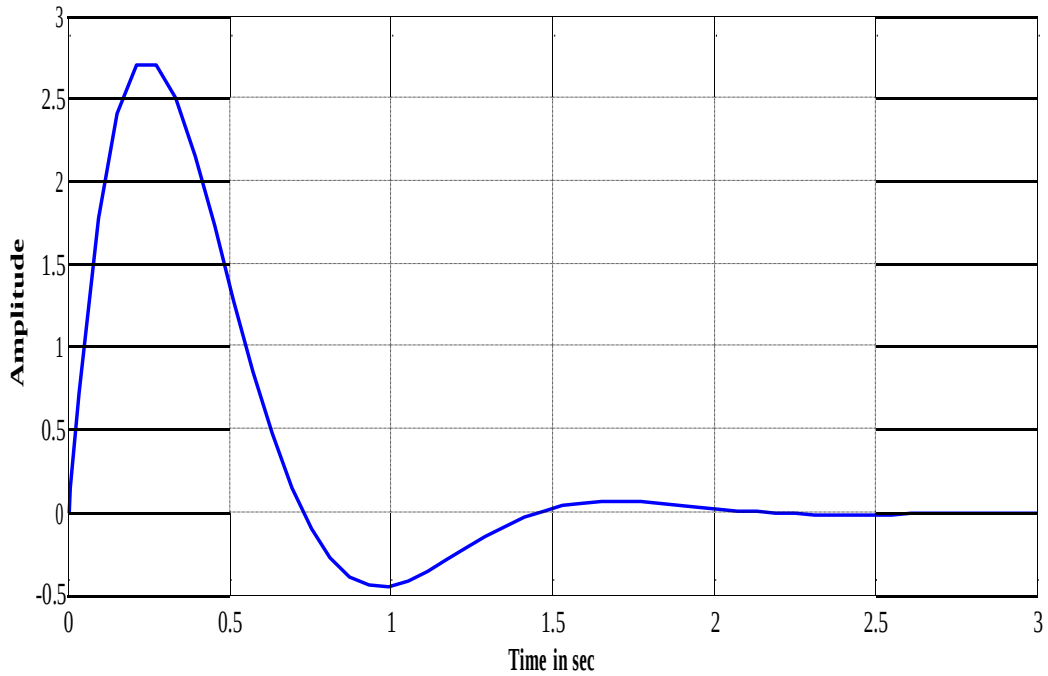
RESULT

Thus the real time simulation for the given differential equation $\dot{x} = -ax + bu$ using MATLAB Software was executed.

SIMULINK MODEL



OUTPUT



Expt. No: 4	Date:

MATHEMATICAL MODELING AND SIMULATION OF PHYSICAL SYSTEMS IN AT LEAST TWO FIELDS MECHANICAL, ELECTRICAL AND CHEMICAL PROCESS

AIM

To design mathematical modelling and simulation of physical systems in at least two fields: mechanical, electrical, and chemical processes, using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

RESULT

Thus the mathematical modelling and simulation of physical systems in at least two fields: mechanical, electrical, and chemical processes, using MATLAB software, were done.

PROGRAM:

```
function[Weights_Model] = system_identification()

%clear all;
%clc;
% Direct modeling of Electrical system of the plant using LMS algorithm
%Plant Parameters
Weights_Plant=[0.26 0.93 0.26] ; % Transfer function coefficients
N=input('Enter the no of the iterations/input samples: '); %input number of
samples.
Input_Plant= rand(1,N); % input to the plant with mean zero and input to
model is also same.

% Model Parameters
Weights_Model=[0 0 0];
Input_Model=Input_Plant;
Learning_Para=input('Enter the Learning Parameter value:'); % Learning rate
of the model

%Signal Power to noise power
snr=input('Enter the snr value : ');

%Signal Power and Noise Power
sp=var(Input_Plant);
np =(sp)*power(10,-(snr/10));

%Noise to be added
Noise=sqrt(np)*(rand(1,N)-0.5);



%-------------------Intialization-----------------%
% Plant Output. Plant_Output(1)=Input_Plant(1)*Weights_Plant(1);

Plant_Output(2)=Input_Plant(2)*Weights_Plant(1)+Input_Plant(1)*Weights_Plan
t(2);

    %Model Output
    Model_Output(1)=Input_Model(1)*Weights_Model(1);

Model_Output(2)=Input_Model(2)*Weights_Model(1)+Input_Model(1)*Weights_Mode
l(2);

    % Plant output with added noise
    Plant_Output(1)=Plant_Output(1)+Noise(1);
    Plant_Output(2)=Plant_Output(2)+Noise(2);

    % Error
    error(1)=Plant_Output(1)-Model_Output(1);
    error(2)=Plant_Output(2)-Model_Output(2);

     %Weight Values

Weights_Model(1)=Weights_Model(1)+(2*Learning_Para*Input_Model(1)*error(1))
;
    Weights_Model(1:2)=Weights_Model(1:2)+(2*Learning_Para*Input_Model(2:-
1:1)*error(2));
```

Expt. No: 5	Date:

SYSTEM IDENTIFICATION THROUGH PROCESS REACTION CURVE

AIM

To design the system identification through the process reaction curve by using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

```
%------Applying Adaptive LMS algorithm--------%
for i=3:1:N

    %Outputs
    Plant_Output(i)=Input_Plant(i)*Weights_Plant(1)+Input_Plant(i-
1)*Weights_Plant(2)+Input_Plant(i-2)*Weights_Plant(3);
    Model_Output(i)=Input_Model(i)*Weights_Model(1)+Input_Model(i-
1)*Weights_Model(2)+Input_Model(i-2)*Weights_Model(3);

    %White gaussian noise signal added to the output of the plant.
    Plant_Output(i)=Plant_Output(i)+Noise(i);
    error(i)=Plant_Output(i)-Model_Output(i);

    %Weight Update using LMS algorithm
    Weights_Model=Weights_Model + (2*Learning_Para*Input_Model(i:-1:i-
2)*error(i));
end

%Plots of error square and log of normalized error square

t=1:1:N;
error_square=power(error,2);
Max_error_square=max(error_square);

Normalized_error_Square=error_square./Max_error_square;

plot(t,error_square);
legend('error square');
title('Plot of error square Vs no of iterations');
xlabel('iterations');
ylabel('error square')

figure,
plot(t,log(Normalized_error_Square));
legend('Normalized error square log');
title('Plot of Normalized error square log Vs no of iterations');
xlabel('iterations');
ylabel('Normalized error square log')

%-----------------------------Testing mode-----------------%

s=30;
Input_test=rand(1,s);


for j= 3:1:s


    Plant_Out_test(1)=Input_test(1)*Weights_Plant(1);

Plant_Out_test(2)=Input_test(2)*Weights_Plant(1)+Input_test(1)*Weights_Plan
t(2);
    Plant_Out_test(j)=Input_test(j)*Weights_Plant(1)+Input_test(j-
1)*Weights_Plant(2)+Input_test(j-2)*Weights_Plant(3);

    Out_Model_test(1)=Input_test(1)*Weights_Model(1);
```

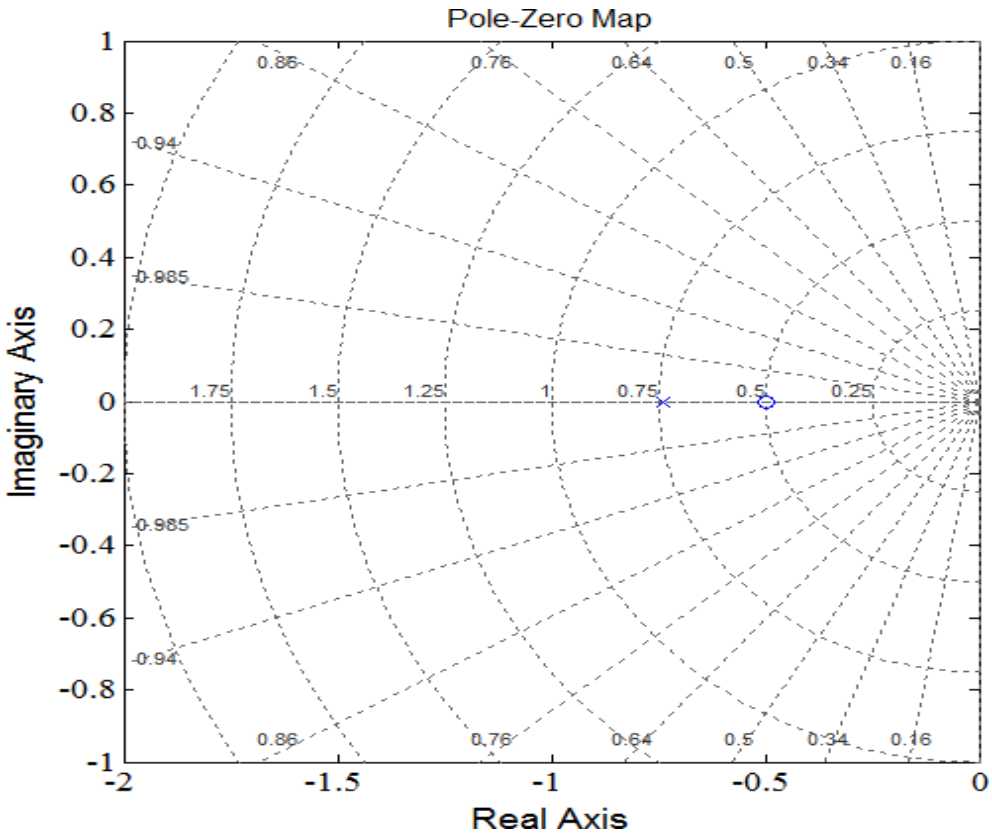
```
Out_Model_test(2)=Input_test(2)*Weights_Model(1)+Input_test(1)*Weights_Mode
l(2);
    Out_Model_test(j)= Input_test(j)*Weights_Model(1)+Input_test(j-
1)*Weights_Model(2)+Input_test(j-2)*Weights_Model(3);

    error_test(1)= Plant_Out_test(1)-Out_Model_test(1);
    error_test(2)= Plant_Out_test(2)-Out_Model_test(2);
    error_test(j)= Plant_Out_test(j)-Out_Model_test(j);

end
p=1:1:s;
figure,
plot(p,Plant_Out_test,'-ro',p,Out_Model_test,'-.b');
legend('Plant Out test','Out Model test');
title('Comparision of outputs of Plant and Model during testing');
xlabel('iterations');
ylabel('Output')

square_error= error_test.^2;
SSE=sum(square_error);


end
```

RESULT

Thus the design of the system identification through the process reaction curve using MATLAB Software was executed.

PROGRAM

```
s = tf('s');
G = (2*s+1)/(s^2+3*s+2);
k = 0.7;
T = feedback(G*k,1);
pzmap(T)
grid, axis([-2 0 -1 1])
```



Expt. No: 6	Date:

STABILITY ANALYSIS USING POLE ZERO MAPS AND ROUTH HURWITZ CRITERION IN SIMULATION PLATFORM

A. STABILITY ANALYSIS USING POLE ZERO MAPS

AIM

To check the stability analysis using pole zero maps for the given system or transfer function using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

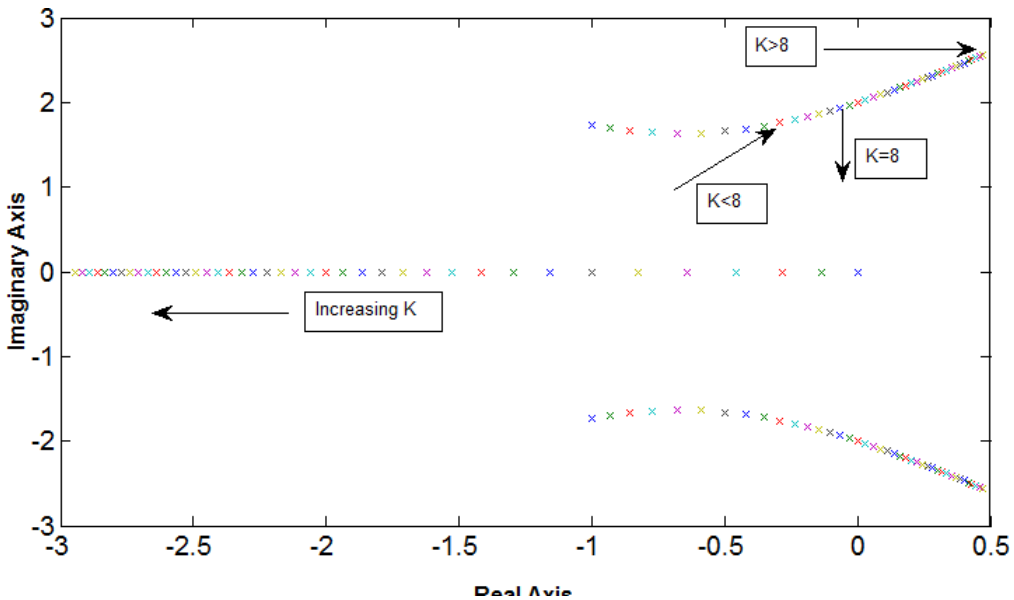
- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

RESULT

Thus the designs for stability analysis using pole zero maps for the given system or transfer function using MATLAB were executed.

PROGRAM

```
% This script computes the roots of the characteristic
% equation D(s) = s^3 + 2 s^2 + 4 s + K for 0 < K < 20
K=[0:0.5:20];                % 0 < K < 20
for i=1:length(K);           % for loop
q=[1 2 4 K(i)];
p(:,i)=roots(q);
end
figure(1)
plot(real(p),imag(p),'x'),
grid xlabel('Real axis')
ylabel('lmaginary axis')
gtext('K < 8')               % Writing text on graphic
gtext('K = 8')
gtext('K > 8')
num=[1];
den=[1 2 4 9];
sysg=tf(num,den);
sys=feedback(sysg,[1]);
pole(sys)                    % poles of closed loop system
figure(2)
step(sys);
```



Expt. No: 6	Date:

STABILITY ANALYSIS USING POLE ZERO MAPS AND ROUTH HURWITZ CRITERION IN SIMULATION PLATFORM

B.STABILITY ANALYSIS USING ROUTH HURWITZ CRITERION IN SIMULATION PLATFORM

AIM

To check the stability analysis is using Routh Hurwitz Criterion for the given system or transfer function using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

RESULT

Thus the stability analysis is using Routh Hurwitz Criterion for the given system or transfer function using MATLAB Software was executed.

PROGRAM

```
% Characteristic polynomial
% C(s)=s^3+2s^2+10s^2+15
% Input coeff=Vector of coefficients of the C(s);e.g.,[1 2 10 15]
clc;
clear;
coeff=input('Enter the coefficients:')
L=length(coeff);
if (rem(L,2)==0)
    Routh_array=zeros(L,L/2);
    for i=1:L/2
        Routh_array(1,i)=coeff(1,2*i-1);
        Routh_array(2,i)=coeff(1,2*i);
    end
else
    Routh_array=zeros(L,(L+1)/2);
    for i=1:(L+1)/2
      Routh_array(1,i)=coeff(1,2*i-1);
      if i==(L+1)/2
          break;
      end
      Routh_array(2,i)=coeff(1,2*i);
    end
end
for i=3:size(Routh_array,1)
    if Routh_array(i-1,1)==0
        Routh_array(i-1,1)=0.001;
    end
    for J=1:size(Routh_array,2)-1
        Routh_array(i,J)=(-1/Routh_array(i-1,1))*det([Routh_array(i-2,1)...
            Routh_array(i-2,J+1);Routh_array(i-1,1) Routh_array(i-1,J+1)]);
    end
end
Routh_array
S=sign(Routh_array);
count=0;
for i=1:L
    if S(i,1)==1
        count=count+1;
    end
end
if count==L
    disp('The system is stable')
else
    disp('The system is unstable')
end
% verify
fprintf('\n');
disp('verification:')
Roots=roots(coeff);
disp('poles:')
disp(Roots)
```

Expt. No: 6	Date:

STABILITY ANALYSIS USING POLE ZERO MAPS AND ROUTH HURWITZ CRITERION IN SIMULATION PLATFORM

C.STABILITY ANALYSIS USING ROUTH HURWITZ CRITERION IN SIMULATION PLATFORM

AIM

To check the stability analysis is using Routh Hurwitz Criterion for the given system or transfer function using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

OUTPUT:

```
Enter the coefficients: [1 1 2 2 11 10]

coeff =

     1     1     2     2    11    10


Routh_array =

    1.0000    2.0000   11.0000
    1.0000    2.0000   10.0000
    0.0010    1.0000         0
 -998.0000   10.0000         0
    1.0000         0         0
   10.0000         0         0

The system is unstable

verification: poles:
   1.0589 + 1.4691i
   1.0589 - 1.4691i
  -1.0961 + 1.4467i
  -1.0961 - 1.4467i
  -0.9256
```

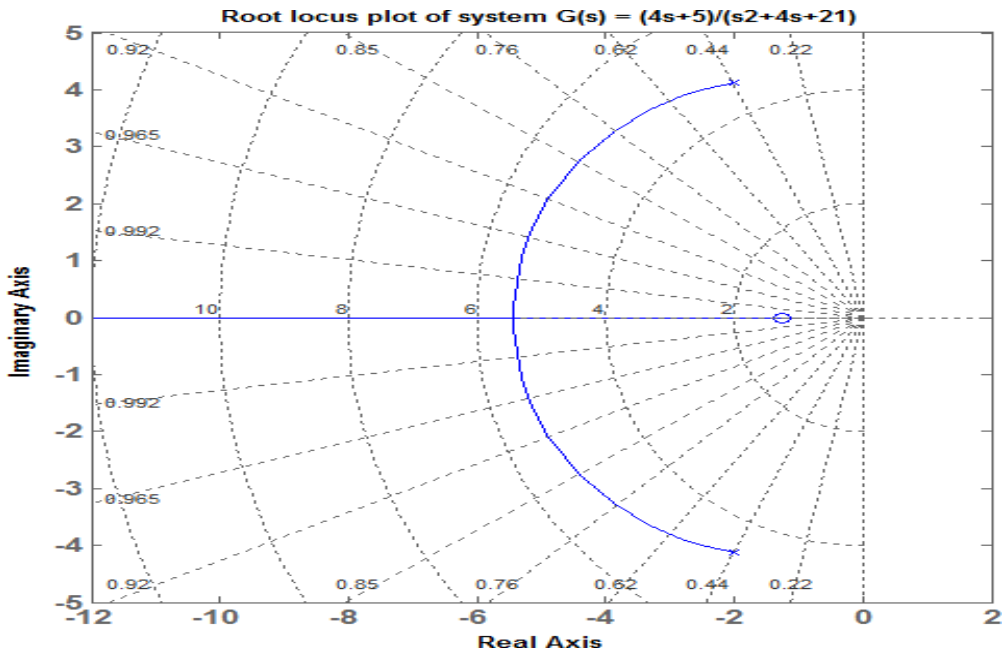
RESULT

Thus the stability analysis is using Routh Hurwitz Criterion for the given system or transfer function using MATLAB Software was executed.

PROGRAM

```
num=[0 4 5];
den=[1 4 21];
sys1=tf(num, den);
rlocus(sys1)
grid
title('Root locus plot of system G(s) = (4s+5)/(s2+4s+21)')
```

OUTPUT:



Expt. No: 7	Date:

ROOT LOCUS BASED ANALYSIS IN SIMULATION PLATFORM

AIM

To Check the stability analysis of the given system or transfer function of rootlocus using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

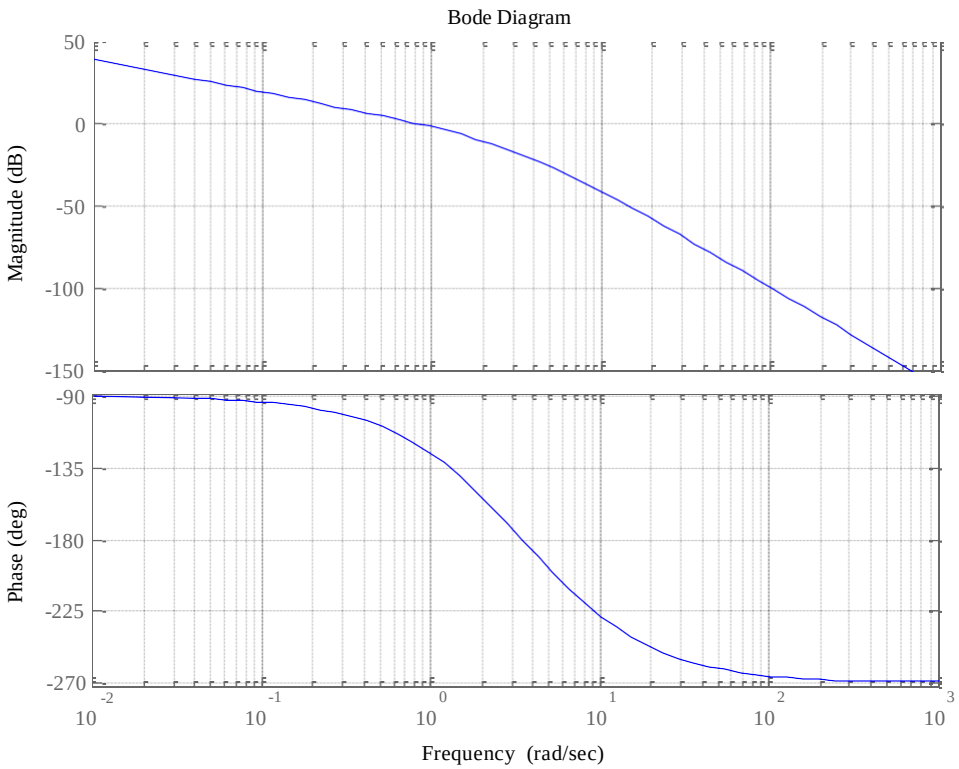
RESULT

Thus the designs of root locus for the given transfer function using MATLAB Software was executed.

PROGRAM

%Bode Plot for the Transfer Function=$10/(s^3+8s^2+12s)$

```
nu=[10];
de=[1 8 12 0];
sys=tf(nu,de);
bode(sys);
grid on;
```



Expt. No:8	Date:

DETERMINATION OF TRANSFER FUNCTION OF A PHYSICAL SYSTEM USING FREQUENCY RESPONSE AND BODE'S ASYMPTOTES.

AIM

To Design the bode plot for the given system and also determine the gain and phase margin using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

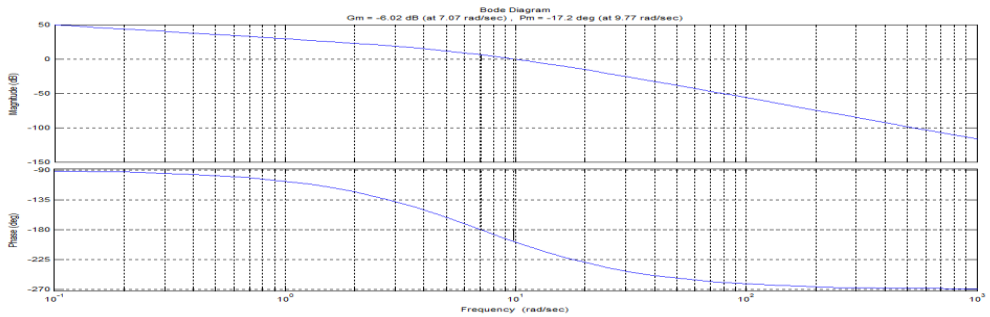
RESULT

Thus the design of bode plot for the given system using MATLAB Software was executed.

PROGRAM

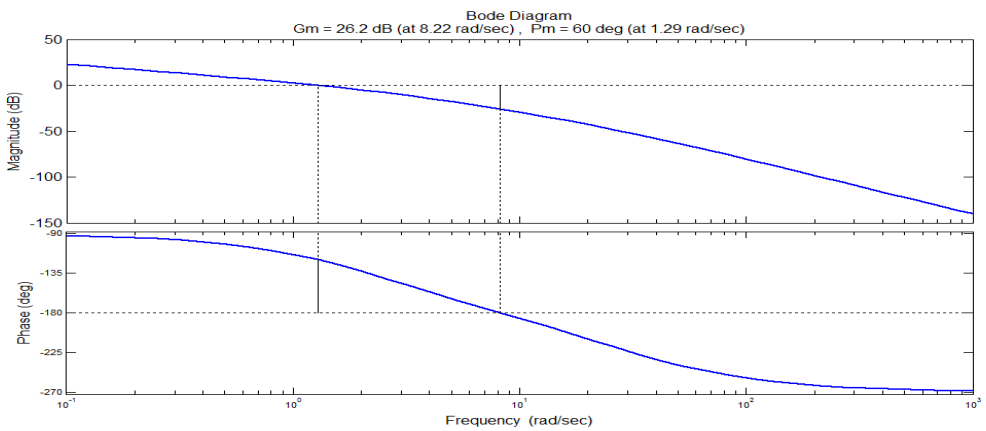
1. Lag Compensator

```
K=9600
d1=conv([1 4 0],[1 80]) % used to formulate the polynomial
G=tf(K,d1) % generate the transfer function.
margin(G)
pmreq=input('Enter Required Phase Margin') % enter 33 as given in question
pmreq=pmreq+5
phgcm=pmreq-180
wgcm=input('Enter new gain cross over frequency') % from the first bode
plot use the mouse pointer and locate wgcm corresponding to phgcm.
[Beta,p]=bode(G,wgcm)
T=10/wgcm
Zc=1/T
Pc=1/(Beta*T)
Gc=tf([1 Zc],[1 Pc])
sys=Gc*G/Beta
margin(sys)
```



2..LEAD COMPENSATOR

```
clear variables;
% plant G parameters
s = tf('s');
a = 2.5; b = 27;
%% a) Gain K for PO=10
PO = 10; % percentage overshoot
zeta = log(100/PO)/sqrt(pi^2+ (log(100/PO))^2 ); % damping ratio
PM_d = round(100*zeta)+1; % PM desired at the nearest round value
h = tand(180-90-PM_d);
omega_c = roots([1 (a+b)/h -a*b]); % new omega_c
K = omega_c(2)*sqrt(omega_c(2)^2+a^2)*sqrt(omega_c(2)^2+b^2); % new gain K
G = K/( s*(s+a)*(s+b) ); % Plant G
figure;
margin(G);
```



Expt. No:9	Date:

DESIGN OF LAG, LEAD COMPENSATORS AND EVALUATION OF CLOSED LOOP PERFORMANCE

AIM

To design the lag, lead compensators and evaluation of closed loop Performance using MATLAB Software.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

RESULT

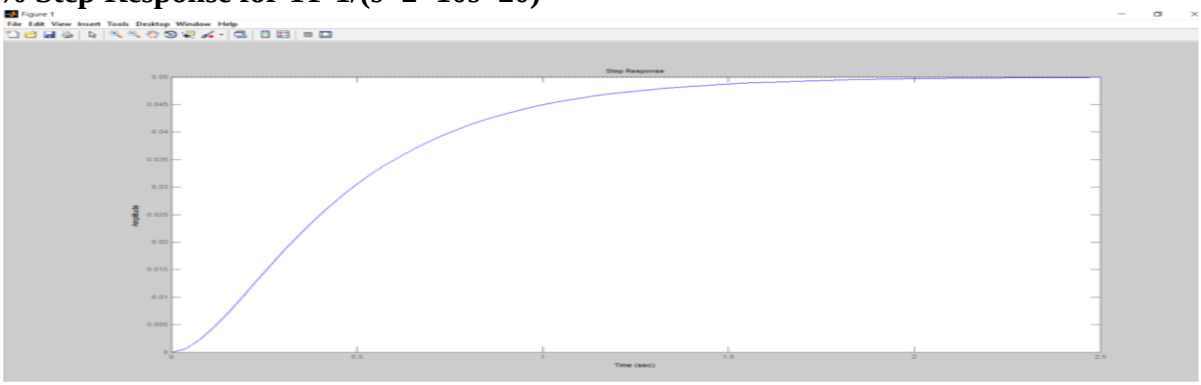
Thus the design of lag, lead compensators and evaluation of closed loop Performance using MATLAB Software was executed.

PROGRAMME

```
% Step Response for TF 1/(s^2+10s+20)
num=[1];
den=[1 10 20];
figure(1);
step(num,den)
```

OUTPUT

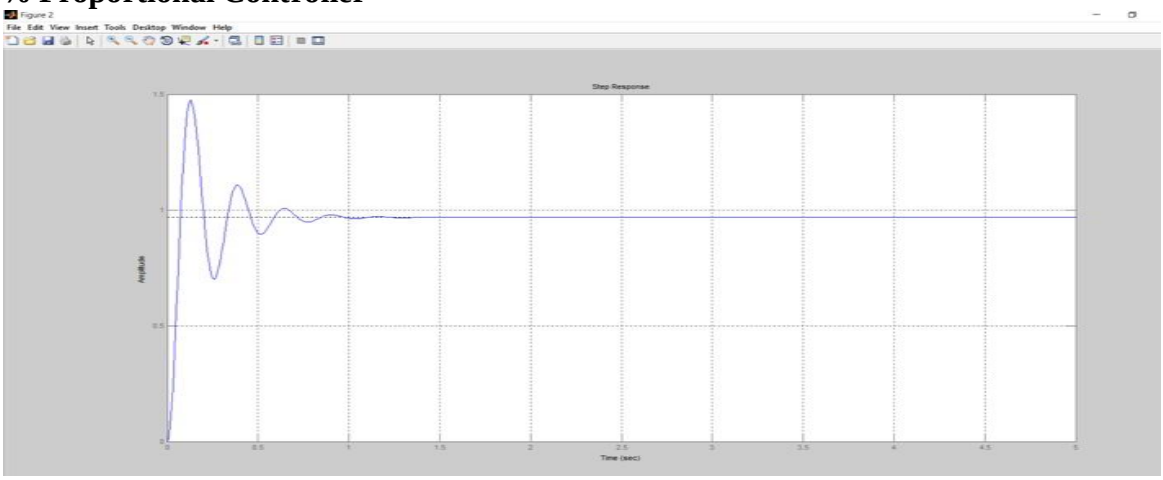
% Step Response for TF 1/(s^2+10s+20)



```
% Proportional Controller
 Kp=600;
num1=[Kp];
den1=[1 10 20+Kp];
t=0:0.01:2;
figure(2);
step(num1,den1,t)
grid;
```

OUTPUT

% Proportional Controller



Expt. No: 10	Date:

DESIGN OF PID CONTROLLERS AND EVALUATION OF CLOSED LOOP PERFORMANCE

AIM

To Study the effect of P, PI, PID controllers using Mat lab-Simulink.

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

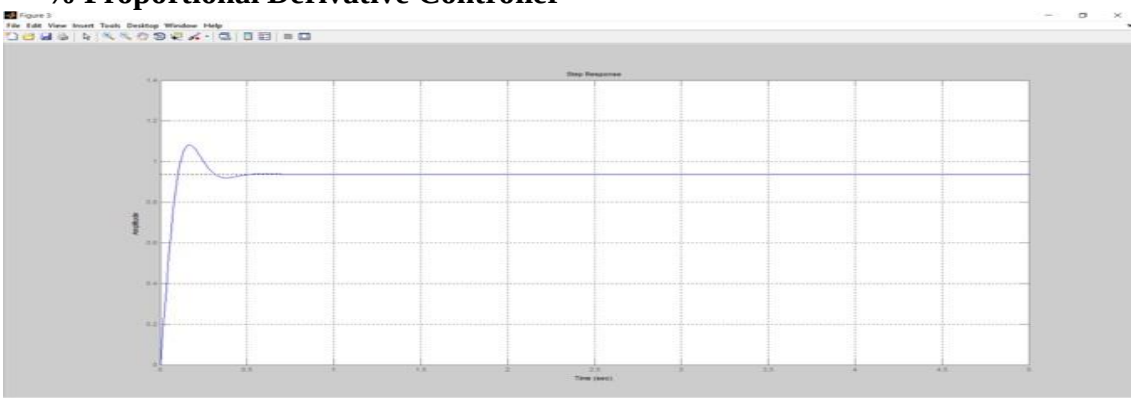
12. To build a SIMULINK model to obtain response of a P, PI, PID Controllers, the following procedure is followed:
13. In MATLAB software open a new model in SIMULINK library browser.
14. From the continuous block in the library drag the transfer function block.
15. From the source block in the library drag the step input.
16. From the sink block in the library drag the scope.
17. From the math operations block in the library drag the summing point.
18. From the discrete block in the library drag the PID controller.
19. Connect all to form a system and give unity feedback to the system.
20. For changing the parameters of the blocks connected double click the respective block.
21. Start simulation and observe the results in scope.
22. Compare the simulated and theoretical results.

% Proportional Derivative Controller

```
Kp=300;
Kd=10;
num2=[Kd  Kp];
den2=[1 10+Kd 20+Kp];
t=0:0.01:2;
figure(3);
step(num2,den2,t)
grid;
```

OUTPUT:

% Proportional Derivative Controller

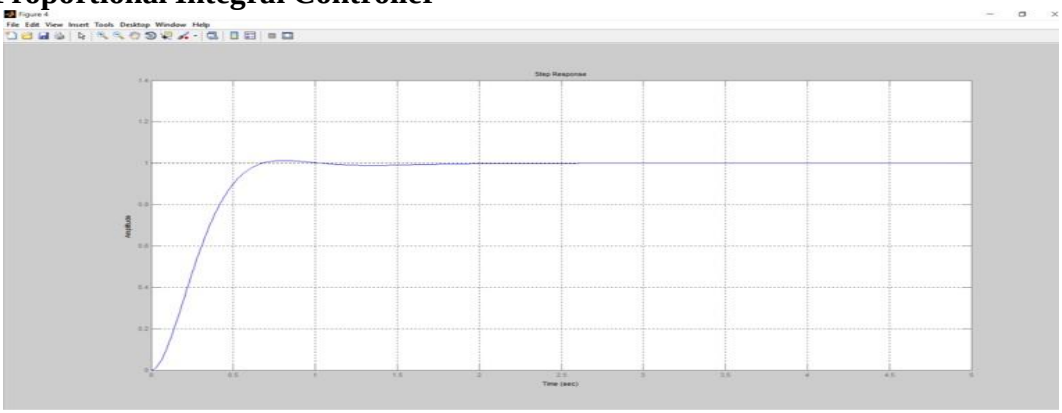


% Proportional Integral Controller

```
Kp1=30;
Ki=70;
num3=[Kp1 Ki];
den3=[1 10 20+Kp1 Ki];
t=0:0.01:2;
figure(4);
step(num3,den3,t)
grid;
```

OUTPUT:

% Proportional Integral Controller

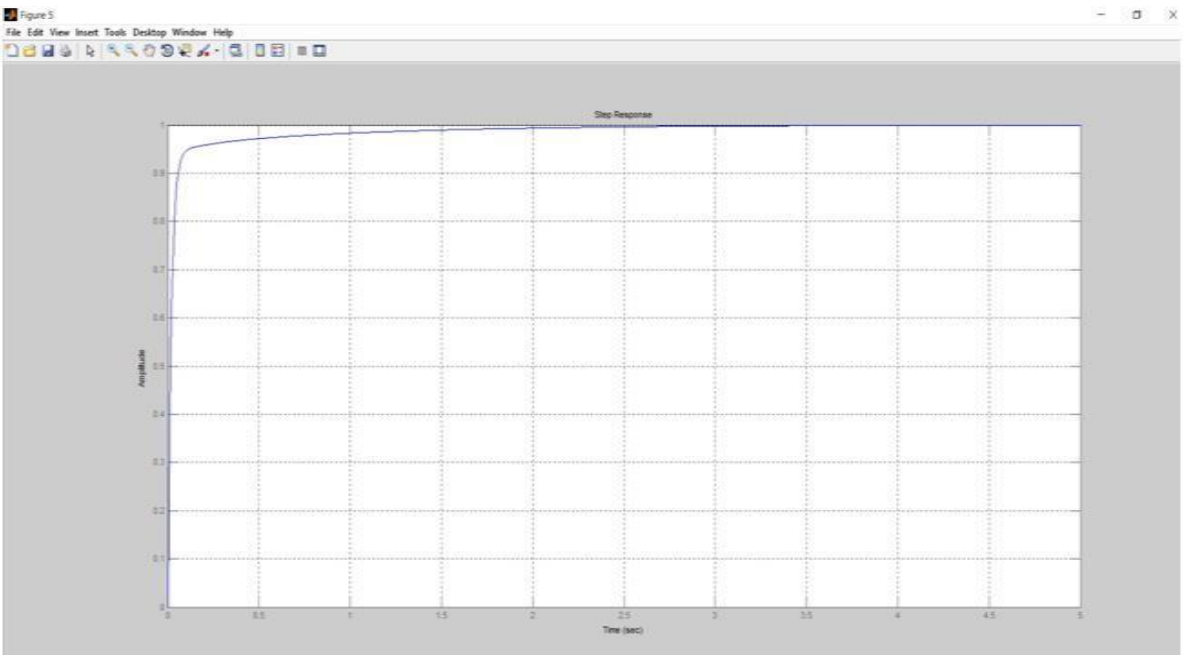


%Proportional Integral Derivative Controller

```
Kp2=350;
Kd2=50;
Ki2=300;
num4=[Kd2 Kp2  Ki2];
den4=[1 10+Kd2 20+Kp2 Ki2];
t=0:0.01:2;
figure(5);
step(num4,den4,t)
grid;
```

OUTPUT

%Proportional Integral Derivative Controller



RESULT

Thus the effect of P, PI, PD, and PID controller has been verified by using Matlab coding.

PROGRAM:

```
%% Effect of sampling and verification of sampling theorem
clear
close all
clc

%% Problem
Vm1=20;                     % Voltage magnitude for v1
Vm2=10;                     % Voltage magnitude for v2
DOB=27;                     % Date of birth
MOB=11;                     % Month for birth
f1=MOB*10;                  % Frequency for v1
f2=DOB*10;                  % Frequency for v2
w1=2*pi*f1;                 % Frequency for v1 (rad/sec)
w2=2*pi*f2;                 % Frequency for v2 (rad/sec)
t=0:0.00005:0.04;           % Time for plotting signal
v1=Vm1*sin(w1*t);           % v1
v2=Vm2*sin(w2*t);           % v2
v=v1+v2;                    % Combination

figure(1)
subplot(2,2,1)
plot(t,v1)
hold on
plot(t,v2)
grid on
xlabel('Time (s)')
ylabel('v_1 and v_2')
title('Voltage waveforms')
legend('v_1','v_2')

subplot(2,2,2)
plot(t,v)
grid on
xlabel('Time (s)')
ylabel('v')
title('Combined waveform')

%% Defining sampling periods
fm=max(f1,f2);          % Maximum frequency component
fs1=2*fm; Ts1=1/fs1;    % Just same
fs2=3*fm; Ts2=1/fs2;    % fs>2fm (satisfying sampling theorem)
fs3=1*fm; Ts3=1/fs3;    % fs<2fm (not satisfying sampling theorem)

%% fs>2fm (satisfying sampling theorem)
ts2=0:Ts2:0.04;
v1s2=Vm1*sin(w1*ts2);
v2s2=Vm2*sin(w2*ts2);
vs2=v1s2+v2s2;
subplot(2,2,3)
stem(ts2,vs2)
xlabel('Time (s)')
ylabel('v')
title('Signal sampled with f_s>2f_m')
```

Expt. No: 11	Date:

DISCRETIZATION OF CONTINUOUS SYSTEM AND EFFECT OF SAMPLING

AIM

1. To plot a continuous-time signal using MATLAB.

2. To sample the signal at a faster, lower, and just right (satisfying criteria of sampling theorem) sampling rates.

APPARATUS REQUIRED

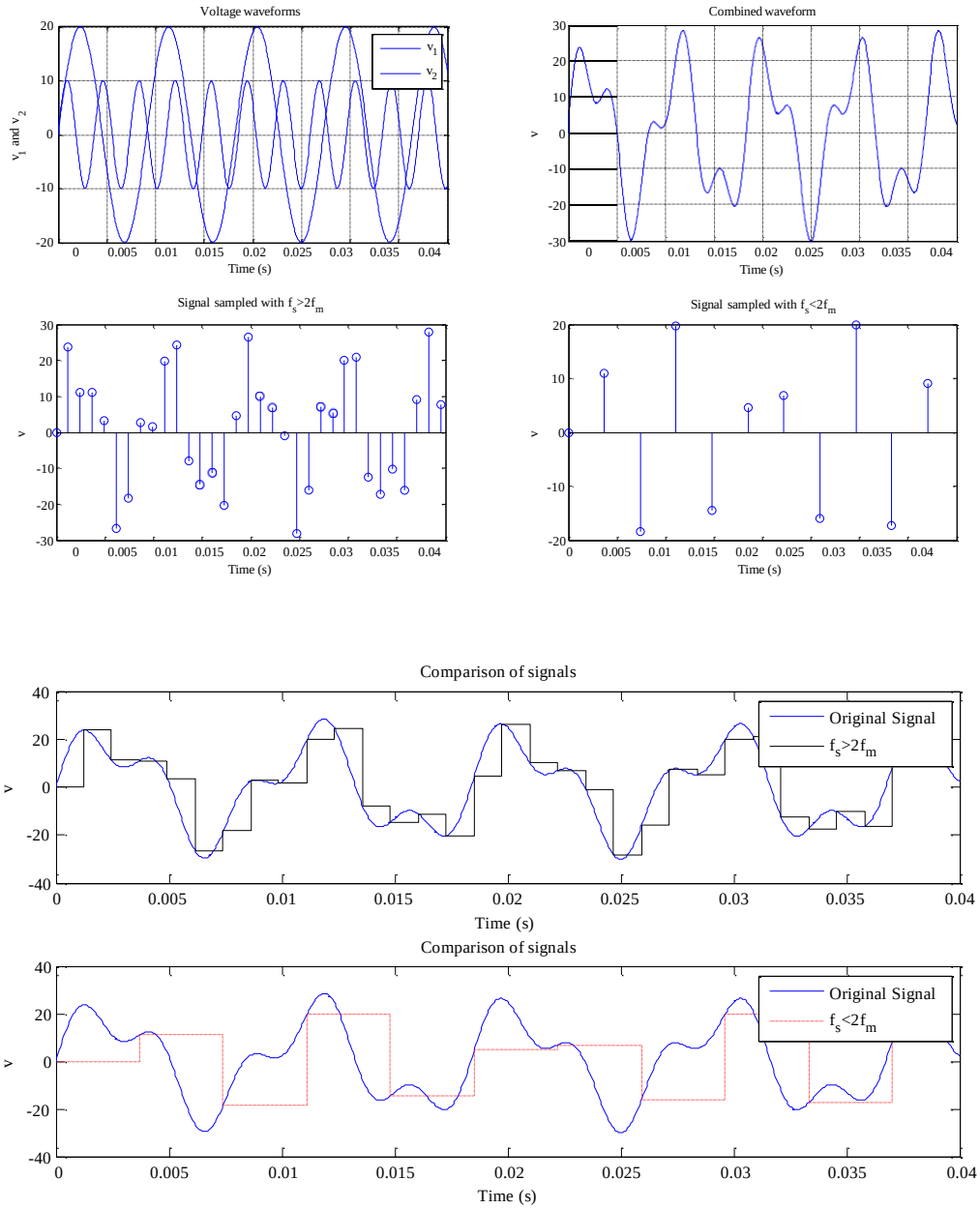
- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

```
%% fs<2fm (not satisfying sampling theorem)
ts3=0:Ts3:0.04;
v1s3=Vm1*sin(w1*ts3); v2s3=Vm2*sin(w2*ts3);
vs3=v1s3+v2s3;
subplot(2,2,4)
stem(ts3,vs3)
xlabel('Time (s)')
ylabel('v')
title('Signal sampled with f_s<2f_m')

%% Comparison of signals (by plotting zoh type signals)
figure(2)
subplot(2,1,1)
plot(t,v,'b')
hold on
stairs(ts2,vs2,'k')
xlabel('Time (s)')
ylabel('v')
title('Comparison of signals')
legend('Original Signal','f_s>2f_m')

subplot(2,1,2)
plot(t,v,'b')
hold on
stairs(ts3,vs3,'--r')
xlabel('Time (s)')
ylabel('v')
title('Comparison of signals')
legend('Original Signal', 'f_s<2f_m')
```

OUTPUT



RESULT

Thus the continuous-time signal and sample the signal at a faster, lower, and just right (satisfying criteria of sampling theorem) sampling rates were done.

PROGRAM:

```
clc;
A =[1 2 1;-1 -4 -3; -1 2 3];
B = [1; 4; 6];
C = [1 1 2];
D=0;
disp('Rank of the Matrix')
Rankc=rank([B A*B A^2*B]) % To check the controllability
Ranko= rank([C' A'*C' A'^2*C']) % To check the observability
Rankoc= rank([C*B C*A*B C*A^2*B]) % To check the output Controllability
```

OUTPUT

Rank of the Matrix
Rankc = 3
Ranko = 3
Rankoc =1

From the above, the system is completely state controllable and observable. The system is not output controllable since the rank of the matrix is not three.

Alternate Program:

```
clc;
A =[1 2 1;-1 -4 -3; -1 2 3];
B = [1; 4; 6];
C = [1 1 2];
D=0;
disp('Rank of the Matrix')
Rankc=rank([B A*B A^2*B]) % To check the controllability
Ranko= rank([C' A'*C' A'^2*C']) % To check the observability
Rankoc= rank([C*B C*A*B C*A^2*B]) % To check the output Controllability
M=ctrb(A,B);
rank_of_M=rank(M);
if(rank_of_M==0)
    disp('the given tf is not controllable');
else
    disp('the given tf is  controllable ');% since rank exists which is
equal to order of A
end
systemorder=length(A);
N=(obsv(A,C));
rank_of_N=rank(N);
if(rank_of_N==0)
    disp('the given tf is not observable');% since rank exists which is
equal to order of A
else
    disp('the given tf is  observable');
end
```

OUTPUT

Rank of the Matrix
Rankc = 3
Ranko = 3
Rankoc =1
the given tf is controllable

the given tf is observable

Expt. No:12	Date:

TEST OF CONTROLLABILITY AND OBSERVABILITY IN CONTINUOUS AND DISCRETE DOMAIN IN SIMULATION PLATFORM

AIM

To check for controllability and observability by MATLAB Program

APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

RESULT

Thus the controllability and observability of the system verified by MATLAB Program.

PROGRAM:

```
% Description: M-file showing design and implementation of dynamic
% pole placement state feedback controller
%
clear ; clf ; % clear memory and figure
A = [0 1; -5 -8] ; % state matrices
B = [0; 2] ;
C = [1 1] ;
Acl = [0 C; 0 A(1,:); 0 A(2,:)] ; % closed-loop system w/error dynamics
Bcl = [0; B] ;
yd = 3.0 ; % desired output
dt = 0.001 ; % simulation time step
% Peform First State Feedback Design for poles at -2,-2 and -20
K = acker(Acl, Bcl, [-2 -2 -20]) % determine feedback gains
x = [0; 0] ; y = C*x ; % initial conditions
inte = 0.0 ;
tvec = 0.0 ; yvec = 0.0 ; i = 1 ; % define vectors for storing outputs
for t = 0.0:dt:5.0, % loop over time for simulation
inte = inte + dt*(y - yd) ; % integral of output error
u = -K*[inte; x] ; % state controller
xdot = A*x + B*u ; % plant dynamics
x = xdot*dt + x ; % euler integrate dynamics
y = C*x ; % output equation
yvec(i) = y ; tvec(i) = t ; % store output & time into vectors
i = i + 1 ; % increment vector index
end ;
plot(tvec, yvec) ; % plot output with labels
xlabel('time (sec)') ; ylabel('y') ;
title('Output Response of Controlled System') ;
hold on ;
% Perform Second State Feedback Design for ploles at -1,-2 and -20
[num,den] = ss2tf(A,B,C,[0]) ; % find TF for open loop system
printsys(num,den,'s') ;
K = acker(Acl, Bcl, [-1 -2 -20]) % determine feedback gains
x = [0; 0] ; y = C*x ; % initial conditions
inte = 0.0 ;
tvec = 0.0 ; yvec = 0.0 ; i = 1 ; % define vectors for storing outputs
for t = 0.0:dt:5.0, % loop over time for simulation
inte = inte + dt*(y - yd) ; % integral of output error
u = -K*[inte; x] ; % state controller
xdot = A*x + B*u ; % plant dynamics
x = xdot*dt + x ; % euler integrate dynamics
y = C*x ; % output equation
yvec(i) = y ; tvec(i) = t ; % store output & time into vectors
i = i + 1 ; % increment vector index
end ;
plot(tvec, yvec, 'r-.') ; % plot alternative design output
hold off ;
legend('poles = -2, -2, -20', 'poles = -1, -2, -20') ;
```

Expt. No: 13	Date:

STATE FEEDBACK AND STATE OBSERVER DESIGN AND EVALUATION OF CLOSED LOOP PERFORMANCE

AIM

To check for State feedback and state observer design and evaluation of closed loop performance by MATLAB Program.

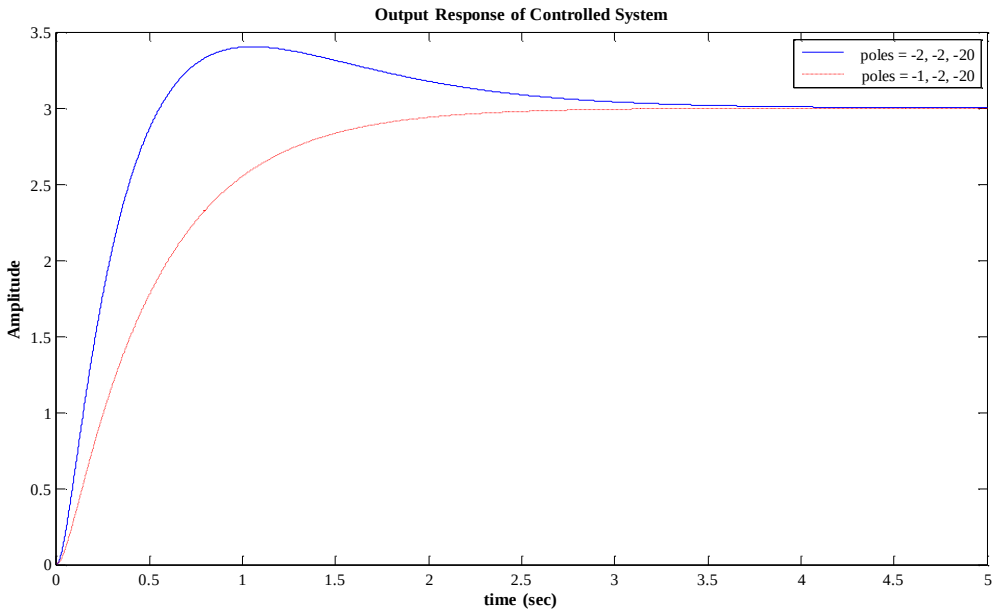
APPARATUS REQUIRED

- A PC with MATLAB package

PROCEDURE

- Open the MATLAB software using a MATLAB icon Open a blank M- File or Simulink file (File, New, M file)Type the given program in M file
- Run the program using debug option or using F5 key
- Find the stability of the system in the output graph and compare it with theoretical value

OUTPUT:



Pole locations are more than 10% in error.

K = 40.0000 -0.5000 8.0000

num/den = 2 s + 2

----

s^2 + 8 s + 5

K = 20.0000 8.5000 7.5000

ALTERNATE PROGRAM

```
%%  Original Plant
a=[-20.6 2;2 -1]
b=[5;1]
c=[1 1]
d=0
sys=ss(a,b,c,d)
eig(sys)
rank(obsv(sys))
%% Observer pole placement at  -10 and -9
% This observer will lead to a fast approximation of the states
L_T=place(a',c',[-10,-9])
L=L_T'
%% Observer pole placement at  -1 and -2
% This observer will lead to a slower approximation of the states
% L_anastrofos=place(a',c',[-1,-2])
% L=L_anastrofos'
% State observer Feedback
a=[-20.6 1;0 -1]
b=[0;1]
c=[1 1]
d=0
sys=ss(a,b,c,d)
eig(sys)
rank(obsv(sys))
rank(ctrb(sys))
K=place(a,b,[-5 -6])
```

OUTPUT:

a = -20.6000 2.0000 2.0000 -1.0000
b = 5 1
c =1 1
d = 0
a = x1 x2 x1 -20.6 x2 2 -1
b = u1 x1 5 x2 1
c = x1 x2 y1 1 1
d = u1 y1 0
Continuous-time model.
ans = -20.8020 -0.7980
ans = 2
L_T = -6.7429 4.1429
L = -6.7429 4.1429
a = -20.6000 1.0000 0 -1.0000
b = 0 1
c = 1 1
d =0
a = x1 x2 x1 -20.6 1 x2 0 -1
b = u1 x1 0 x2 1
c = x1 x2 y1 1 1
d = u1 y1 0
Continuous-time model.
ans = -20.6000 -1.0000
ans = 2
ans = 2
K =227.7600 -10.6000

RESULT

Thus the State feedback and state observer design and evaluation of closed loop performance by MATLAB Program was done.