DEPARTMENT OF ELECTRICAL ENGINEERING

CONTROL SYSTEM LABORATORY MANUAL

NATIONAL INSTITUTE OF TECHNOLOGY (NIT) ANDHRA PRADESH

(An Institution of National Importance)

DEPARTMENT OF ELECTRICAL ENGINEERING

TADEPALLIGUDEM– 534101, WEST GODAVARI DIST., A.P., INDIA

राष्ट्रीय प्रौद्योगिकी संस्थान आंध्रप्रदेश

NATIONAL INSTITUTE OF TECHNOLOGY (NIT) ANDHRA PRADESH

DEPARTMENT OF ELECTRICAL ENGINEERING

III B. Tech (EEE) II- Semester

EE 356 - Control Systems Lab

List of Experiments

1. Time-response of first and second order systems.
   a) Simulation of a typical second order system and determination of step response and evaluation of time- domain specifications.
   b) Evaluation of the effect of additional poles and zeroes on time response of second order system.
2. Frequency-response of second order system.
3. Study of P, PI & PID controller.
4. To study the effect of P, PI, PD and PID controller on the step response of a feedback control system
5.
   a) Evaluation of effect of pole location on stability
   b) Effect of loop gain of a negative feedback system on stability
6.
   a) Effect of open loop and zeroes on root locus contour
   b) To estimate the effect of open loop gain on the transient response of closed loop system by using Root locus
   c) Comparative study of Bode, Nyquist and Root locus with respect to Stability.
7. Design and study of lag, lead and Lag-lead compensator networks.
8. DC Position Control
9. DC Motor Speed Control
10.
    a) Block diagram reduction technique using MATLAB
    b) State model for classical transfer function & vice versa using MATLAB.

Control System Lab

INDEX

S. No	Experiment Name	Date	Hardware / MATLAB
1.	Time response of second order system a) Simulation of a typical second order system and determination of step response and evaluation of time- domain specifications. b) Evaluation of the effect of additional poles and zeroes on time response of second order system.	20/01/2021	MATLAB
2.	Frequency response of a second order system	27/01/2021	MATLAB
3.	Study of P, PI, PD, PID Controller	03/02/2021	Hardware
4.	Study the effect of P, PI, PD and PID controller on the Step response of a feedback control system	10/02/2021	Hardware & MATLAB
5.	a) Evaluation of effect of pole location on stability. b) Effect of loop gain of a negative feedback system on stability	17/02/2021	MATLAB
6.	a) Effect of open loop and zeroes on root locus contour. b) To estimate the effect of open loop gain on the transient response of closed loop system by using Root locus. c) Comparative study of Bode, Nyquist and Root locus with respect to Stability.	24/02/2021	MATLAB
7.	Design and study of lag, lead and Lag-lead compensator networks	10/03/2021	MATLAB
8.	DC position Control	17/03/2021	Hardware
9.	DC Motor Speed Control	24/03/2021	Hardware
10.	a) Block diagram reduction technique using MATLAB b) State model for classical transfer function & vice versa using MATLAB	31/04/2021	MATLAB

Experiment No:01
TIME RESPONSE OF FIRST AND SECOND ORDER SYSTEM

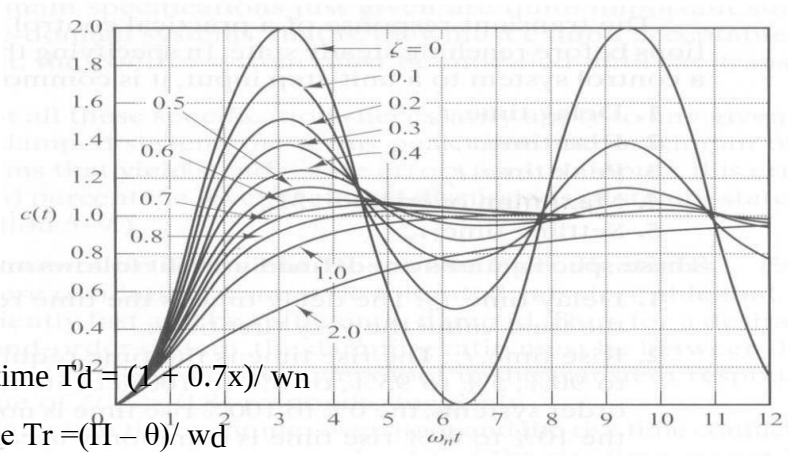
Aim: To obtain time response of a First order and second order system in case of under damped, over damped and critically damped systems.
Apparatus Required: PC loaded with MATLAB

Theory:

The time response of control system consists of two parts. Transient response and steady state response. $C(t) = C_{tr}(t) + C_{ss}(t)$. Most of the control systems use time as its independent variable. Analysis of response means to see the variation of output with respect to time. The output of the system takes some finite time to reach to its final value. Every system has a tendency to oppose the oscillatory behavior of the system which is called damping. The damping is measured by a factor called damping ratio of the system. If the damping is very high then there will not be anyoscillations in the output. The output is purely exponential. Such system is called an over damped system.

If the damping is less compared to over damped case then the system is called a critically damped system. If the damping is very less then the system is called under damped system. With no damping system is undamped.

$1<x<\infty$ --- Over damped system.
$x=1$ --- Critically damped system. $0<x<1$ --- Under damped system.
$x=0$ --- Undamped system.



Time domain specifications: Delay time $T_d = (1 + 0.7x)/ w_n$

Rise Time $T_r = (\Pi - \theta)/ w_d$
Peak overshoot time = $T_p = \Pi/ w_d$
% $M_p = \exp(- \Pi x/\text{sqrt}(1-x^2))$
Settling Time $T_s = 4/xw_n$ (2% tolerance)

Procedure:

Open the MATLAB command window.

1) Click on file-new-M file to open the MATLAB editor window.
2) In the given MATLAB editor window enter the program to obtain the step response.

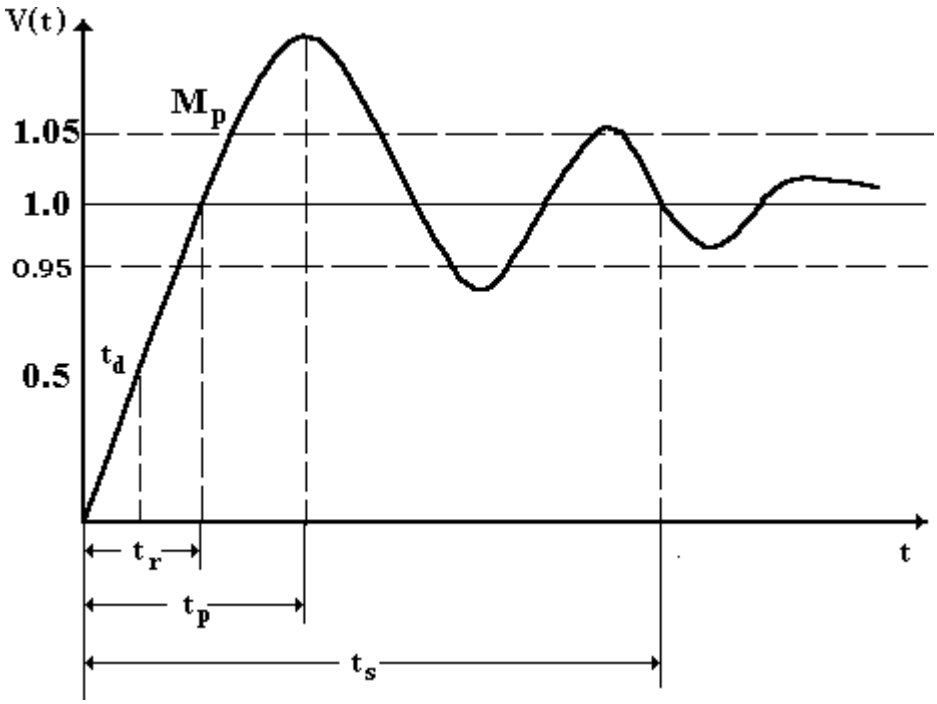
```
% step response of a second order system
wn = input('enter the natural frequency');
x = input('enter damping ratio');
num = [wn*wn];
den = [1 2*x*wn wn*wn];
sys = tf (num, den)
```

step(sys);

3) Save the file in work directory.
4) Run the program and enter the respective value for natural frequency, damping ratio and time
5) The graphs displayed are according to the above values

6) The values of wd, td, theta, tr, ts, Mp can be obtained by
    a) Right click on the figure window and select grid to get grids on the curve.
    b) Right click on the figure window and select characteristics and enable peak response, settling time & rise settling
    c) Repeat the steps 5,6,7 for different values of x.

Graph:



Tabular Columns :

Time Domain Specifications	From MATLAB	By Calculation
Rise Time		
Peak Time		
Settling Time		
Max. Overshoot		

1b) Evaluation of the effect of additional poles and zeros on time response of second order system.

MATLAB program to evaluate the effect of additional poles and zeros on time response of second order system

For the second order system, if we add a pole it changes to third order.

To study the effect of additional poles,

Location of poles	Effect on time response
-R	
-1/R	
-5R	

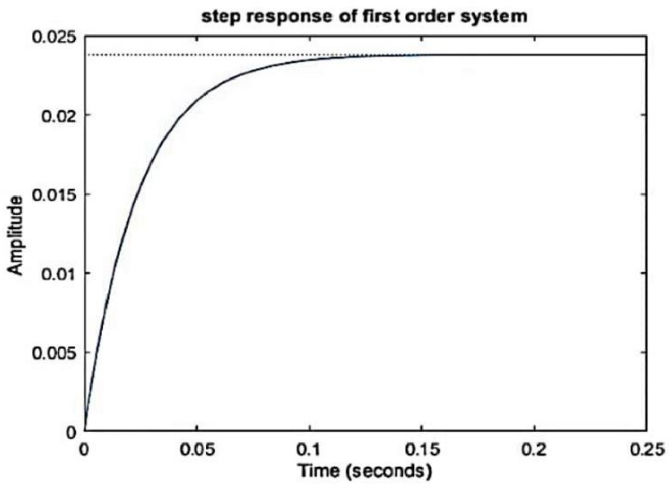
To study the effect of additional zeros,

Location of zeros	Effect on time response
-R	
-1/R	
-5R	

1. First Order System: (TF=1/s+41)

Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx

```
clc;
clear all

Step response of first order unity feedback system

%%G(s)=1/(s+41)
p=[0 1];
q=[1 41];
sys=tf(p,q);
t=feedback(sys,1);
figure(1)
step(t);
title('step response of first order system');
```



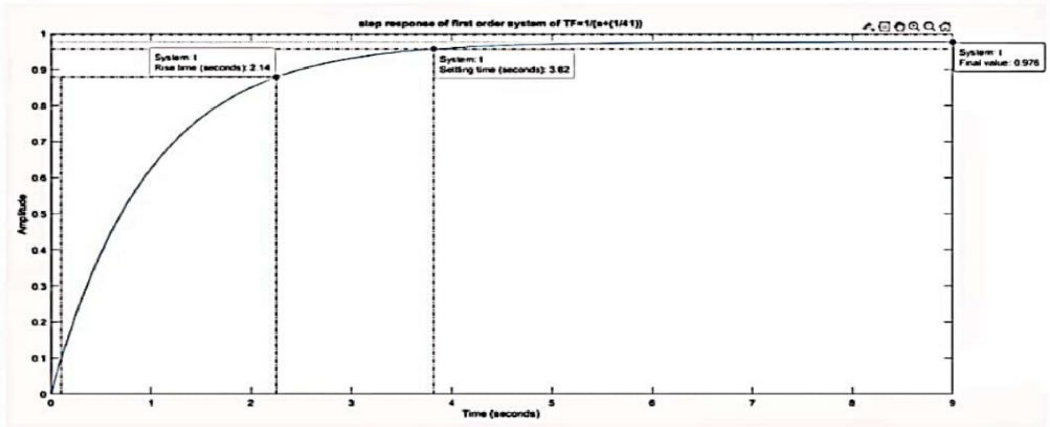
mmand Window

```
>> stepinfo(t)

ans =

  struct with fields:

       RiseTime: 0.0523
   SettlingTime: 0.0931
    SettlingMin: 0.0215
    SettlingMax: 0.0238
      Overshoot: 0
     Undershoot: 0
           Peak: 0.0238
       PeakTime: 0.2511
```

2.TF=1/(s+(1/41)):

Step response of first order unity feedback system TF=1/(s+(1/41))

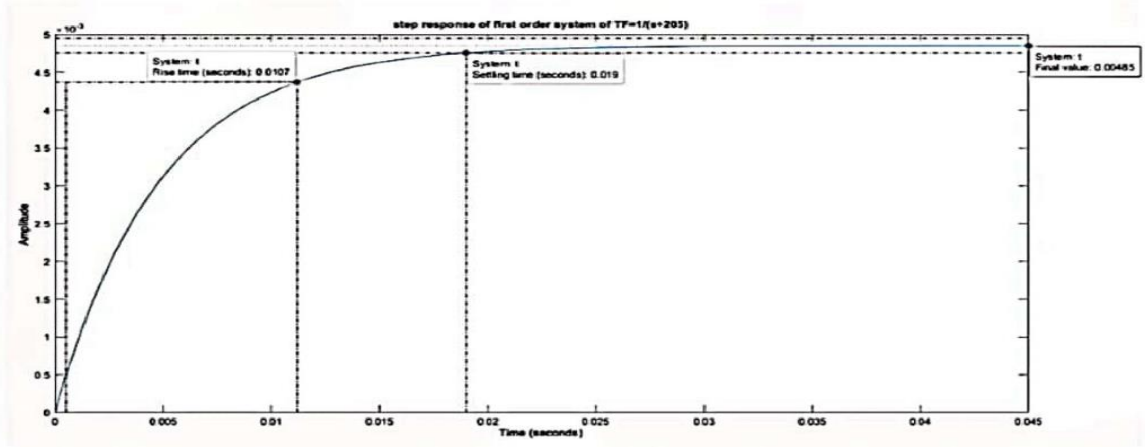
```
12  %%G(s)=1/(s+(1/41))
13  p=[0 1];
14  q=[1 1/41];
15  sys12=tf(p,q);
16  t=feedback(sys12,1);
17  figure(13)
18  step(t);
19  title('step response of first order system of TF=1/(s+(1/41))');
```



3.TF=1/s+205:

Step response of first order unity feedback system TF=1/(s+205)

```
20  %%G(s)=1/(s+205)
21  p=[0 1];
22  q=[1 205];
23  sys11=tf(p,q);
24  t=feedback(sys11,1);
25  figure(14)
26  step(t);
27  title('step response of first order system of TF=1/(s+205)');
```



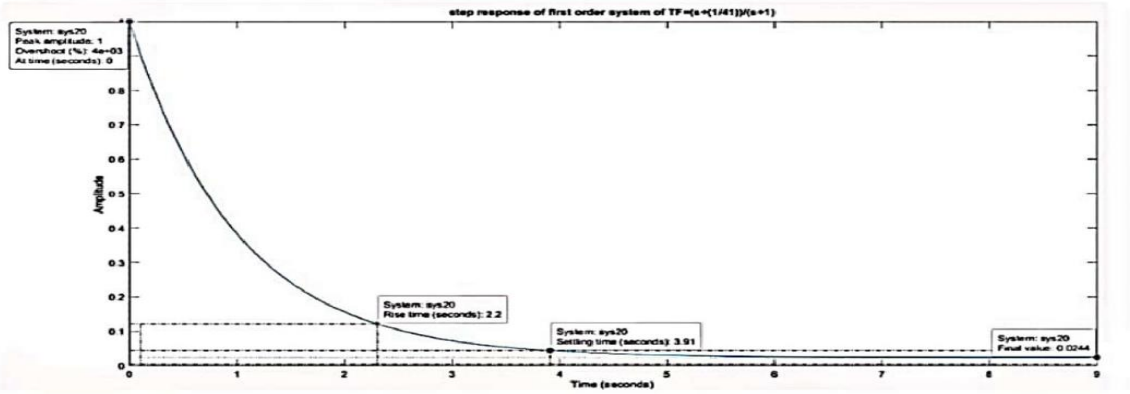
Effect of Zeroes on first order system:

1.Pole at -1, Zero at -1/41:

Effect of zeroes on first order system

Step response of first order system (pole at -1 and zero at -1/41)

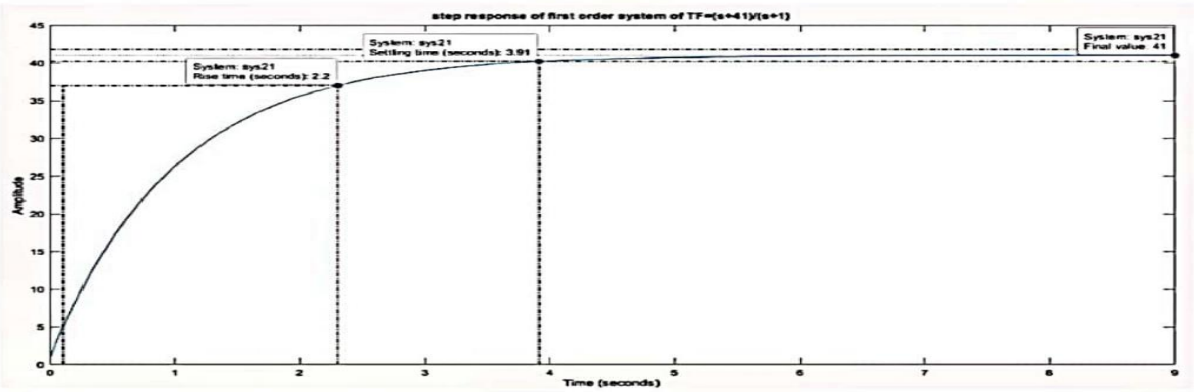
```
28  %%G(s)=(s+(1/41))/(s+1)
29  p=[1 1/41];
30  q=[1 1];
31  sys20=tf(p,q);
32  t=feedback(sys20,1);
33  figure(20)
34  step(sys20);
35  title('step response of first order system of TF=(s+(1/41))/(s+1)');
```



2.Pole at -1, Zero at -41:

Step response of first order system (pole at -1 and zero at -41)

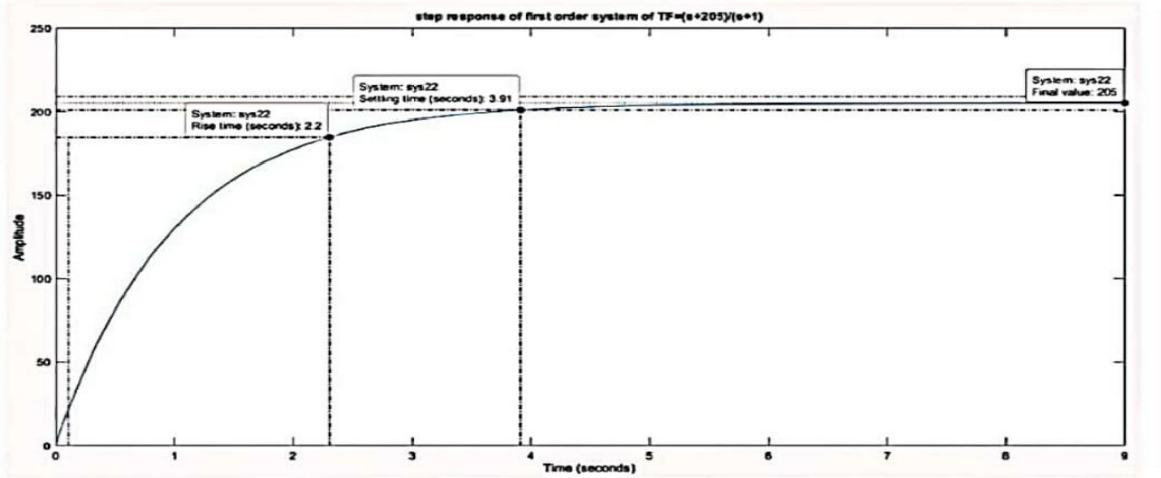
```
36  %%G(s)=(s+41)/(s+1)
37  p=[1 41];
38  q=[1 1];
39  sys21=tf(p,q);
40  t=feedback(sys21,1);
41  figure(21)
42  step(sys21);
43  title('step response of first order system of TF=(s+41)/(s+1)');
```



3.Pole at -1, Zero at -205(-5*41):

Step response of first order system (pole at -1 and zero at -205)

```
%%G(s)=(s+205)/(s+1)
p=[1 205];
q=[1 1];
sys22=tf(p,q);
t=feedback(sys22,1);
figure(22)
step(sys22);
title('step response of first order system of TF=(s+205)/(s+1)');
```

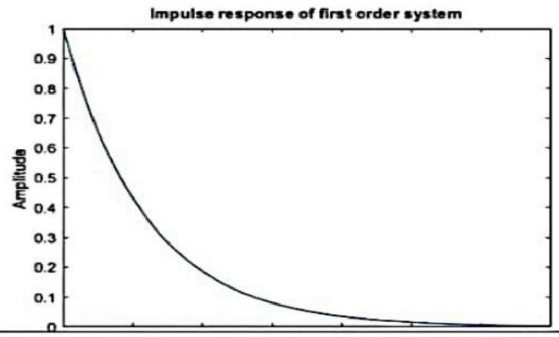


Impulse Response of 1st order system:

Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx*

Impulse response of first order unity feedback system

```
%%G(s)=1/(s+41)
p=[0 1];
q=[1 41];
sys=tf(p,q);
t=feedback(sys,1);
figure(2)
impulse(t);
title('impulse response of first order system');
```



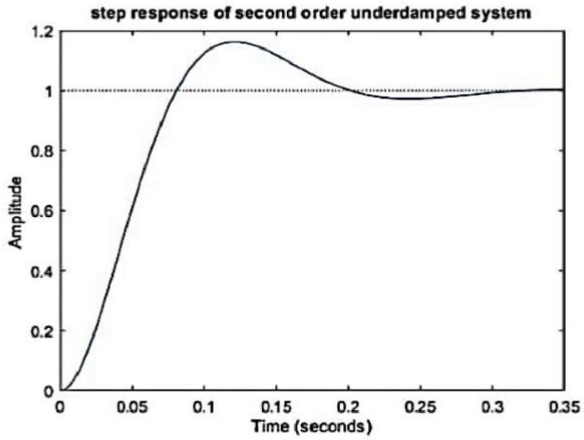
2. Second Order System:

1.Underdamped:

Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx *

Step response of second order underdamped system

```
%%G(s)=w^2/(s^2+2*e*w*s+w^2)
w=30;
e=0.5;
p=[0 0 w*w];
q=[1 2*e*w w*w];
sys=tf(p,q);
figure(3)
step(sys);
title('step response of second order underdamped system');
```



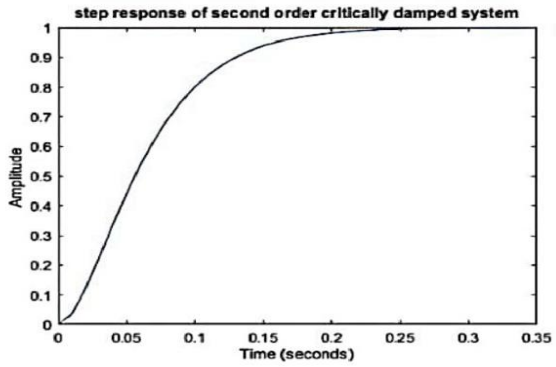

```
>> stepinfo(sys)

ans =

  struct with fields:

        RiseTime: 0.0546
    SettlingTime: 0.2692
     SettlingMin: 0.9315
     SettlingMax: 1.1629
       Overshoot: 16.2929
      Undershoot: 0
            Peak: 1.1629
        PeakTime: 0.1197
```

ii. Critically Damped:

Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx *

Step response of second order critically damped system

```
%%G(s)=w^2/(s^2+2*e*w*s+w^2)
w=30;
e=1;
p=[0 0 w*w];
q=[1 2*e*w w*w];
sys1=tf(p,q);
figure(4)
step(sys1);
title('step response of second order critically damped system');
```



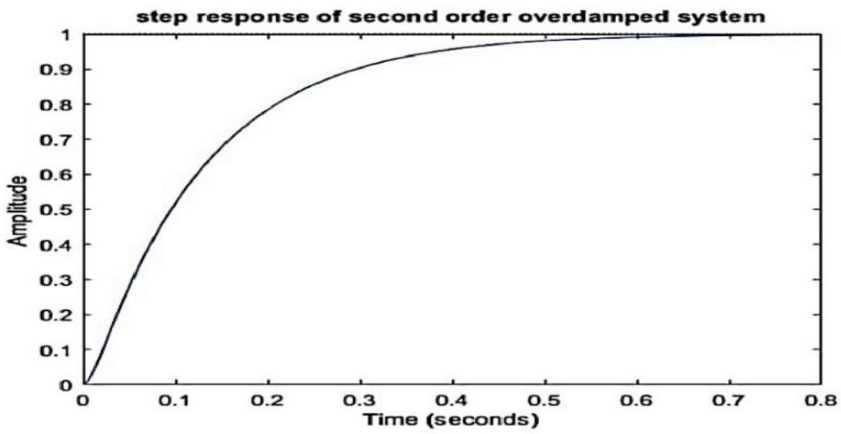

```
>> stepinfo(sys1)

ans =

  struct with fields:

        RiseTime: 0.1123
    SettlingTime: 0.1948
     SettlingMin: 0.9008
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 0.7900
```

iii. Over Damped:

Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx *

Step response of second order over damped system

```
%%G(s)=w^2/(s^2+2*e*w*s+w^2)
w=30;
e=2;
p=[0 0 w*w];
q=[1 2*e*w w*w];
sys2=tf(p,q);
figure(5)
step(sys2);
title('step response of second order overdamped system');
```




```
>> stepinfo(sys2)

ans =

  struct with fields:

        RiseTime: 0.2744
    SettlingTime: 0.4960
     SettlingMin: 0.9017
     SettlingMax: 0.9993
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9993
        PeakTime: 0.9109
```

Effect of Poles And Zeroes On second order system:

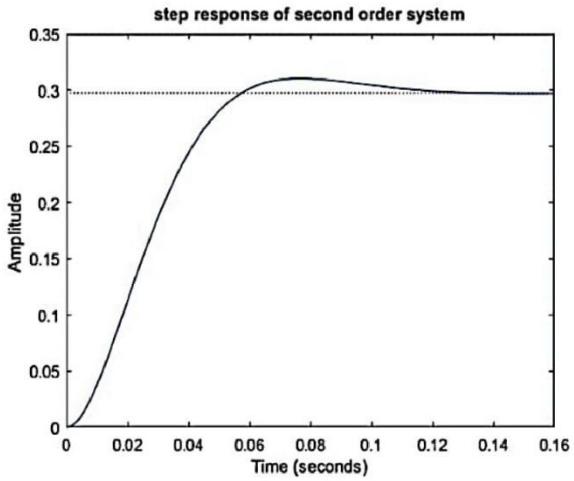
Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx

Effect of poles and zeores

```
z=[];
p=[-41+41i -41-41i];
k=1000;
sys3=zpk(z,p,k)
```

```
sys3 =

        1000
  ------------------
  (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.
```

```
figure(6)
step(sys3);
title('step response of second order system');
```




```
>> stepinfo(sys3)

ans =

  struct with fields:

        RiseTime: 0.0371
    SettlingTime: 0.1028
     SettlingMin: 0.2687
     SettlingMax: 0.3103
       Overshoot: 4.3210
      Undershoot: 0
            Peak: 0.3103
        PeakTime: 0.0764
```

Addition of pole at -1/41:

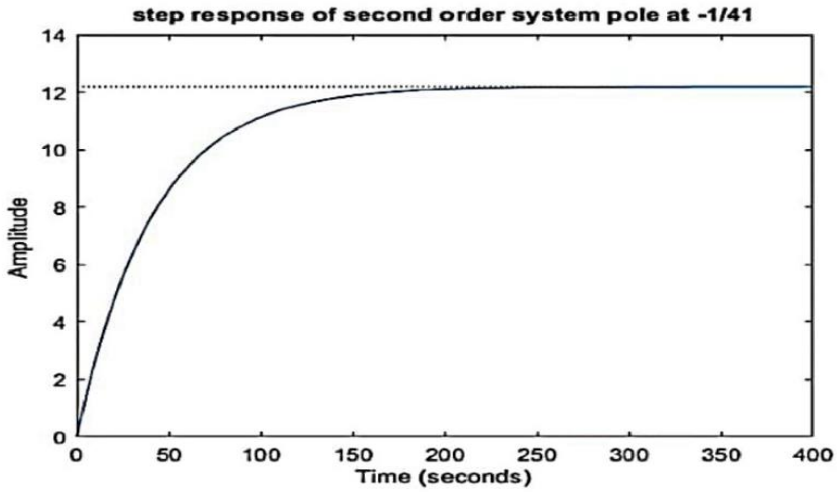
Exp_1_Batch_2_code.m | Exp_1_Batch_2_code.mlx | +

Addition of pole at -1/41

```
53  z=[];
54  p=[-41+41i -41-41i -1/41];
55  k=1000;
56  sys4=zpk(z,p,k)
```

```
sys4 =

                1000
  -------------------------------
  (s+0.02439) (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.
```

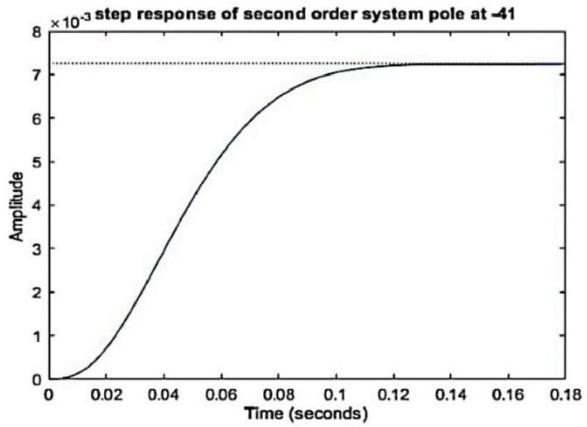
```
57  figure(7)
58  step(sys4);
59  title('step response of second order system pole at -1/41');
```




```
>> stepinfo(sys4)

ans =

  struct with fields:

        RiseTime: 90.0776
    SettlingTime: 160.4189
     SettlingMin: 11.0298
     SettlingMax: 12.1948
       Overshoot: 0
      Undershoot: 0
            Peak: 12.1948
        PeakTime: 432.3794
```

Addition of pole at -41:

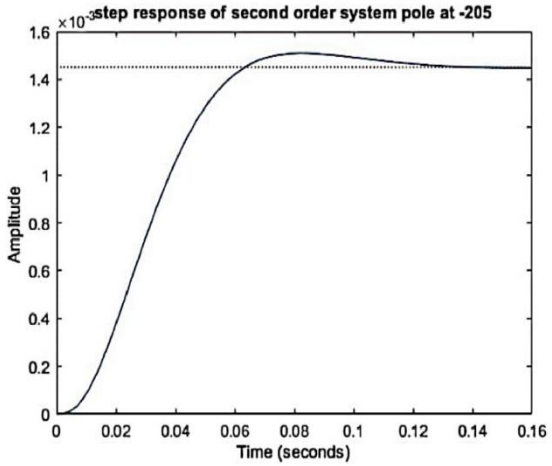
```
Addition of pole at -41

z=[];
p=[-41+41i -41-41i -41];
k=1000;
sys5=zpk(z,p,k)

sys5 =

           1000
  -------------------------
  (s+41) (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.

figure(R)
step(sys5);
title('step response of second order system pole at -41');
```




```
>> stepinfo(sys5)

ans =

  struct with fields:

       RiseTime: 0.0610
   SettlingTime: 0.1050
    SettlingMin: 0.0066
    SettlingMax: 0.0072
      Overshoot: 0
     Undershoot: 0
           Peak: 0.0072
       PeakTime: 0.2011
```

Addition of pole at -205(-5*41):

```
Addition of pole at -205

z=[];
p=[-41+41i -41-41i -205];
k=1000;
sys6=zpk(z,p,k)

sys6 =

           1000
  -------------------------
  (s+205) (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.

figure(9)
step(sys6);
title('step response of second order system pole at -205');
```




```
>> stepinfo(sys6)

ans =

  struct with fields:

       RiseTime: 0.0387
   SettlingTime: 0.1078
    SettlingMin: 0.0013
    SettlingMax: 0.0015
      Overshoot: 4.1000
     Undershoot: 0
           Peak: 0.0015
       PeakTime: 0.0831
```

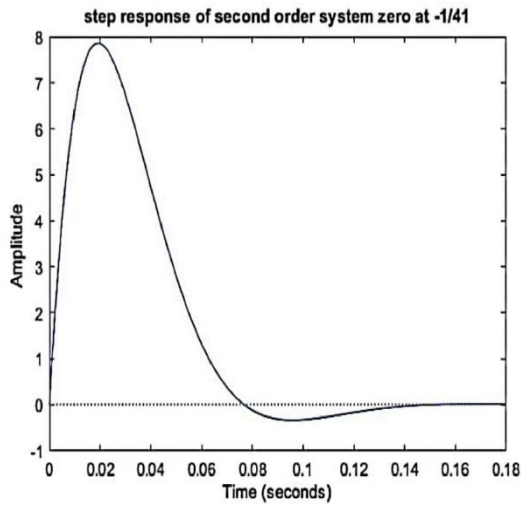
Addition of zero at -1/41:

Addition of zero at -1/41

```
z=[-1/41];
p=[-41+41i -41-41i];
k=1000;
sys7=zpk(z,p,k)
```

```
sys7 =

   1000 (s+0.02439)
  ------------------
  (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.
```

```
figure(10)
step(sys7);
title('step response of second order system zero at -1/41');
```




```
>> stepinfo(sys7)

ans =

  struct with fields:

       RiseTime: 6.0793e-06
   SettlingTime: 0.1220
    SettlingMin: -0.3323
    SettlingMax: 7.8659
      Overshoot: 1.0832e+05
     Undershoot: 4.5804e+03
           Peak: 7.8659
       PeakTime: 0.0191
```

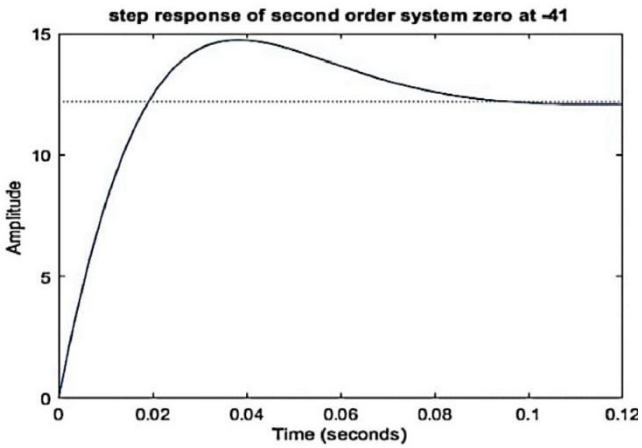
Addition of zero at -41:

Addition of zero at -41

```
z=[-41];
p=[-41+41i -41-41i];
k=1000;
sys8=zpk(z,p,k)
```

```
sys8 =

     1000 (s+41)
  ----------------
  (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.
```

```
figure(11)
step(sys8);
title('step response of second order system zero at -41');
```




```
>> stepinfo(sys8)

ans =

  struct with fields:

       RiseTime: 0.0146
   SettlingTime: 0.0844
    SettlingMin: 11.3784
    SettlingMax: 14.7302
      Overshoot: 20.7874
     Undershoot: 0
           Peak: 14.7302
       PeakTime: 0.0382
```

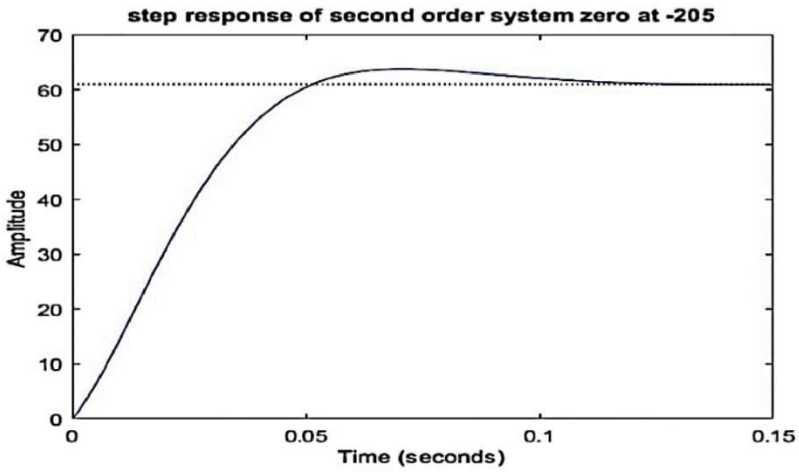
Addition of zero at -205(-5*41):

Addition of zero at -205

```
88  z=[-205];
89  p=[-41+41i -41-41i];
90  k=1000;
91  sys9=zpk(z,p,k)
```

```
sys9 =

     1000 (s+205)
  -----------------
  (s^2 + 82s + 3362)

Continuous-time zero/pole/gain model.
```

```
92  figure(12)
93  step(sys9);
94  title('step response of second order system zero at -205');
```




```
>> stepinfo(sys9)

ans =

  struct with fields:

        RiseTime: 0.0354
    SettlingTime: 0.0978
     SettlingMin: 55.0411
     SettlingMax: 63.7516
       Overshoot: 4.5526
      Undershoot: 0
            Peak: 63.7516
        PeakTime: 0.0708
```

Result:

Experiment No.2

FREQUENCY RESPONSE OF A SECOND ORDER SYSTEM

Aim: To determine frequency response of a second order system and evaluation of Frequency domain specifications using MATLAB.

Theory:

The frequency response of a system or a component is normally performed by keeping the amplitude A fixed and determining B and Φ for a suitable range of frequencies where steady state output may be represented as c(t) = B sin(wt + Φ) The ease and Accuracy of measurements are some of the advantages of the frequency response method. Without the knowledge of transfer function, the frequency response of stable open loop system can be obtained experimentally or the systems with very large time constants, the frequency response test is cumbersome to perform. We can use the data obtained from measurements on the physical system without deriving its mathematical model. Nyquist, bode, Nichols etc are some of the frequency response methods. For difficult cases, such as conditionally stable systems, Nyquist Plot is probably the only method to analyse stability.

$$G(s) = \frac{w_n^2}{s^2+2gw_ns+w_n^2}$$

Resonance Frequency $\omega_r = \omega_n \sqrt{(1- 2\zeta^2)}$

Resonance Peak $M_R = 1/2\zeta \sqrt{(1- \zeta^2)}$

Bandwidth $\acute{\omega}_b = \omega_n \sqrt{(1- \zeta^2)} + \sqrt{(1- \zeta^2)} + 1$

Procedure:

Procedure:

1) Open the MATLAB command window.
2) Click on file/new/M file to open the MATLAB editor window. In MATLAB editor window enter the program
3) Save the program as .M file.
4) Execute the program by selecting run.
5) Obtain the Specifications from the plot.

Theoretical Calculations:

Program:

```
numerator= [100]
denominator= [1 10 100]
sys=tf(numerator, denominator)
```

[Mr,wr]= getPeakGain(sys);

wb=bandwidth(sys)

bode(sys)

margin(sys)

grid

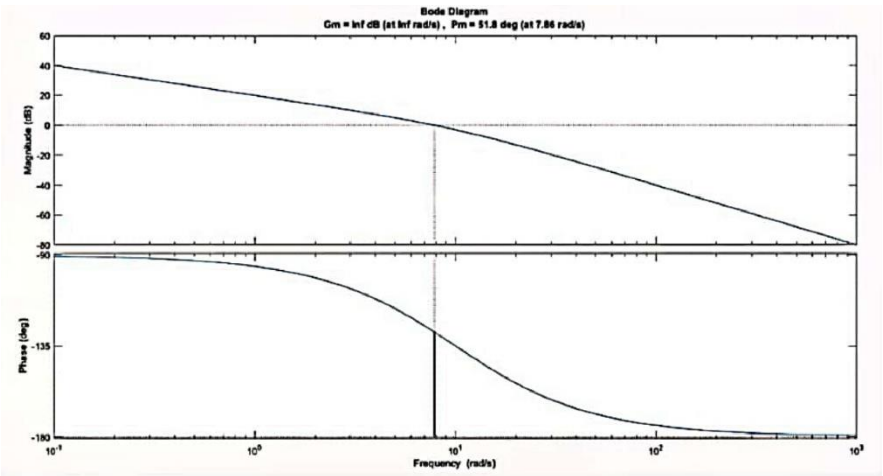
1. TF=100/s^2+10s+100

Command Window

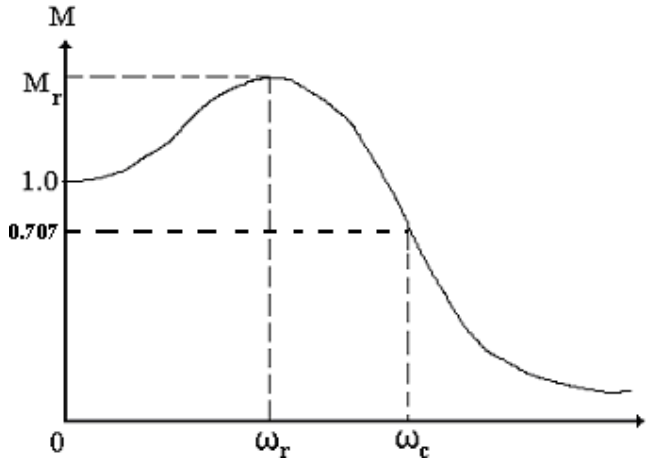
```
>> %%frequency response of second order system
w=input('enter natural frequency');
enter natural frequency10
>> e=input('enter damping ratio');
enter damping ratio0.5
>> num=w*w;
>> den=[1,2*e*w,w*w];
>> openlooptf=tf([100],[1,10,0])

openlooptf =

     100
  ----------
  s^2 + 10 s

Continuous-time transfer function.

>> g=tf(num,den)

g =

        100
  ----------------
  s^2 + 10 s + 100

Continuous-time transfer function.
```

Command Window

```
>> [mr,wr]=getPeakGain(g)

mr =

    1.1547


wr =

    7.0610

>> wb=bandwidth(g)

wb =

   12.7119

>> bode(openlooptf)
>> margin(openlooptf)
>> figure(1)
```



Graph:



Observations

Parameter	Practical Values	Theoretical values
ωr		
Mr		
ωb		
ωgc		
ωpc		
PM		
GM		

Result:

Experiment No: 03

Study of P, PI, PD, PID Controller

Objective: Study of P, PI, PD, PID Controller

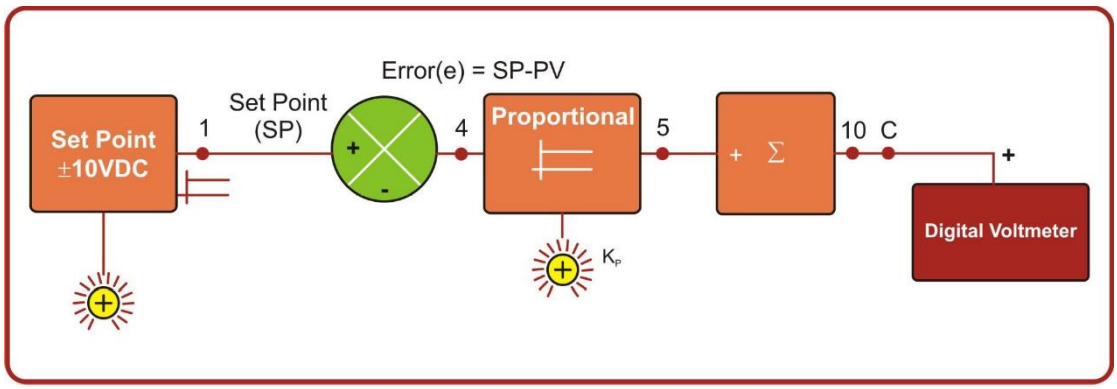
Apparatus Required

1. PID kit	1 No.
2. Patch Cord 16"	20 No.
3. Mains Cord	1 No.
4. TechBook Power Supply	1 No

Objective: Study of Proportional controller.

Procedure:

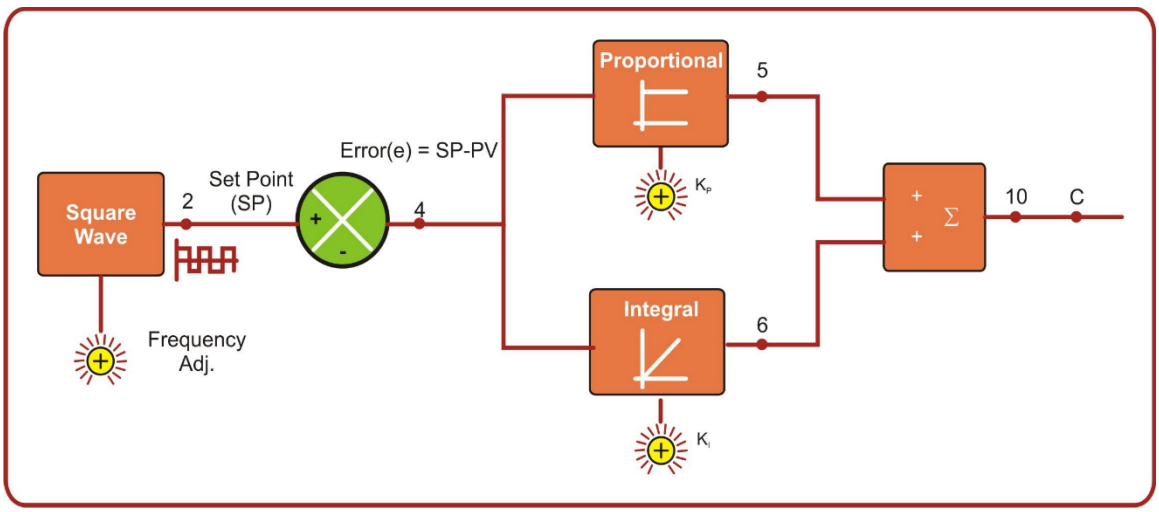
1. Connections for Proportional controller are as shown in the figure.
2. After making, all required connection on board which on the power supply and start the experiment.
3. Ground PV and inputs of summing block which are not in use.
4. Set +1.0V at test point ±10V set point output.
5. Apply set point to proportional input.
6. Check the output of proportional block with digital voltmeter given on board.
7. Vary slowly the Kp value, observe the change in the output, and find out the proportional band that is PB = 100/Kp.

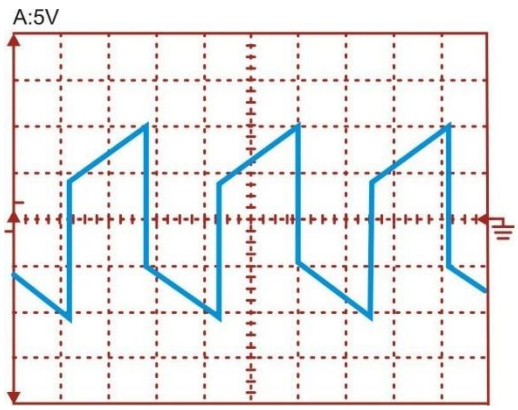


Objective: Study of Proportional + Integrator (PI)

Procedure:

1. Connections for Proportional + Integrator (PI) controller are as shown in the figure.
2. After making, all required connection on board which on the power supply and start the experiment.
3. Ground PV and inputs of summing block which are not in use.
4. Apply square wave to the set point (SP).
5. Check the output of summer block ∑ of summing block on CRO that will look like as shown in figure.
6. Vary slowly the KP & KI value and observe the changes in the output.

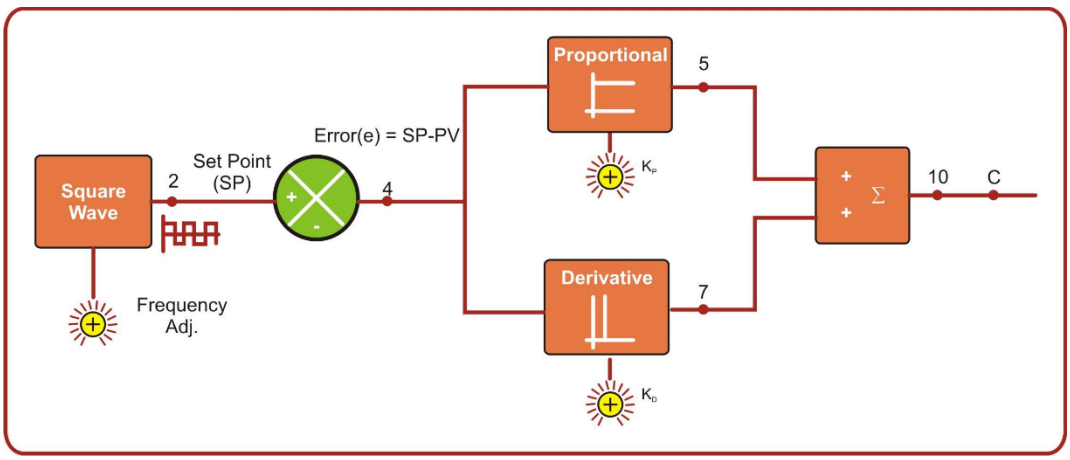


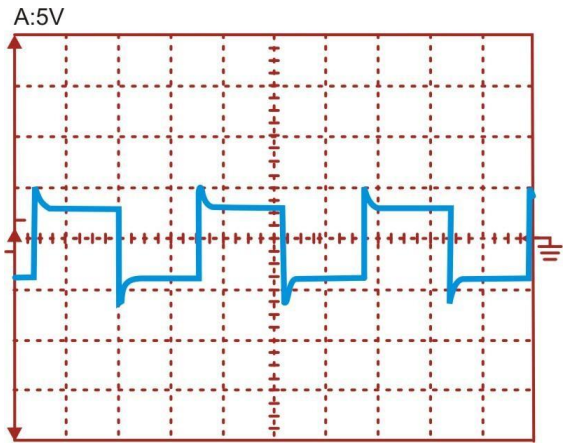


Objective: Study of Proportional + Derivative (PD)

Procedure:

1. Connections for Proportional + Derivative (PD) controller are as shown in the figure.
2. After making, all required connection on board which on the power supply and start the experiment.
3. Ground PV and inputs of summing block which are not in use.
4. Apply square wave to the set point (SP).
5. Check the output of summer block ∑ of summing block on CRO that will look like as shown in figure.
6. Vary slowly the KP & KD value and observe the changes in the output.

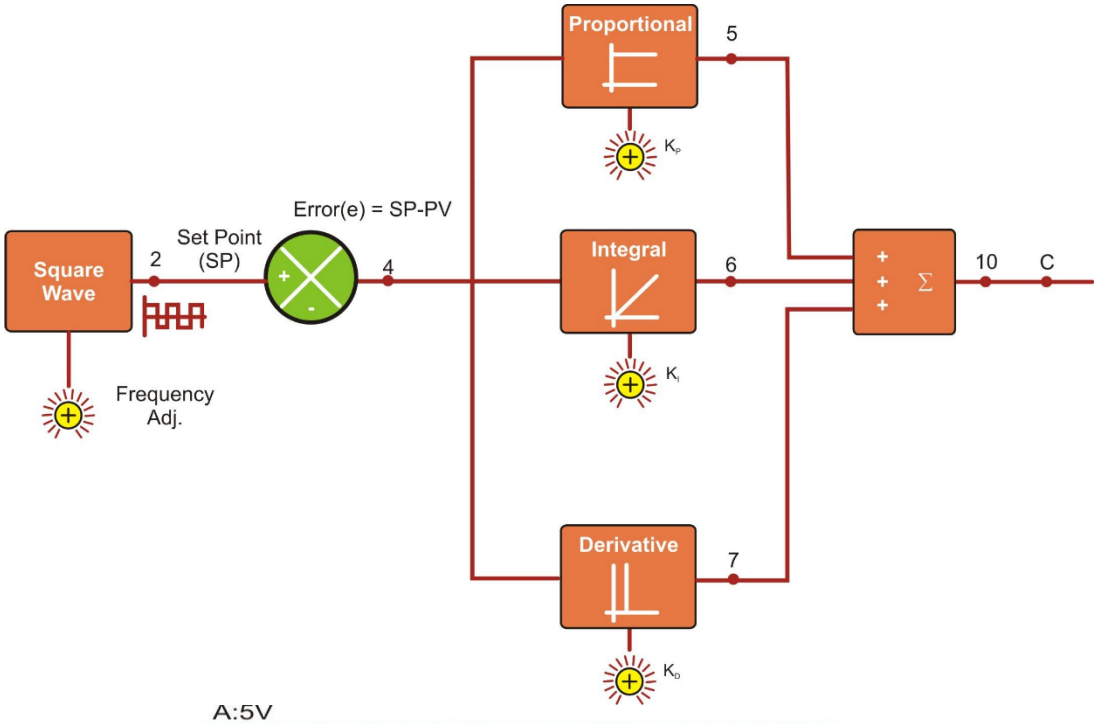


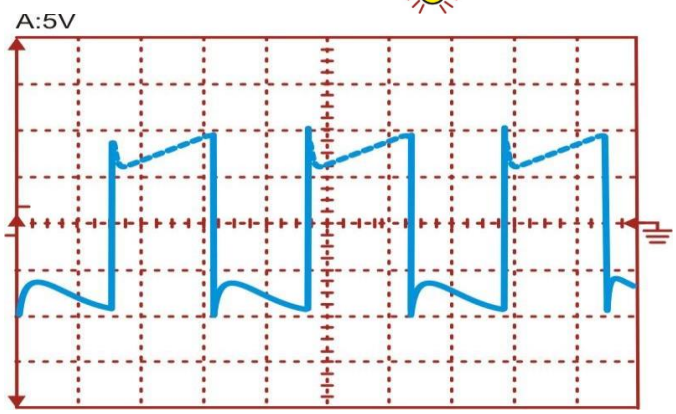


Objective: Study of Proportional + Integrator + Derivative (PID)

Procedure:

1. Connections for Proportional + Integrator + Derivative (PID) controller are as shown in the figure.
2. After making, all required connection on board switch on the power supply and start the experiment.
3. Ground PV and inputs of summing block which are not in use.
4. Apply square wave to the set point (SP).
5. Check the output of summer block ∑ of summing block on CRO that will look like as shown in figure.
6. Vary slowly the KP, KI & Kd value and observe the changes in the output.

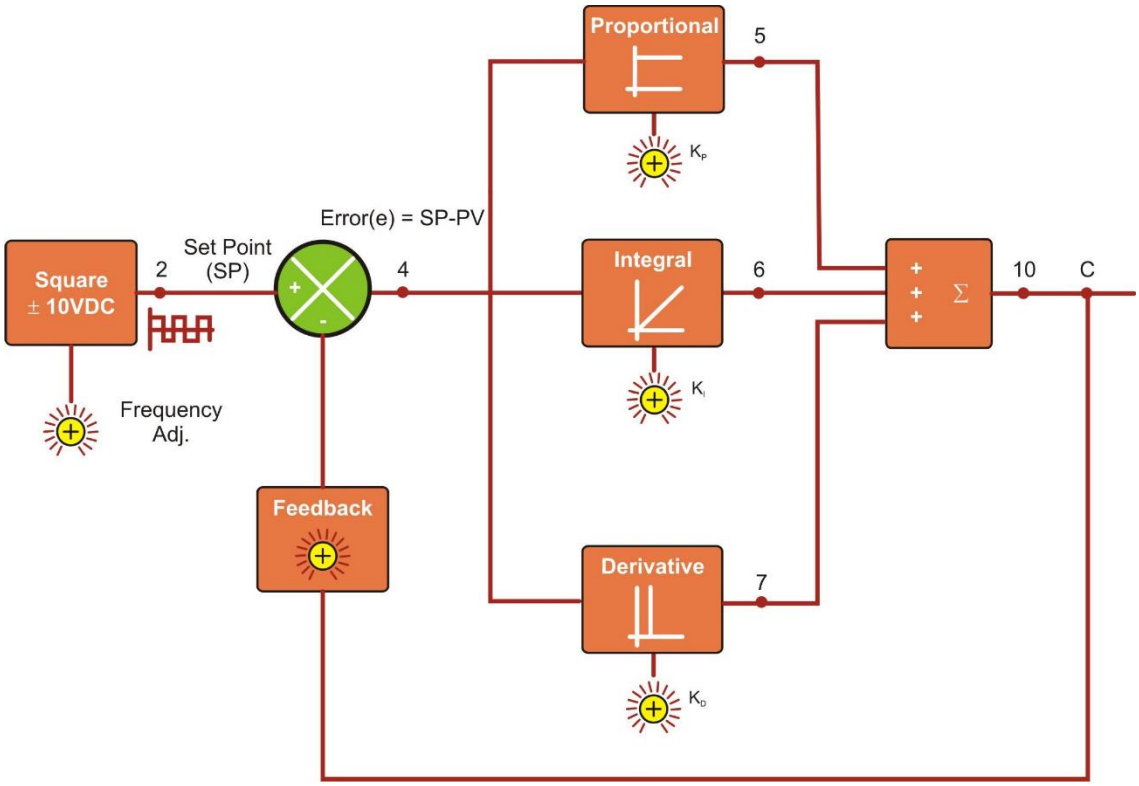




Objective: Study of Proportional + Integrator + Derivative (PID) in close loop

Procedure:

1. Connections for Proportional + Integrator + Derivative (PID) controller are as shown in the figure.
2. After making, all required connection on board Switch on the power supply and start the experiment.
3. Ground inputs of summing block which are not in use.
4. Apply square wave to the set point (SP).
5. Check output of summing block on CRO, you will find that after adjusting Kp, Ki & Kd output signal will be same as input applied at SP of error detector.



Result:

Experiment No: 04

Study the effect of P, PI, PD and PID controller on the step response of a feedback control system

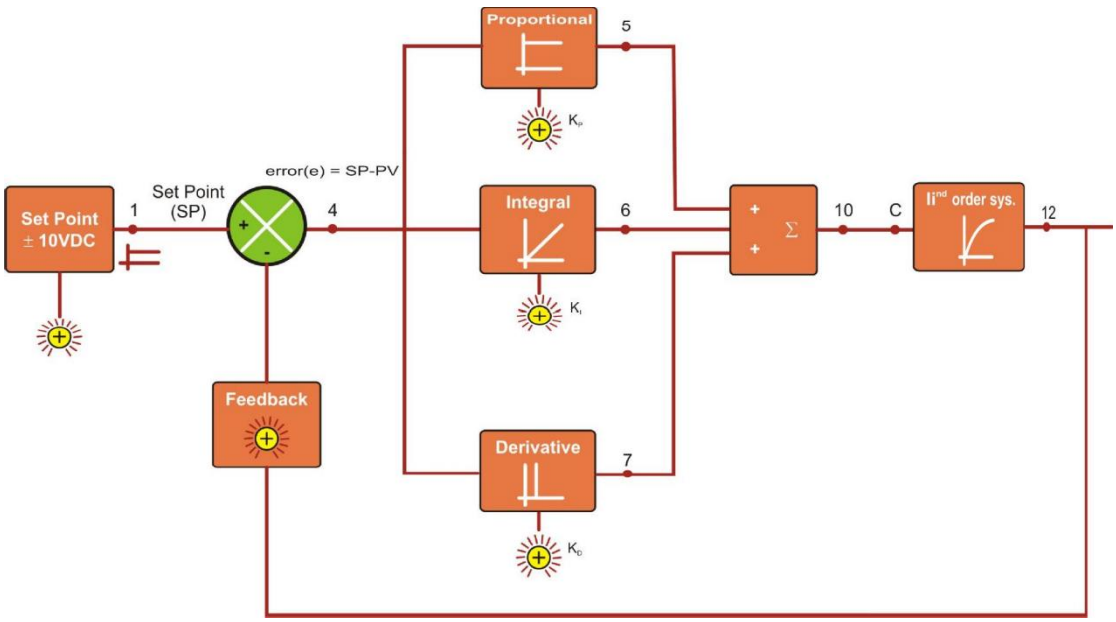
Objective: To Implement PID Controller to closed loop second order system with unit step input.

Apparatus Required:

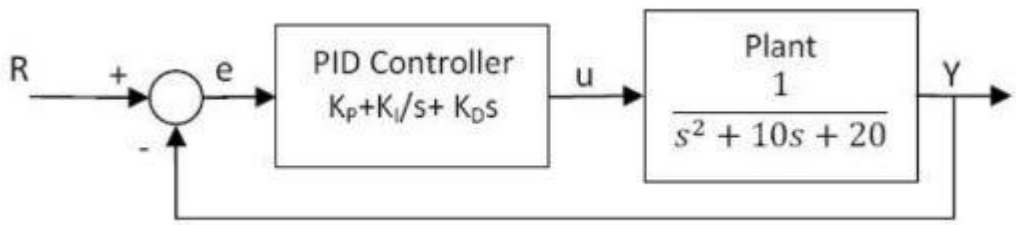
1. PID kit	1 No.
2. Patch Cord 16"	20 No.
3. Mains Cord	1 No.
4. TechBook Power Supply	1 No

Procedure:

1. The Connections for Proportional + Integrator + Derivative (PID) with second order system are as shown in the figure.
2. After making, all required connection on board which on the power supply and start the experiment.
3. Ground inputs of summing block which are not in use.
4. Check the transient response of IInd order system output on Digital storage oscilloscope, you will observe the response. You can easily observe the effect of PID by adjusting Kp, Ki & Kd knobs.

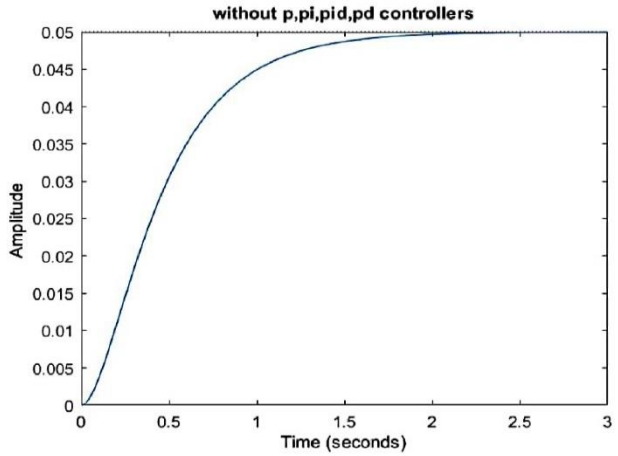


Note: Repeat the same experiment in Simulation for P, PI, PD, PID implementation.



Step Response Without any controller:

```
clc;
clear;
close all;
s=tf('s');
G=1/(s^2+10*s+20);
figure(1)
step(G)
title('without p,pi,pid,pd controllers');
```

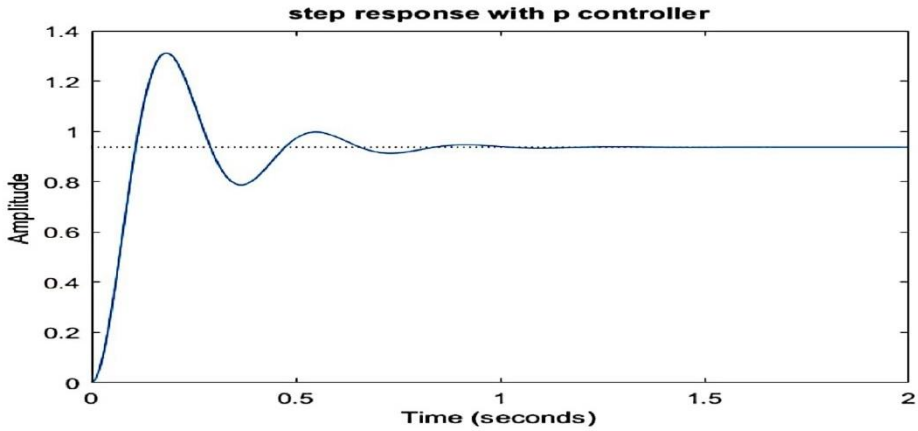


Command Window

```
>> stepinfo(G)

ans =

  struct with fields:

        RiseTime: 0.8843
    SettlingTime: 1.5894
     SettlingMin: 0.0452
     SettlingMax: 0.0500
       Overshoot: 0
      Undershoot: 0
            Peak: 0.0500
        PeakTime: 3.2966
```

With P controller:

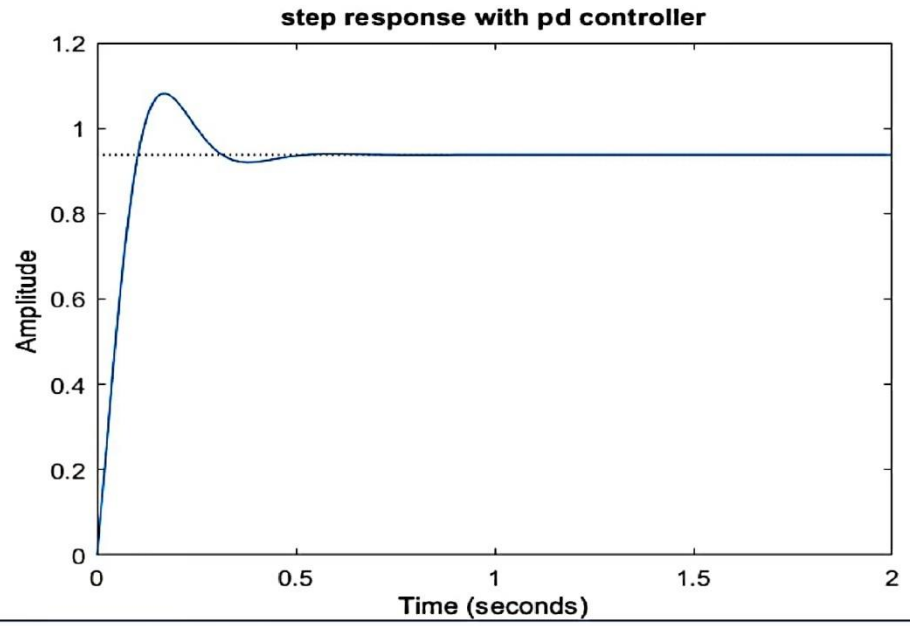
```
% p-controller
kp=300;
C=pid(kp,0,0);
plant=C*G;
a=feedback(plant,1);
b = 0:0.01:2;
figure(2)
step(a,b)
title('step response with p controller');
```




```
>> stepinfo(a)

ans =

  struct with fields:

        RiseTime: 0.0727
    SettlingTime: 0.7724
     SettlingMin: 0.7871
     SettlingMax: 1.3131
       Overshoot: 40.0588
      Undershoot: 0
            Peak: 1.3131
        PeakTime: 0.1842
```

With PD Controller:

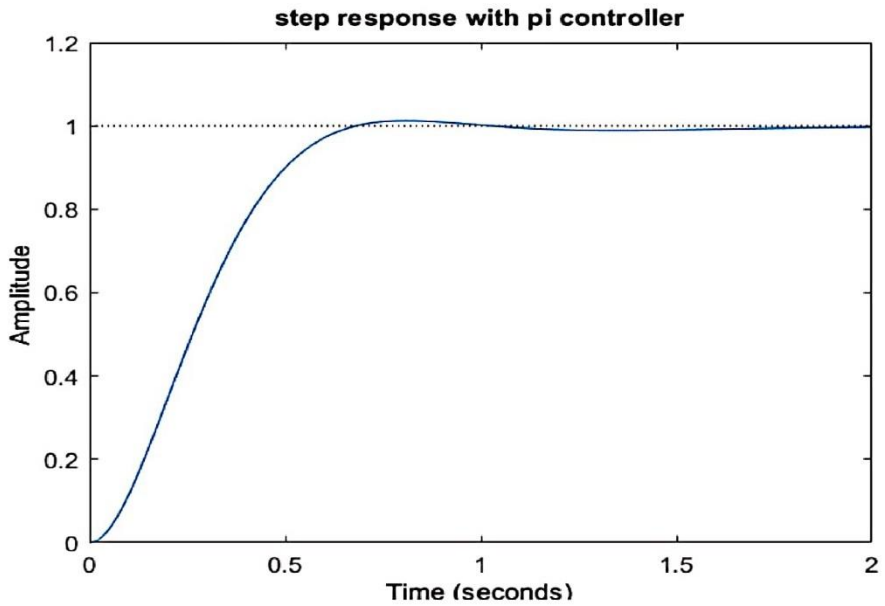
```
kd=10;
kp=300;
C=pid(kp,0,kd);
plant=C*G;
a1=feedback(plant,1);
b1=0:0.01:2;
figure(3);
step(a1,b1)
title('step response with pd controller');
```




```
>> stepinfo(a1)

ans =

  struct with fields:

        RiseTime: 0.0777
    SettlingTime: 0.2897
     SettlingMin: 0.8490
     SettlingMax: 1.0813
       Overshoot: 15.3418
      Undershoot: 0
            Peak: 1.0813
        PeakTime: 0.1704
```

With PI controller:

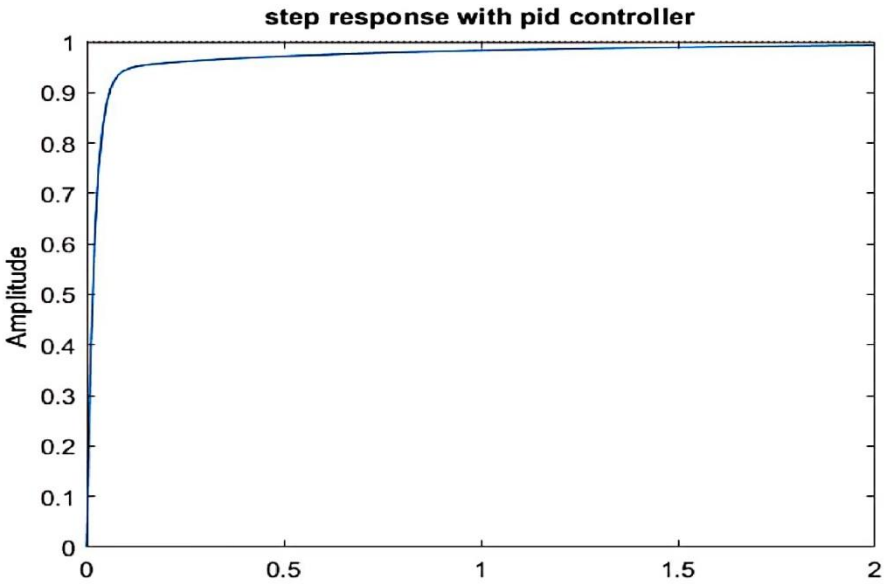
```
34    kp=30;
35    ki=70;
36    C=pid(kp,ki);
37    plant=C*G;
38    a2=feedback(plant,1);
39    b2=0:0.01:2;
40    figure(4);
41    step(a2,b2)
42    title('step response with pi controller');
```




```
>> stepinfo(a2)

ans =

  struct with fields:

        RiseTime: 0.4084
    SettlingTime: 0.6182
     SettlingMin: 0.9058
     SettlingMax: 1.0126
       Overshoot: 1.2588
      Undershoot: 0
            Peak: 1.0126
        PeakTime: 0.8143
```

With PID Controller:

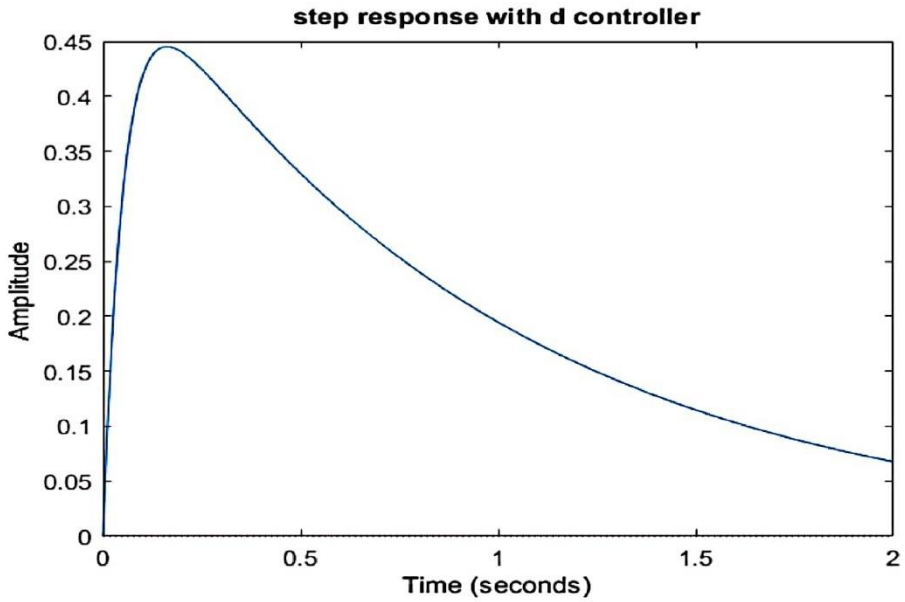
```
46    kp=350;
47    ki=300;
48    kd=50;
49    C=pid(kp,ki,kd);
50    plant=C*G;
51    a3=feedback(plant,1);
52    b3=0:0.01:2;
53    figure(5)
54    step(a3,b3)
55    title('step response with pid controller');
```




```
>> stepinfo(a3)

ans =

  struct with fields:

        RiseTime: 0.0548
    SettlingTime: 0.8308
     SettlingMin: 0.9006
     SettlingMax: 0.9959
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9959
        PeakTime: 2.4775
```

With D Controller:

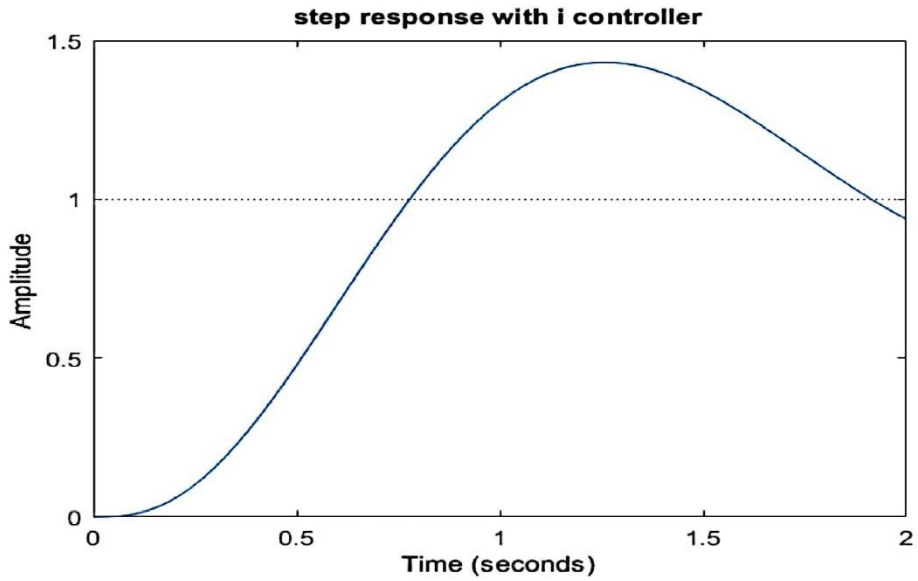
```
58  %d
59  kd=10;
60  C=pid(0,0,kd);
61  plant=C*G;
62  a4=feedback(plant,1);
63  b4=0:0.01:2;
64  figure(6)
65  step(a4,b4)
66  title('step response with d controller');
```




```
>> stepinfo(a4)

ans =

  struct with fields:

        RiseTime: 0
    SettlingTime: 3.9213
     SettlingMin: 5.7893e-04
     SettlingMax: 0.4452
       Overshoot: Inf
      Undershoot: 0
            Peak: 0.4452
        PeakTime: 0.1604
```

With I Controller:

```
%i
ki=70;
C=pid(0,ki,0);
plant=C*G;
a5=feedback(plant,1);
b5=0:0.01:2;
figure(7)
step(a5,b5)
title('step response with i controller');
```

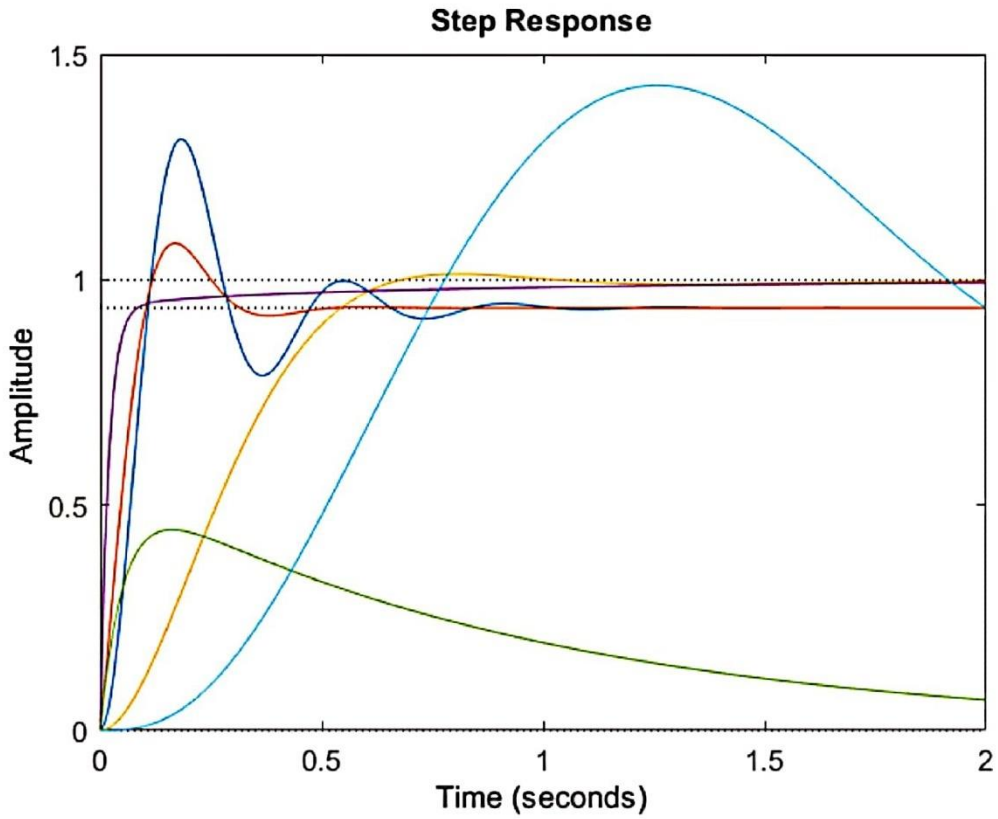



```
>> stepinfo(a5)

ans =

  struct with fields:

        RiseTime: 0.4734
    SettlingTime: 5.0784
     SettlingMin: 0.8028
     SettlingMax: 1.4315
       Overshoot: 43.1471
      Undershoot: 0
            Peak: 1.4315
        PeakTime: 1.2605
```

Comparison Graph:

comparision graph

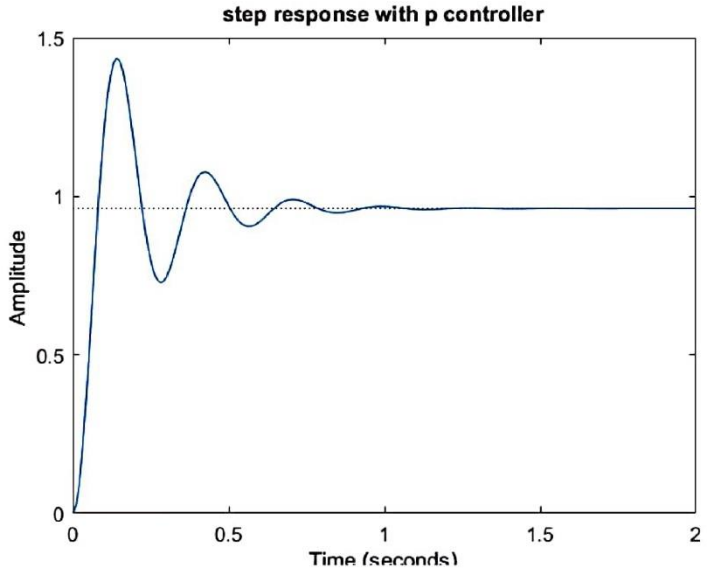
```
78  figure(8)
79  step(a,b);
80  hold on;
81  step(a1,b1);
82  hold on;
83  step(a2,b2);
84  hold on;
85  step(a3,b3);
86  hold on;
87  step(a4,b4);
88  hold on;
89  step(a5,b5);
90  hold on;
```



K_p=500:

With P controller:

```
%% p-controller
kp=500;
C=pid(kp,0,0);
plant=C*G;
a=feedback(plant,1);
b = 0:0.01:2;
figure(2)
step(a,b)
title('step response with p controller');
```



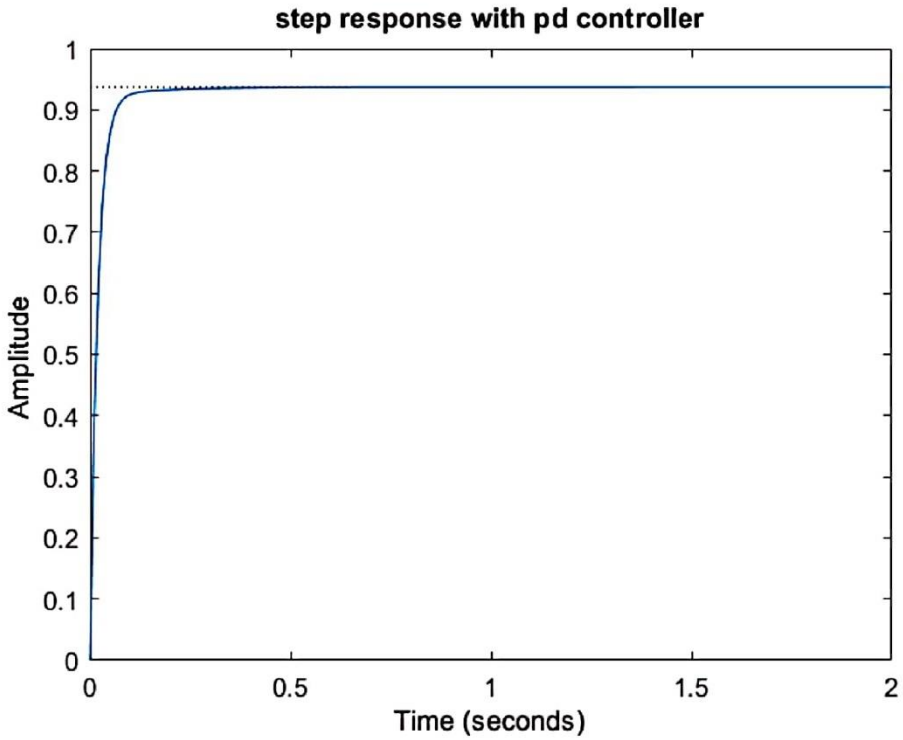
Command Window

```
>> stepinfo(a)

ans =

  struct with fields:

        RiseTime: 0.0540
    SettlingTime: 0.7443
     SettlingMin: 0.7278
     SettlingMax: 1.4350
       Overshoot: 49.2400
      Undershoot: 0
            Peak: 1.4350
        PeakTime: 0.1382

fx >>
```

Kp=300, kd=50

PD controller:

```
19
20  % pd-controller
21  kd=50;
22  kp=300;
23  C=pid(kp,0,kd);
24  plant=C*G;
25  a1=feedback(plant,1);
26  b1=0:0.01:2;
27  figure(3);
28  step(a1,b1)
29  title('step response with pd controller');
```

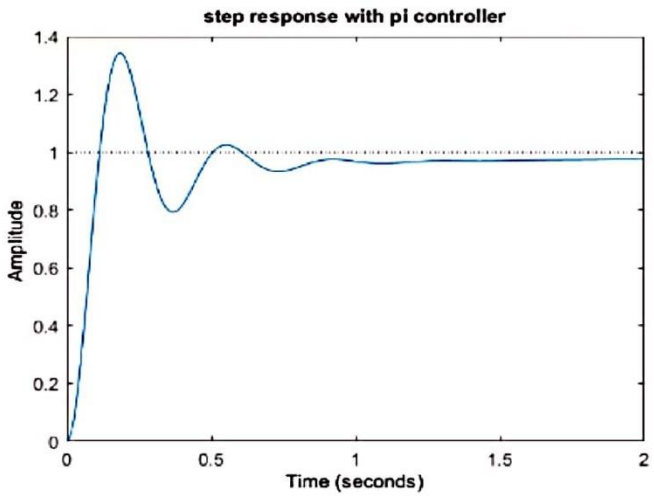


```
>> stepinfo(a1)

ans =

  struct with fields:

        RiseTime: 0.0427
    SettlingTime: 0.0839
     SettlingMin: 0.8459
     SettlingMax: 0.9339
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9339
        PeakTime: 0.2376
```

Kp=300,Ki=120

PI controller:

```
% pi-controller
kp=300;
ki=120;
C=pid(kp,ki,0);
plant=C*G;
a2=feedback(plant,1);
b2=0:0.01:2;
sys=(100*s+5)/(s^2+10*s+20);
[y,t]=step(sys);
sserror=abs(1-y(end));
figure(4);
step(a2,b2)
title('step response with pi controller');
```

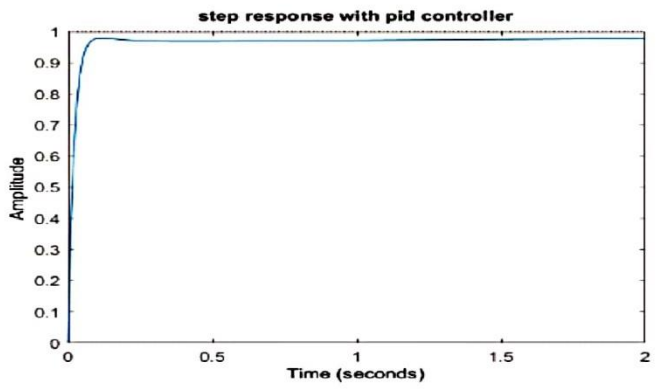


```
>> stepinfo(a2)

ans =

  struct with fields:

        RiseTime: 0.0760
    SettlingTime: 2.5328
     SettlingMin: 0.7925
     SettlingMax: 1.3432
       Overshoot: 34.3160
      Undershoot: 0
            Peak: 1.3432
        PeakTime: 0.1819
```

Kp=500, Kd=50, Ki=120

PID controller:

```
% pid-controller
kp=500;
kd=50;
ki=120;
C=pid(kp,ki,kd);
plant=C*G;
a3=feedback(plant,1);
b3=0:0.01:2;
figure(5)
step(a3,b3)
title('step response with pid controller');
```



```
>> stepinfo(a3)

ans =

  struct with fields:

        RiseTime: 0.0449
    SettlingTime: NaN
     SettlingMin: 0.9017
     SettlingMax: 0.9790
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9790
        PeakTime: 0.1152
```

Effect of each controller on the closed loop response are summarized below:

CL Response	Rise time	Overshoot	Settling time	Steady state error
K_P=				
K_P= K_I=				
K_P= K_D=				
K_P= K_I= K_D=				
K_P= K_I= K_D= (Ziegler–Nichols)				

Result:

Experiment No:05

a) **Evaluation of effect of pole location on stability**
b) **Effect of loop gain of a negative feedback system on stability**

AIM: Evaluation of effect of pole location and loop gain of a negative feedback system on stability

Apparatus: PC loaded with MATLAB

Procedure:

1. Open MATLAB
2. Create new script
3. Enter code run the program and obtain the required details
4. Observe the plots and note the values
5. Save and compare the observations

Theoretical Calculation:

Program:

Effect of poles and zeroes on stability

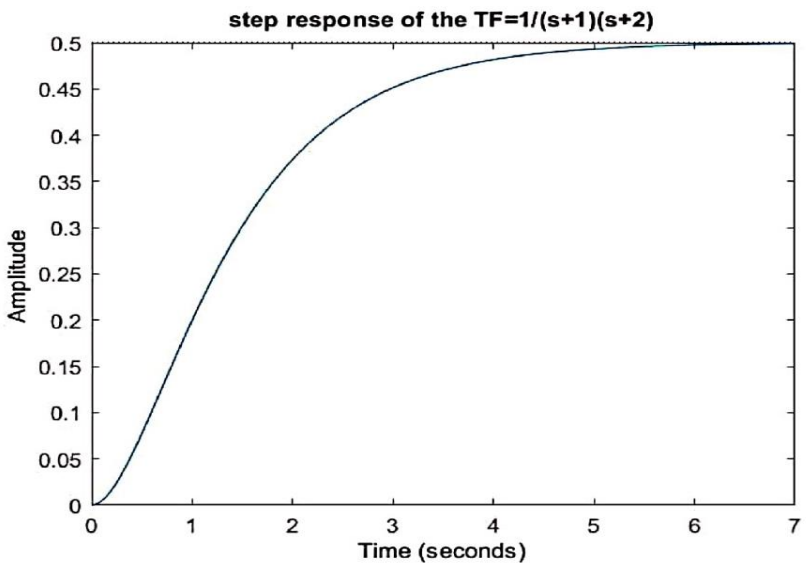
1.TF=1/(s+1)(s+2)

Open Loop response of transfer function 1/(s+1)(s+2)

```
p=[1];
q=[1 3 2];
G=tf(p,q);
figure(1)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: 2.5901
    SettlingTime: 4.6002
     SettlingMin: 0.4511
     SettlingMax: 0.4996
       Overshoot: 0
      Undershoot: 0
            Peak: 0.4996
        PeakTime: 7.7827
```

```
title('step response of the TF=1/(s+1)(s+2)');
```



closed loop response for unity negative feedback system of the transfer function 1/(s+1)(s+2)

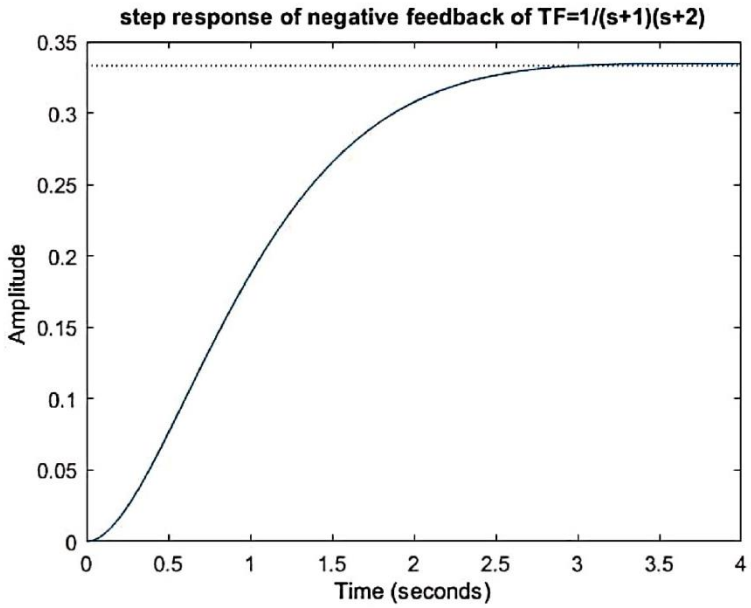
```
p=[1];
q=[1 3 2];
G=tf(p,q);
t=feedback(G,1)
```

```
t =

        1
  -------------
  s^2 + 3 s + 3

Continuous-time transfer function.
```

```
figure(2)
step(t);
stepinfo(t)
```

```
ans = struct with fields:
        RiseTime: 1.5787
    SettlingTime: 2.5089
     SettlingMin: 0.3017
     SettlingMax: 0.3348
       Overshoot: 0.4333
      Undershoot: 0
            Peak: 0.3348
        PeakTime: 3.6227
```

```
title('step response of negative feedback of TF=1/(s+1)(s+2)');
```



closed loop response for unity positive feedback system of the transfer function 1/(s+1)(s+2)

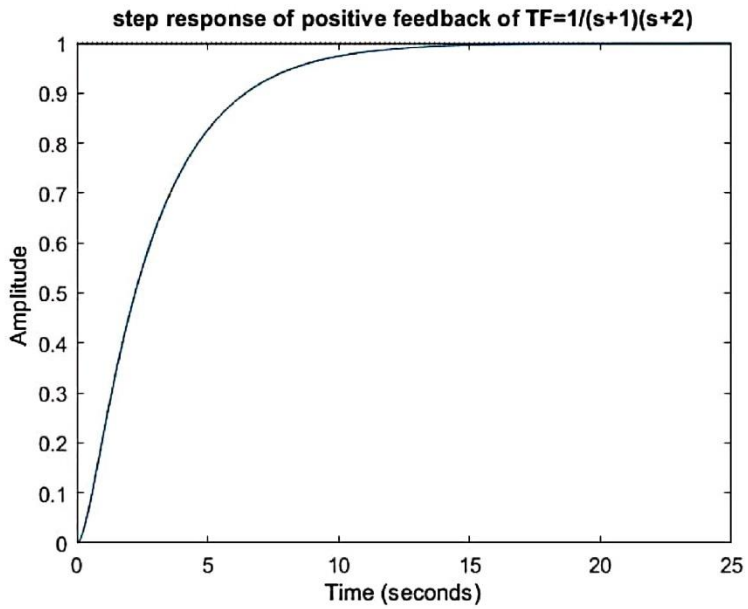
```
p=[1];
q=[1 3 2];
G=tf(p,q);
t=feedback(G,-1)
```

```
t =

        1
  -------------
  s^2 + 3 s + 1

Continuous-time transfer function.
```

```
figure(3)
step(t);
stepinfo(t)
```

```
ans = struct with fields:
        RiseTime: 5.8584
    SettlingTime: 10.6547
     SettlingMin: 0.9012
     SettlingMax: 0.9999
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9999
        PeakTime: 25.9983
```

```
title('step response of positive feedback of TF=1/(s+1)(s+2)');
```

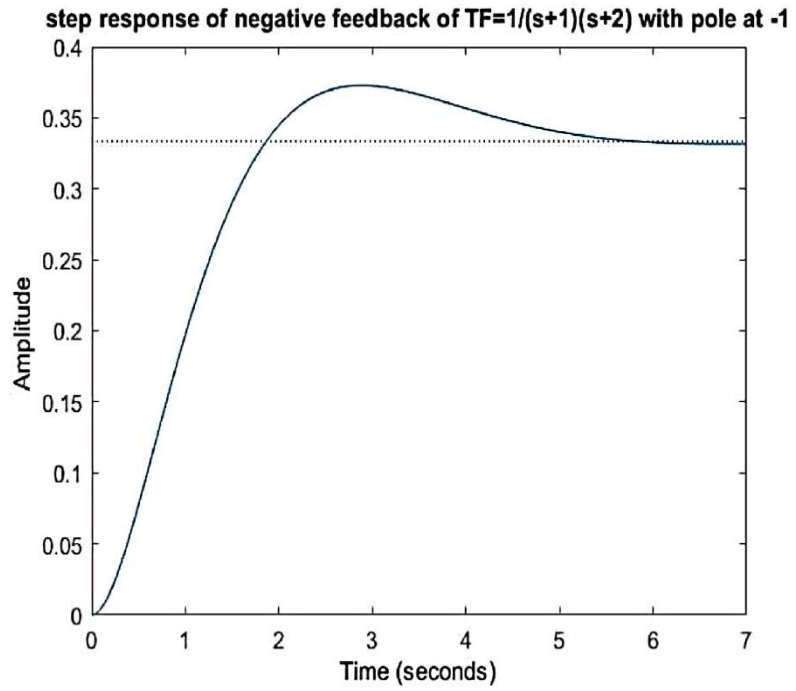


closed loop response for the TF=1/(s+1)(s+2) with pole at -1 in negative feedback path

```
p=[1 1];
q=[1 4 5 3];
G=tf(p,q);
figure(4)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: 1.2722
    SettlingTime: 5.0078
     SettlingMin: 0.3047
     SettlingMax: 0.3728
       Overshoot: 11.8450
      Undershoot: 0
            Peak: 0.3728
        PeakTime: 2.8764
```

```
title('step response of negative feedback of TF=1/(s+1)(s+2) with pole at -1');
```



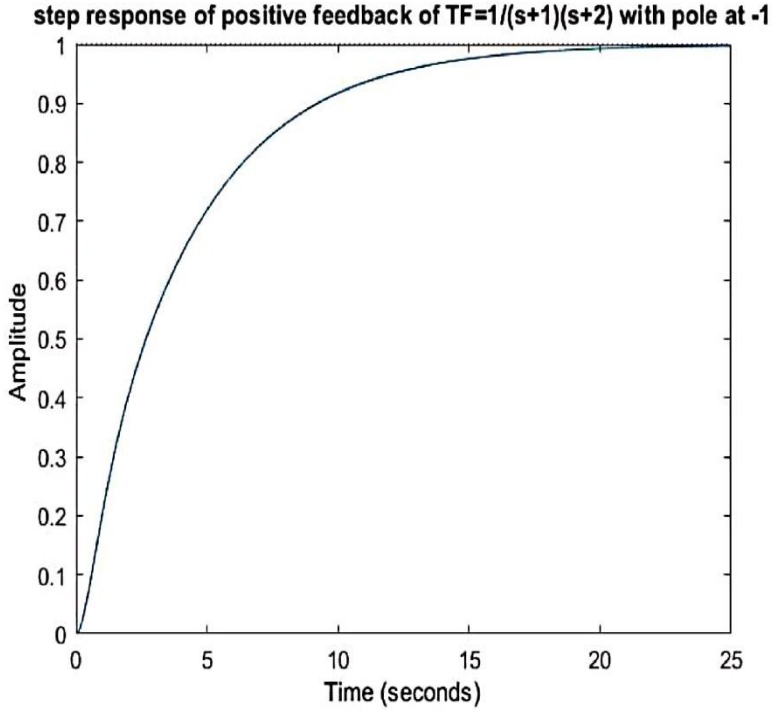
closed loop response for the TF=1/(s+1)(s+2) with pole at -1 in positive feedback path

```
p=[1 1];
q=[1 4 5 1];
G=tf(p,q);
figure(5)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
```

```
        RiseTime: 8.6212
    SettlingTime: 15.7786
     SettlingMin: 0.9003
     SettlingMax: 0.9988
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9988
        PeakTime: 27.4234
```

```
title('step response of positive feedback of TF=1/(s+1)(s+2) with pole at -1');
```

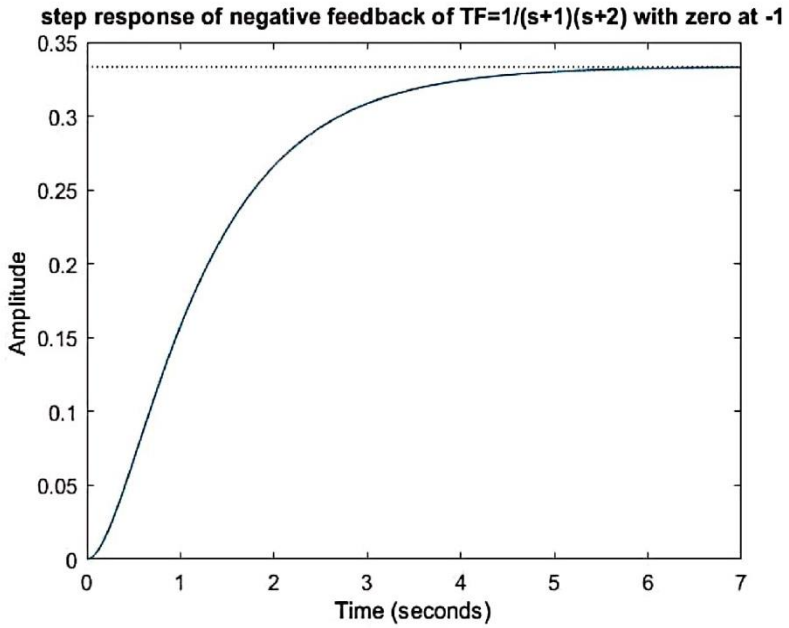


closed loop response for the TF=1/(s+1)(s+2) with zero at -1 in negative feedback path

```
p=[1];
q=[1 4 3];
G=tf(p,q);
figure(6)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: 2.3911
    SettlingTime: 4.3175
     SettlingMin: 0.3008
     SettlingMax: 0.3332
       Overshoot: 0
      Undershoot: 0
            Peak: 0.3332
        PeakTime: 7.9516
```

```
title('step response of negative feedback of TF=1/(s+1)(s+2) with zero at -1');
```

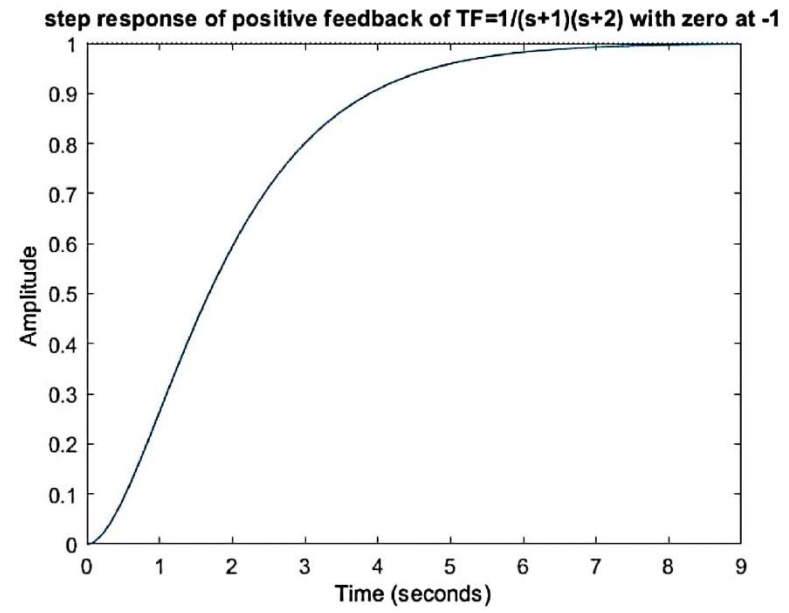


closed loop response for the TF=1/(s+1)(s+2) with zero at -1 in positive feedback path

```
p=[1];
q=[1 2 1];
G=tf(p,q);
figure(7)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: 3.3579
    SettlingTime: 5.8339
     SettlingMin: 0.9000
     SettlingMax: 0.9994
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9994
        PeakTime: 9.7900
```

```
title('step response of positive feedback of TF=1/(s+1)(s+2) with zero at -1');
```



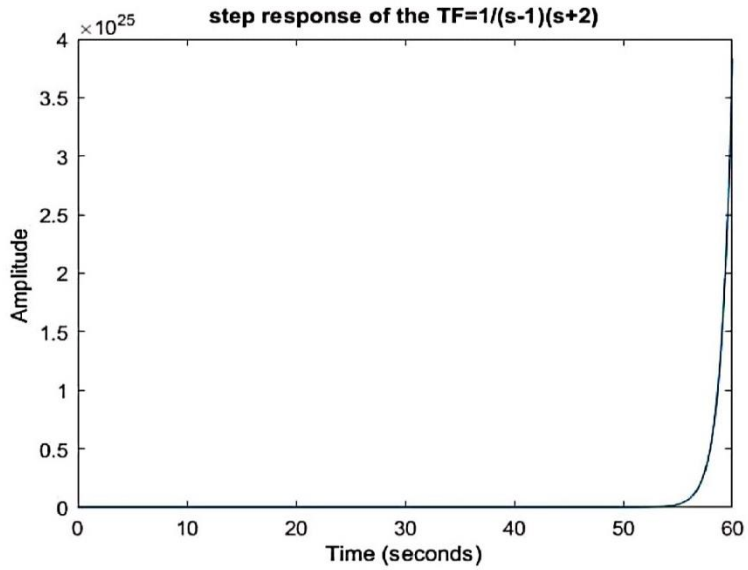
2.TF=1/(s-1)(s+2)

Open Loop response of transfer function 1/(s-1)(s+2)

```
p=[1];
q=[1 1 -2];
G=tf(p,q);
figure(8)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of the TF=1/(s-1)(s+2)');
```



closed loop response for unity negative feedback system of the transfer function 1/(s-1)(s+2)

```
p=[1];
q=[1 1 -2];
```

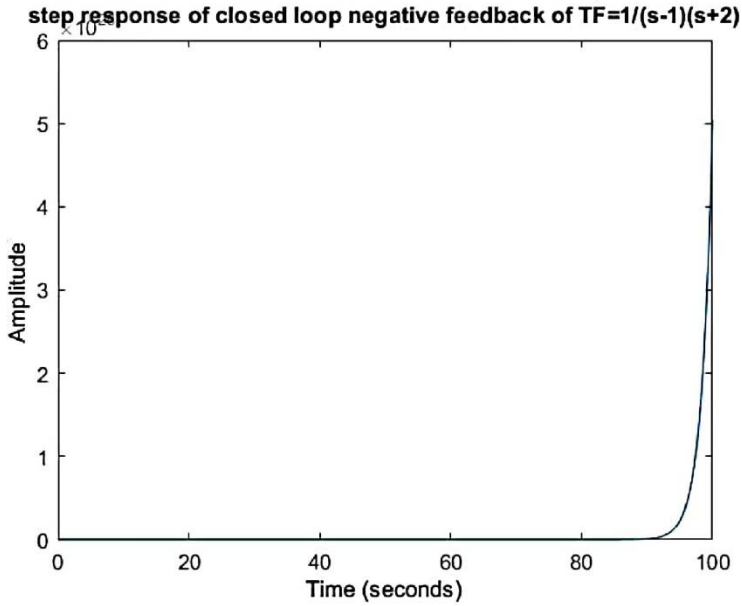
```
G=tf(p,q);
t=feedback(G,1)
```

```
t =

        1
  -----------
  s^2 + s - 1

Continuous-time transfer function.
```

```
figure(9)
step(t);
stepinfo(t)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title ('step response of closed loop negative feedback of TF=1/(s-1)(s+2)');
```



closed loop response for unity positive feedback system of the transfer function 1/(s-1)(s+2)

```
p=[1];
```

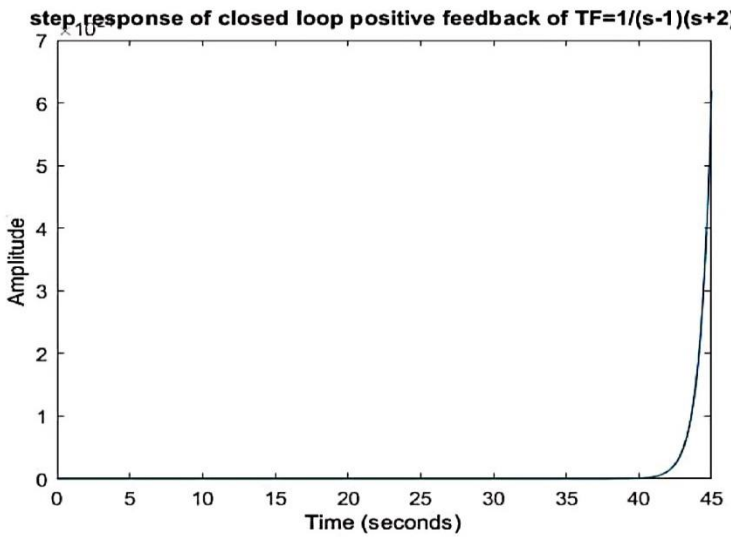
```
q=[1 1 -2];
G=tf(p,q);
t=feedback(G,-1)
```

```
t =

       1
  -----------
  s^2 + s - 3

Continuous-time transfer function.
```

```
figure(10)
step(t);
stepinfo(t)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of closed loop positive feedback of TF=1/(s-1)(s+2)');
```

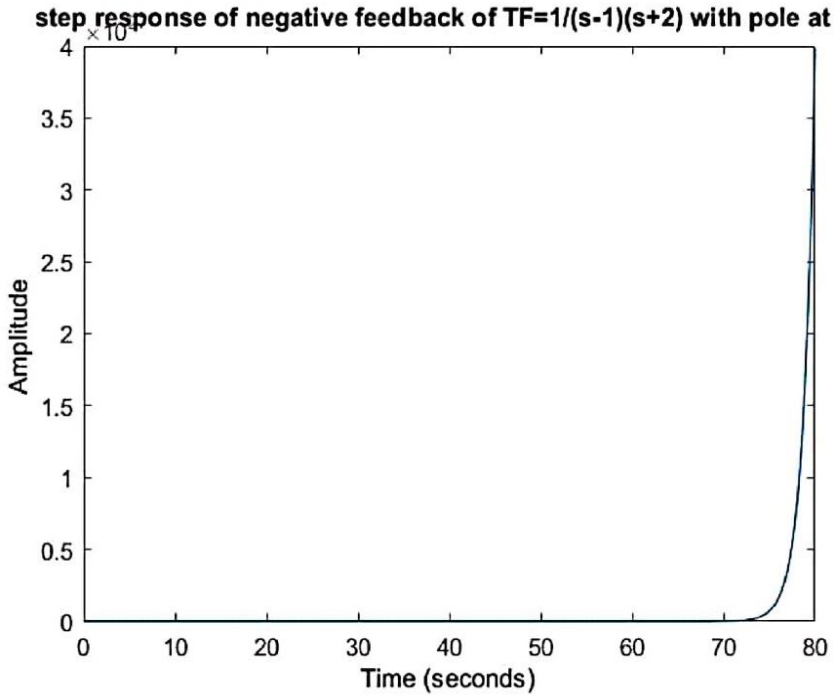


closed loop response for the TF=1/(s-1)(s+2) with pole at -1 in negative feedback path

```
p=[1 1];
q=[1 2 -1 -1];
G=tf(p,q);
figure(11)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of negative feedback of TF=1/(s-1)(s+2) with pole at -1');
```



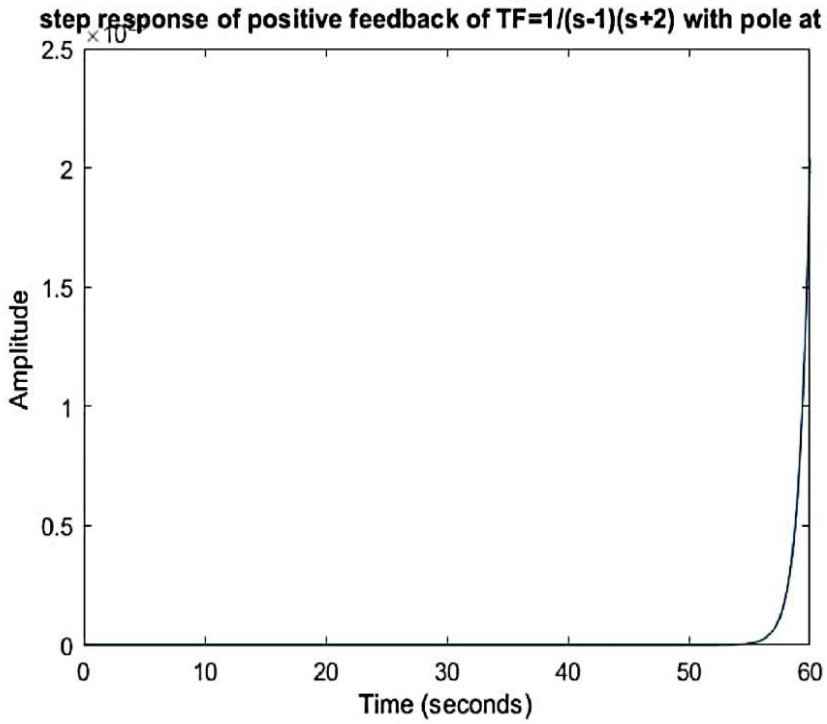
closed loop response for the TF=1/(s-1)(s+2) with pole at -1 in positive feedback path

```
p=[1 1];
q=[1 2 -1 -3];
G=tf(p,q);
figure(12)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
```

```
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of positive feedback of TF=1/(s-1)(s+2) with pole at -1');
```

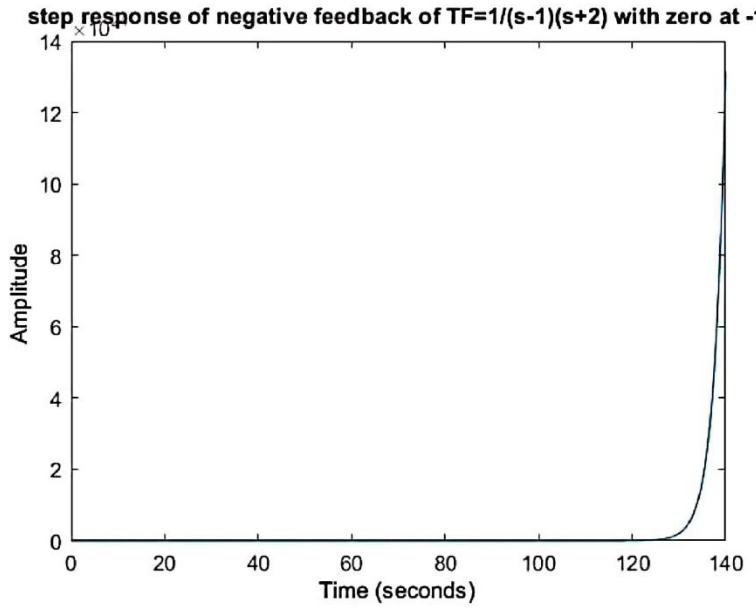


closed loop response for the TF=1/(s-1)(s+2) with zero at -1 in negative feedback path

```
p=[1];
q=[1 2 -1];
G=tf(p,q);
figure(13)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of negative feedback of TF=1/(s-1)(s+2) with zero at -1');
```

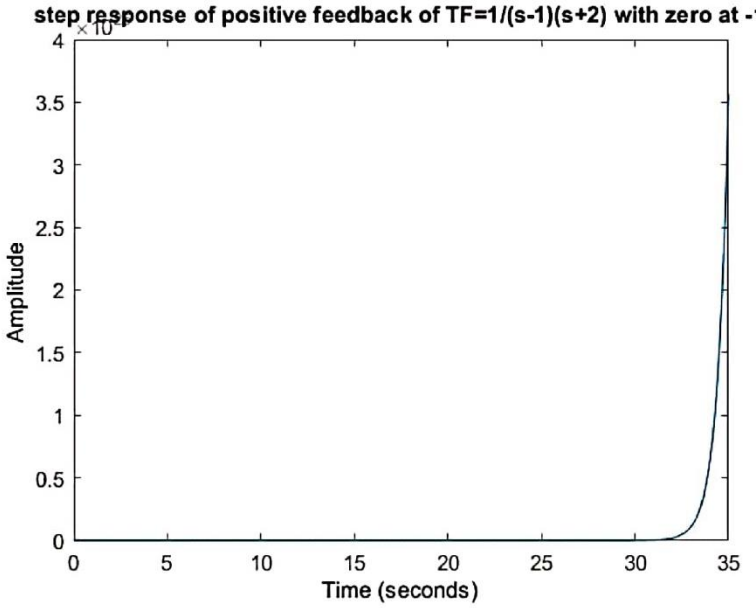


closed loop response for the TF=1/(s-1)(s+2) with zero at -1 in positive feedback path

```
p=[1];
q=[1 0 -3];
G=tf(p,q);
figure(14)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of positive feedback of TF=1/(s-1)(s+2) with zero at -1');
```



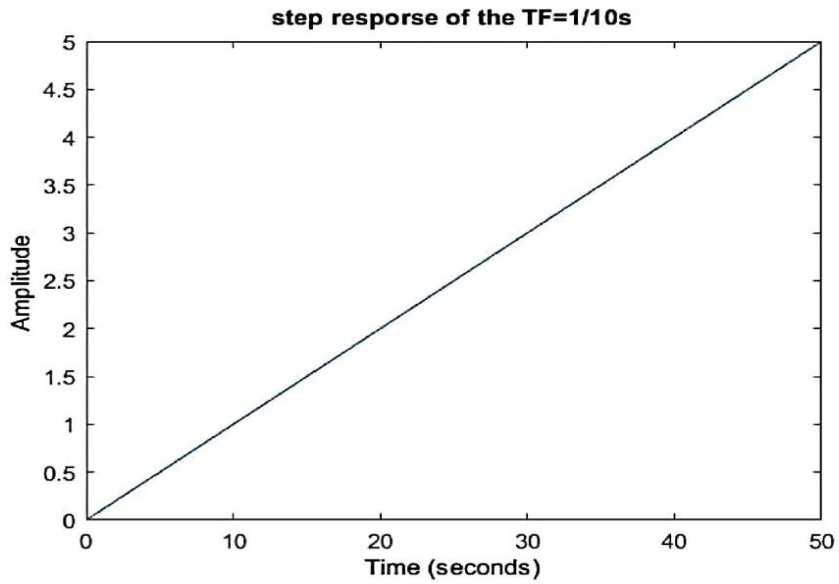
3.TF=1/10s

Open Loop response of transfer function 1/10s

```
p=[1];
q=[10 0];
G=tf(p,q);
figure(15)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
         RiseTime: NaN
     SettlingTime: NaN
      SettlingMin: NaN
      SettlingMax: NaN
        Overshoot: NaN
       Undershoot: NaN
             Peak: Inf
         PeakTime: Inf
```

```
title('step resporse of the TF=1/10s');
```



closed loop response for unity negative feedback system of the transfer function 1/10s

```
p=[1];
q=[10 0];
```

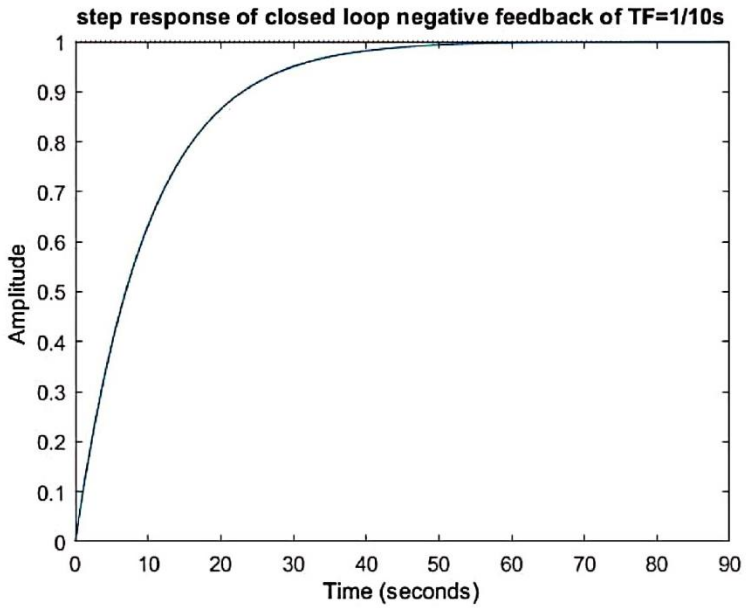
```
G=tf(p,q);
t=feedback(G,1)
```

```
t =

     1
  --------
  10 s + 1

Continuous-time transfer function.
```

```
figure(16)
step(t);
stepinfo(t)
```

```
ans = struct with fields:
         RiseTime: 21.9701
     SettlingTime: 39.1207
      SettlingMin: 0.9045
      SettlingMax: 1.0000
        Overshoot: 0
       Undershoot: 0
             Peak: 1.0000
         PeakTime: 105.4584
```

```
title('step response of closed loop negative feedback of TF=1/10s');
```



closed loop response for unity positive feedback system of the transfer function 1/10s

```
p=[1];
```

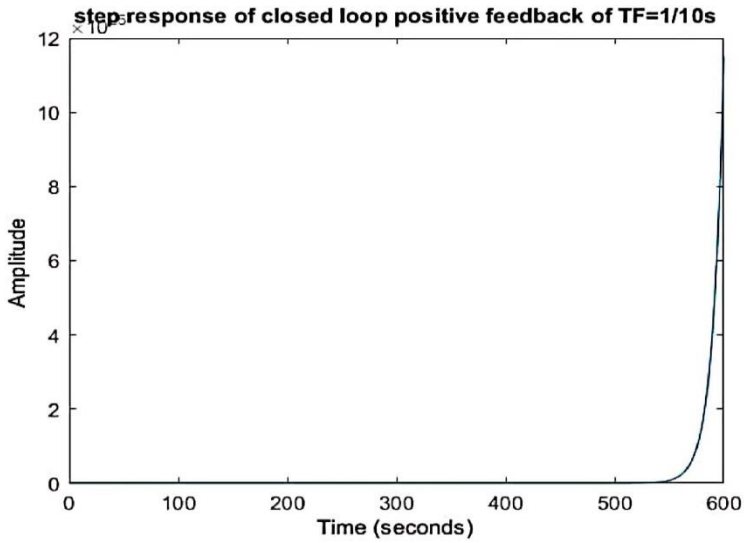
```
q=[10 0];
G=tf(p,q);
t=feedback(G,-1)
```

```
t =

     1
  --------
  10 s - 1

Continuous-time transfer function.
```

```
figure(17)
step(t);
stepinfo(t)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of closed loop positive feedback of TF=1/10s');
```

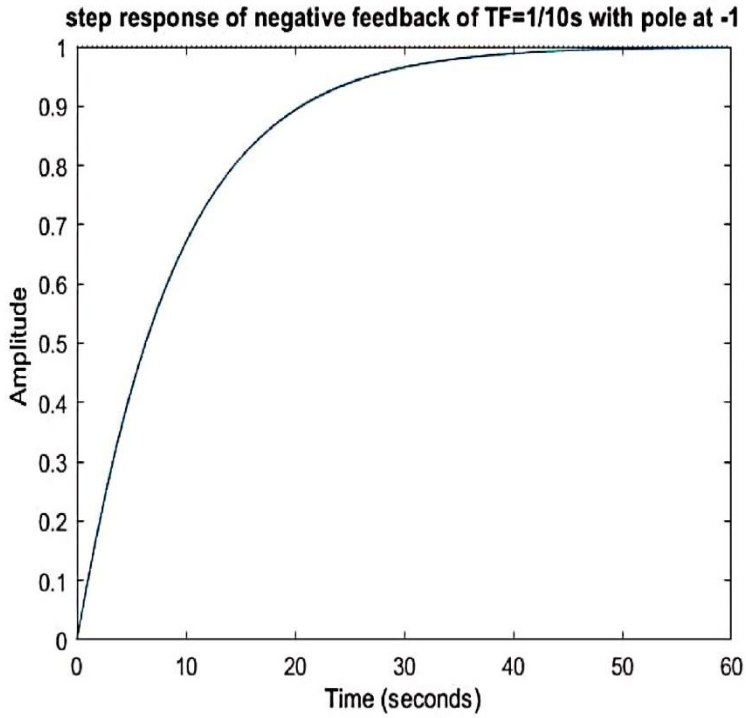


closed loop response for the TF=1/10s with pole at -1 in negative feedback path

```
p=[1 1];
q=[10 10 1];
G=tf(p,q);
figure(18)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: 19.5623
    SettlingTime: 34.8576
     SettlingMin: 0.9029
     SettlingMax: 0.9993
       Overshoot: 0
      Undershoot: 0
            Peak: 0.9993
        PeakTime: 64.9699
```

```
title('step response of negative feedback of TF=1/10s with pole at -1');
```



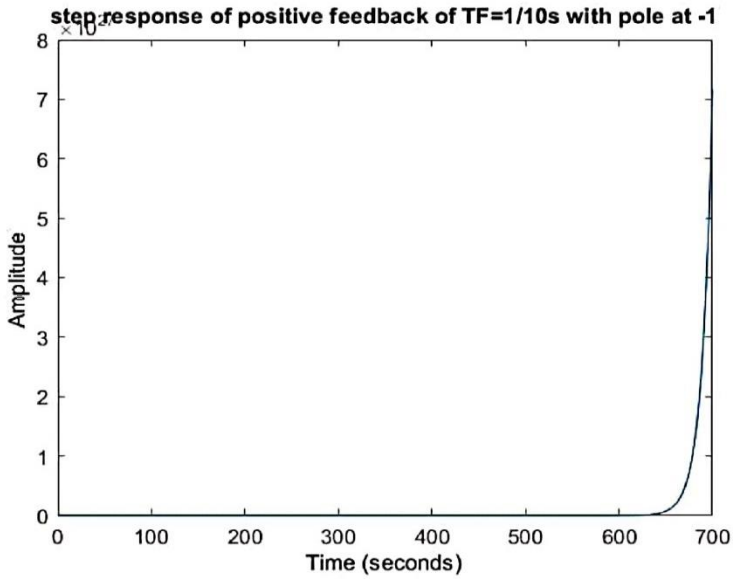
closed loop response for the TF=1/10s with pole at -1 in positive feedback path

```
p=[1 1];
q=[10 10 -1];
G=tf(p,q);
figure(19)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
```

```
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of positive feedback of TF=1/10s with pole at -1');
```

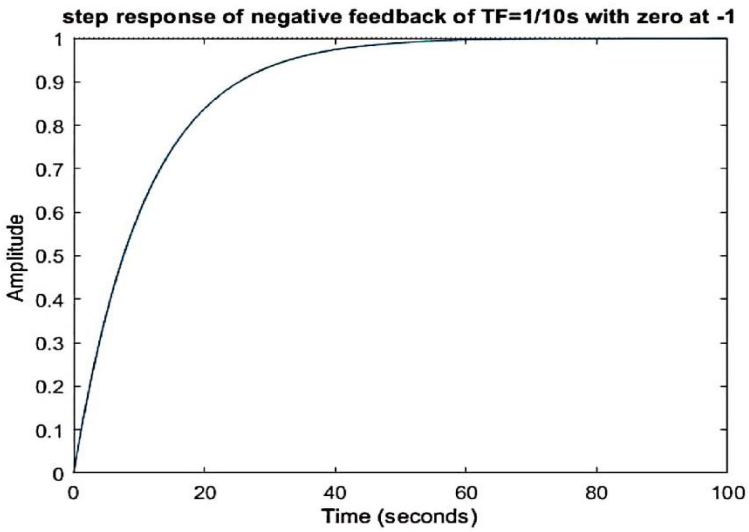


closed loop response for the TF=1/10s with zero at -1 in negative feedback path

```
p=[1];
q=[11 1];
G=tf(p,q);
figure(20)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: 24.1671
    SettlingTime: 43.0328
     SettlingMin: 0.9045
     SettlingMax: 1.0000
       Overshoot: 0
      Undershoot: 0
            Peak: 1.0000
        PeakTime: 116.0042
```

```
title('step response of negative feedback of TF=1/10s with zero at -1');
```

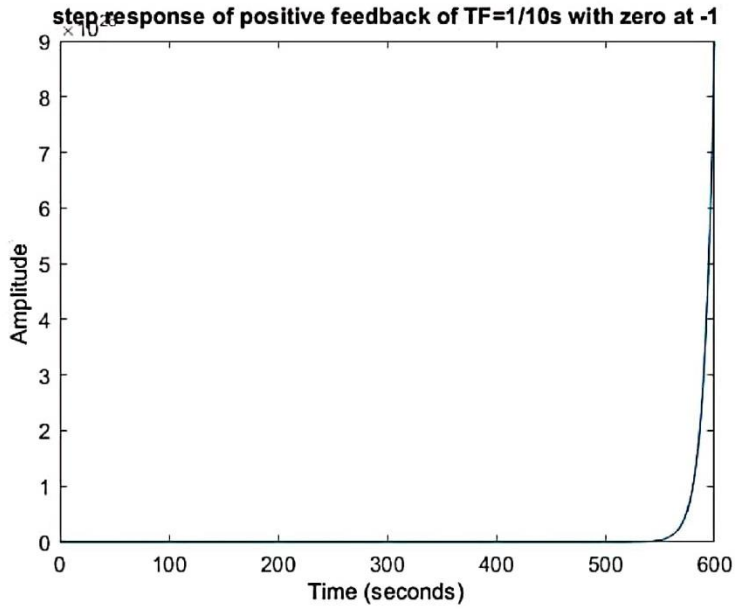


closed loop response for the TF=1/10s with zero at -1 in positive feedback path

```
p=[1];
q=[9 -1];
G=tf(p,q);
figure(21)
step(G);
stepinfo(G)
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
title('step response of positive feedback of TF=1/10s with zero at -1');
```



Result:

Experiment No.06

a) **Effect of open loop and zeroes on root locus contour**
b) **To estimate the effect of open loop gain on the transient response of closed loop system by using Root locus**
c) **Comparative study of Bode, Nyquist and Root locus with respect to Stability**

AIM: To Find the Effect of open loop and zeroes on root locus contour

Apparatus: PC loaded with MATLAB

Theory:

Root locus technique is used to find the roots of the characteristic's equation. This technique provides a graphical method of plotting the locus of the roots in the s plane as a given parameter usually gain is varied over the complete range of values. This method brings in to focus the complete dynamic response of the system. By using root locus method, the designer can predict the effects location of closed loop poles by varying the gain value or adding open loop poles and/or open loop zeroes. The closed loop poles are the roots of the characteristic equation.

Various terms related to root locus technique that we will use frequently in this article.

1. Symmetry of Root Locus: Root locus is symmetric about the x axis or the real axis.
2. Characteristic Equation Related to Root Locus Technique: 1 + G(s)H(s) = 0 is known as characteristic equation. Now on differentiating the characteristic equation and on equating dk/ds equals to zero, we can get break away points.
3. Break away Points: Suppose two root loci which start from pole and moves in opposite direction collide with each other such that after collision they start moving in different directions in the symmetrical way. Or the breakaway points at which multiple roots of the characteristic equation 1 + G(s)H(s)= 0 occur. The value of K is maximum at the points where the branches of root loci break away. Break away points may be real, imaginary or complex.
4. Break in Point: Condition of break in to be there on the plot is written below: Root locus must be present between two adjacent zeros on the real axis.
5. Centre of Gravity: It is also known centroid and is defined as the point on the plot from where all the asymptotes start. Mathematically, it is calculated by the difference of summation of poles and zeros in the transfer function when divided by the difference of total number of poles and total number of zeros. Centre of gravity is always real & it is denoted by σA. Where N is number of poles & M is number of zeros.

$$\sigma_A = \frac{(Sum\ of\ real\ parts\ of\ poles) - (Sum\ of\ real\ parts\ of\ zeros)}{N - M}$$

6. Asymptotes of Root Loci: Asymptote originates from the centre of gravity or centroid and goes to infinity at definite some angle. Asymptotes provide direction to the root locus when they depart break away points.
7. Angle of Asymptotes: Asymptotes makes some angle with the real axis and this angle can be calculated from $Angle\ of\ asymptotes = \frac{(2p+1) \times 180}{N - M}$ the given formula,
   Where p = 0, 1, 2 (N-M-1)
8. Angle of Arrival or Departure: We calculate angle of departure when there exists complex poles in the system. Angle of departure can be calculated as 180- {(sum of angles to a complex pole from the other poles)- (sum of angle to a complex pole from the zeros)}.

9. Intersection of Root Locus with the Imaginary Axis: In order to find out the point of intersection root locus with imaginary axis, we have to use Routh Hurwitz criterion. First, we find the auxiliary equation then the corresponding value of K will give the value of the point of intersection.

Procedure:

1.open the MATLAB and create a new file and open it

2.Run the code by entering it and by entering root locus

3.Repeat it by adding poles and zeroes

4.Observe the effect of pols and zeroes on root locus contour

Transfer fuction: $\frac{36}{s^3+6s^2+11s+6}$

Theoretical Calculation:

Program:

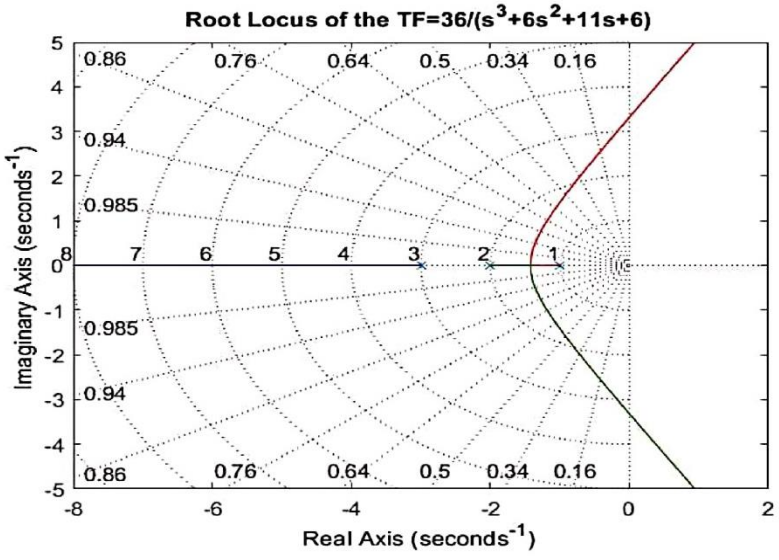
To obtain Root Locus of a Transfer Function

Root Locus of the TF=36/(s^3+6s^2+11s+6)

```
p=[36];
q=[1 6 11 6];
sys1=tf(p,q);
figure(1)
zpk(sys1)
```

```
ans =

          36
  -----------------
  (s+3) (s+2) (s+1)

Continuous-time zero/pole/gain model.
```

```
rlocus(sys1)
grid
title('Root Locus of the TF=36/(s^3+6s^2+11s+6)')
```



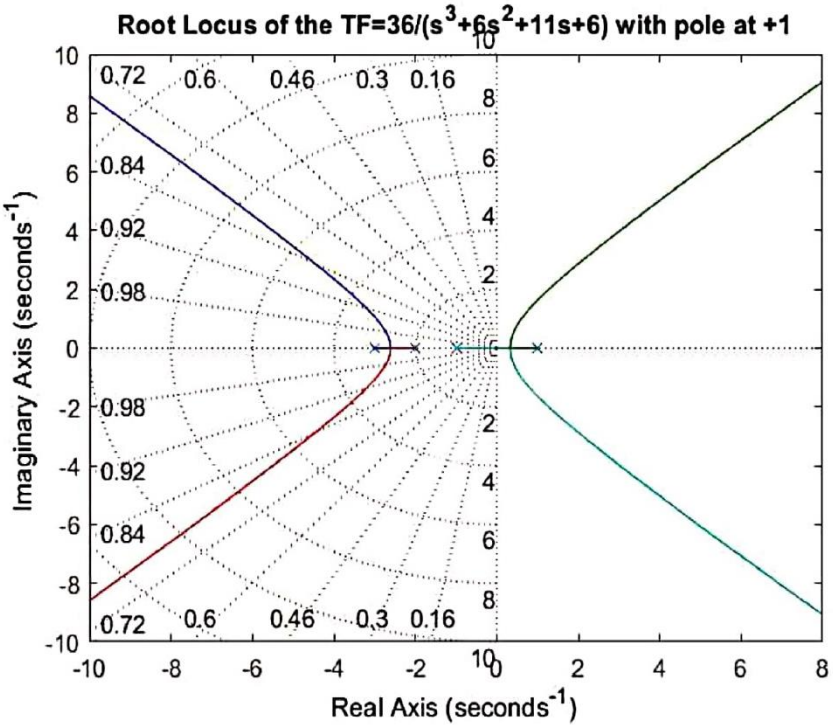
Root Locus of the TF=36/(s^3+6s^2+11s+6) with pole at +1

```
p=[36];
q=[1 5 5 -5 -6];
sys2=tf(p,q);
figure(2)
```

```
zpk(sys2)
```

```
ans =

            36
  ----------------------
  (s+3) (s+2) (s+1) (s-1)

Continuous-time zero/pole/gain model.
```

```
rlocus(sys2)
grid
title('Root Locus of the TF=36/(s^3+6s^2+11s+6) with pole at +1')
```



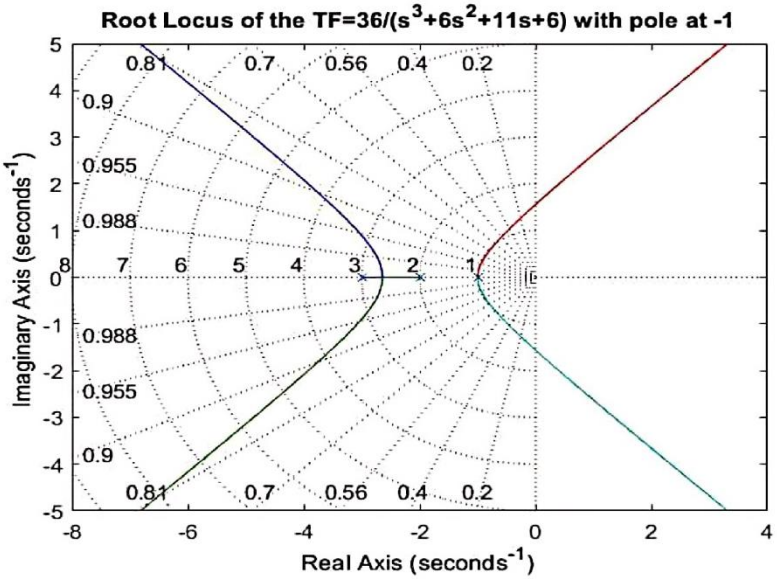
Root Locus of the TF=36/(s^3+6s^2+11s+6) with pole at -1

```
p=[36];
q=[1 7 17 17 6];
sys3=tf(p,q);
figure(3)
zpk(sys3)
```

```
ans =

          36
  -------------------
  (s+3) (s+2) (s+1)^2

Continuous-time zero/pole/gain model.
```

```
rlocus(sys3)
```

```
grid
title('Root Locus of the TF=36/(s^3+6s^2+11s+6) with pole at -1')
```

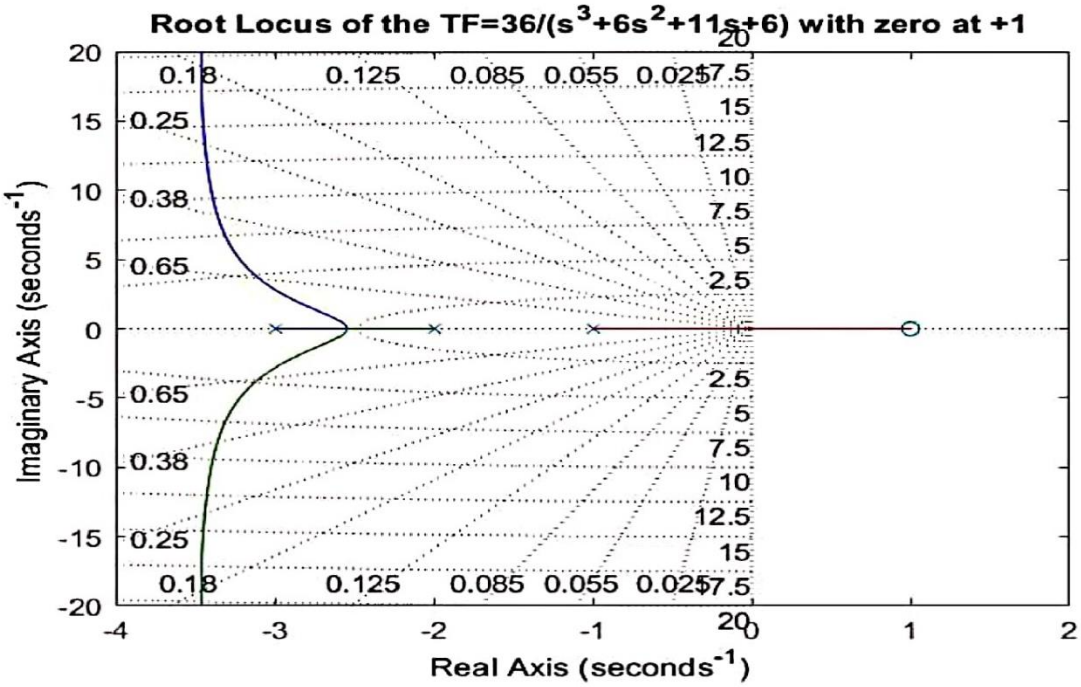


Root Locus of the TF=36/(s^3+6s^2+11s+6) with zero at +1

```
p=[36 -36];
q=[1 6 11 6];
sys4=tf(p,q);
figure(4)
zpk(sys4)
```

```
ans =

      36 (s-1)
  ---------------
  (s+3) (s+2) (s+1)

Continuous-time zero/pole/gain model.
```

```
rlocus(sys4)
grid
title('Root Locus of the TF=36/(s^3+6s^2+11s+6) with zero at +1')
```

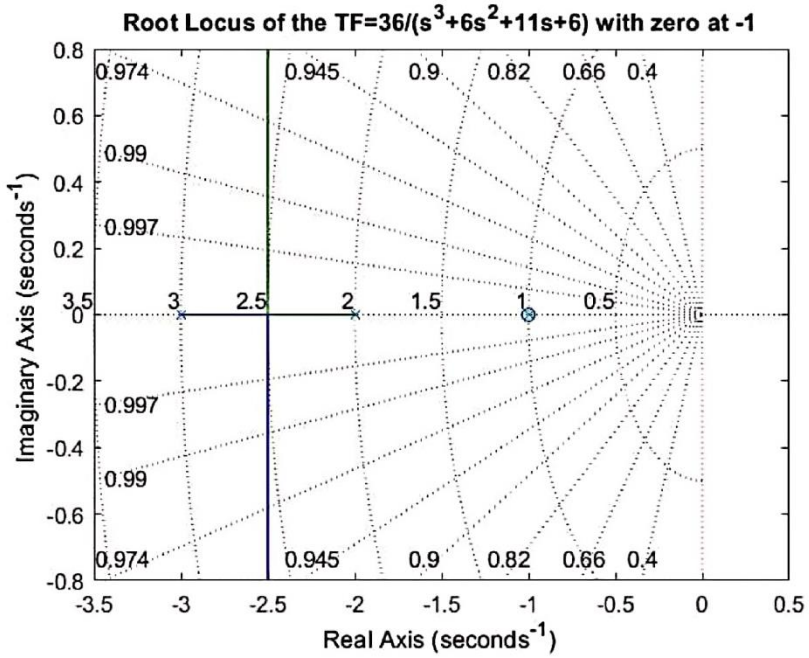


Root Locus of the TF=36/(s^3+6s^2+11s+6) with zero at -1

```
p=[36 36];
q=[1 6 11 6];
sys5=tf(p,q);
figure(5)
zpk(sys5)
```

```
ans =

      36 (s+1)
  ----------------
  (s+3) (s+2) (s+1)

Continuous-time zero/pole/gain model.
```

```
rlocus(sys5)
grid
title('Root Locus of the TF=36/(s^3+6s^2+11s+6) with zero at -1')
```



Experiment No: 6(B)

To estimate the effect of open loop gain on the transient response of closed loop system by using Root locus.

AIM: To estimate the effect of open loop gain on the transient response of closed loop system by using Root locus

Apparatus: PC loaded with MATLAB

Procedure:

1. Open MATLAB Command window
2. Click on new script file and enter the code
3. Obtain root locus plots for different values of K and compare them
4. Observe how the open loop gain effects the transient response of the closed loop system
5. Save the waveforms and exit the MATLAB window

Effect of loop gain on closed loop system by using Root Locus

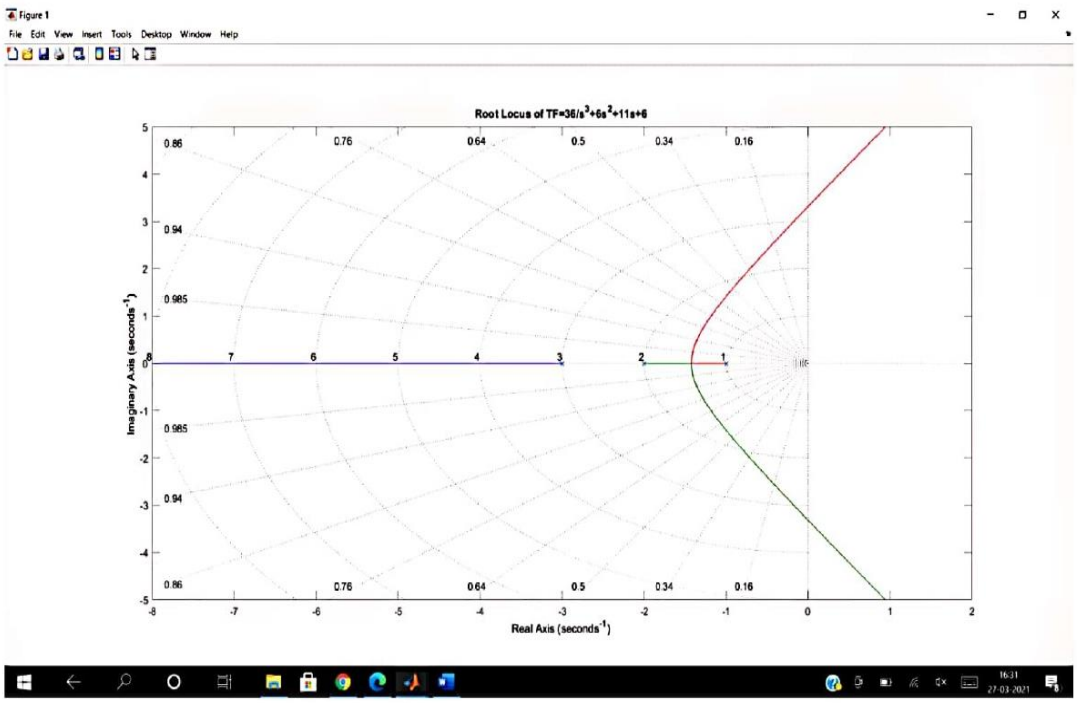
Transfer Function G(s)=k/s^3+6s^2+11s+6

```
p=[36];
q=[1,6,11,6];
sys=tf(p,q);
figure(1);
zpk(sys);
rlocus(sys);
grid
title('Root Locus of TF=36/s^3+6s^2+11s+6');
```

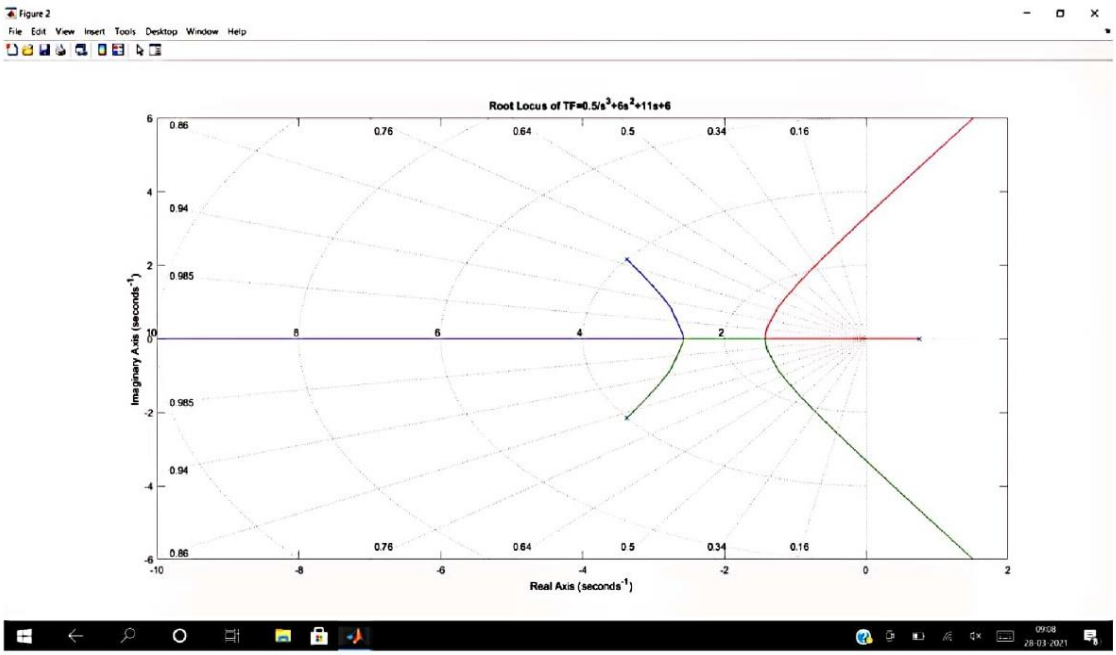
For Different Values of k

```
for i=1:4
    ki=input('enter k value');
    gi=feedback(sys*ki,-1);
    figure(i+1)
    zpk(gi);
    rlocus(gi);
    title('Root Locus of TF=36/s^3+6s^2+11s+6 for k=');
end
```

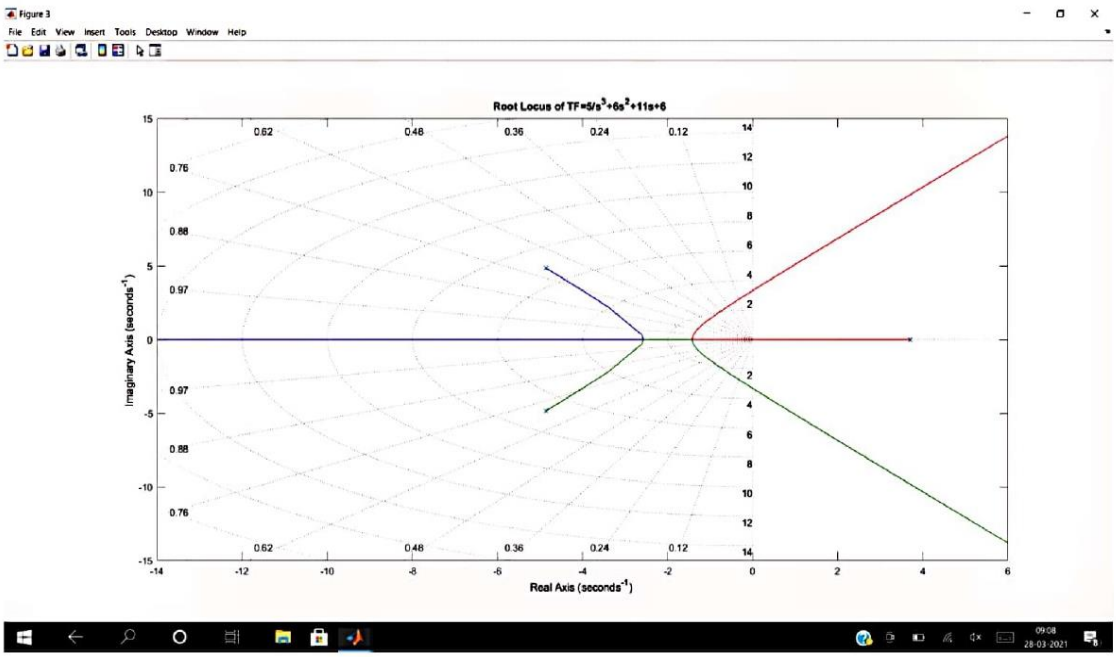
For k=1:



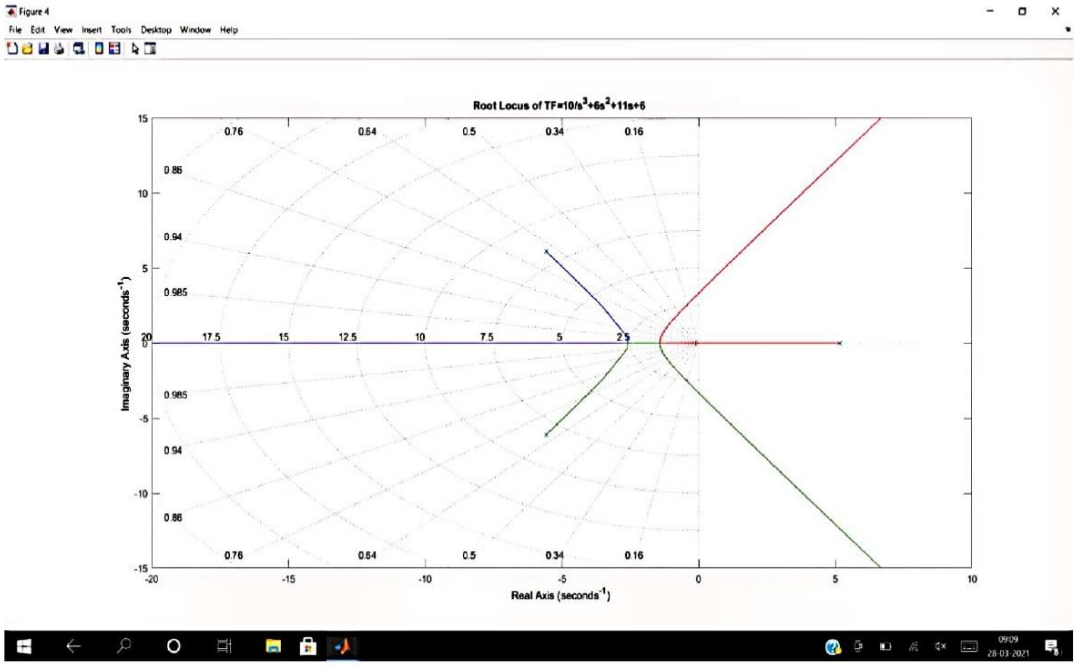
For k=0.5:



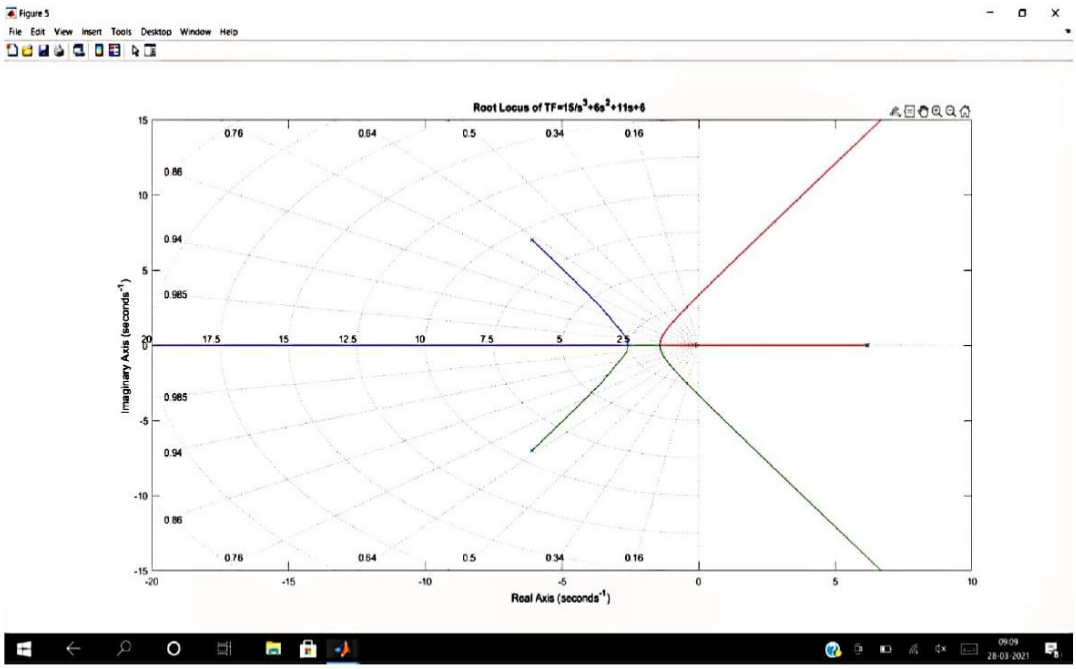
For k=5:



For k=10:



For k=15:



Experiment No: 6(C)

Comparative study of Bode, Nyquist and Root locus with respect to Stability

AIM: Comparative study of Bode, Nyquist and Root locus with respect to Stability

Theory:

One of the most useful representation of transfer function is a logarithmic plot which consists of two graphs, one giving the logarithm of [G (jw)] and the other phase angle of G(jw) both plotted against frequency in logarithmic scale. These plots are called bode plots. The main advantage of using bode diagram is that the multiplication of magnitudes can be converted into addition. Bode plots are a good alternative to the Nyquist plots.

Frequency response specifications:

1. Gain cross over frequency wgc = It is the frequency at which magnitude of G (jw) H (jw) is unity i.e. 1.
2. Phase cross over frequency wpc = It is the frequency at which phase angle of G(jw) H(jw) is - 180^o
3. Gain margin G.M = It is defined as the margin in gain allowable by which gain can be increased till system reaches on the verge of instability. Mathematically it is defined as the reciprocal of the magnitude of the G(jw) H(jw) measured at phase cross over frequency.
4. Phase margin P.M = Amount of additional phase lag which can be introduced in the system till system reaches on the verge of instability. Mathematically it can be defined as

$$P.M = 180^o + \angle G(jw)H(jw) \text{ at } w=wgc$$

Theoretical Calculation:

Procedure:

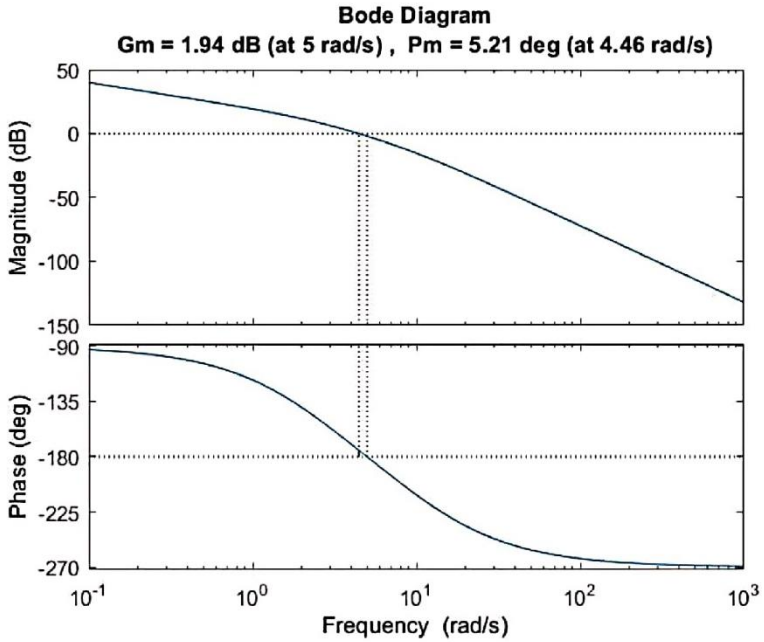
1. Open MATLAB command window and open new script file
2. Enter the code for different transfer functions to plot bode plot.
3. Tabulate the observations gain margins, phase margin, and different observations for different plots.
4. Compare the stability between the different tables and exit from MATLAB

1. Transfer fuction: $\frac{10}{s(1+0.4s)(1+0.1s)}$
2. Transfer fuction: $\frac{36}{s^3+6s^2+11s+6}$

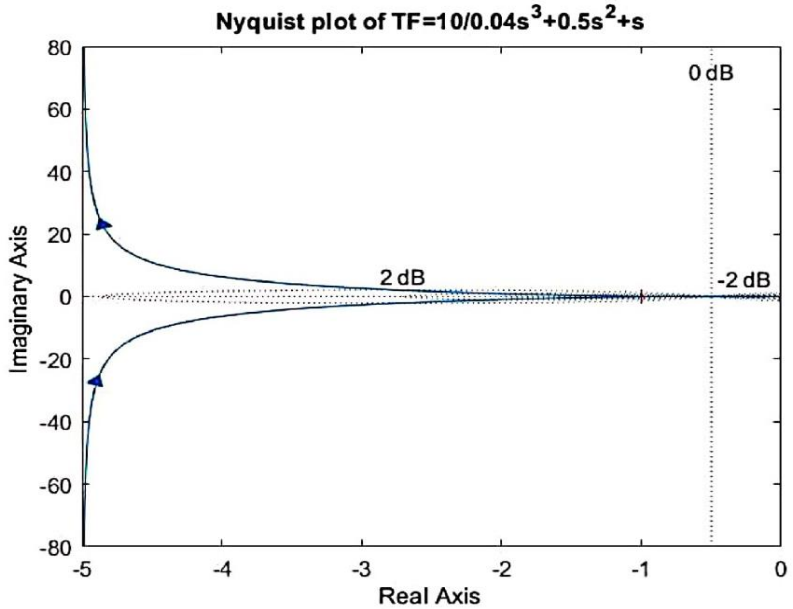
Study of Bode,Nyquist and Root Locus on stability

1.TF=10/0.04s^3+0.5s^2+s

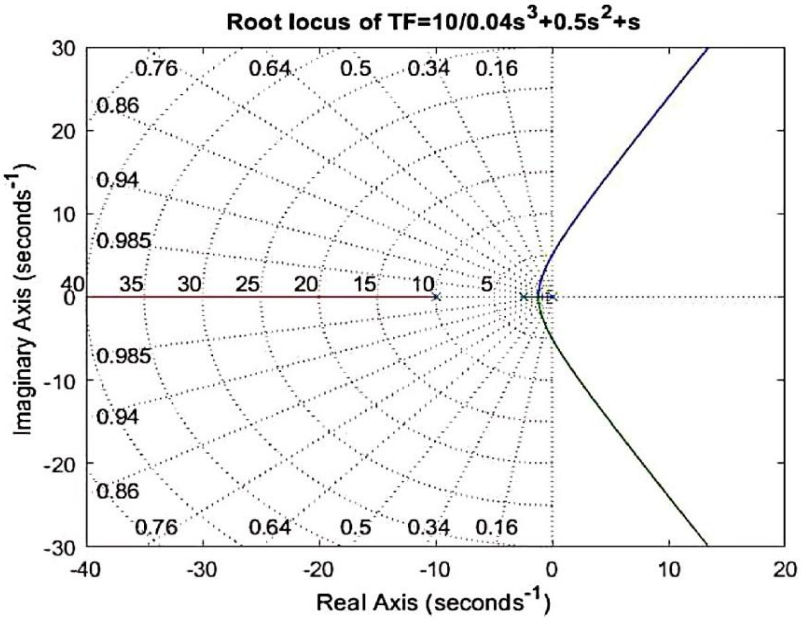
```
num=[10];
den=[0.04,0.5,1,0];
g1=tf(num,den);
figure(5)
margin(g1);
```




```
figure(6);
nyquist(g1);
grid
title('Nyquist plot of TF=10/0.04s^3+0.5s^2+s');
```

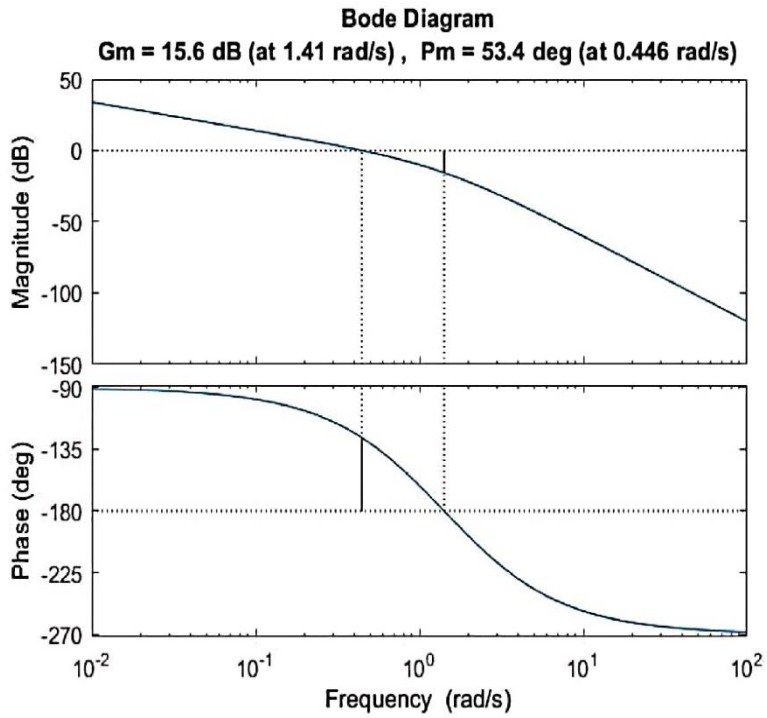


```
figure(7);
rlocus(g1);
grid
title('Root locus of TF=10/0.04s^3+0.5s^2+s');
```

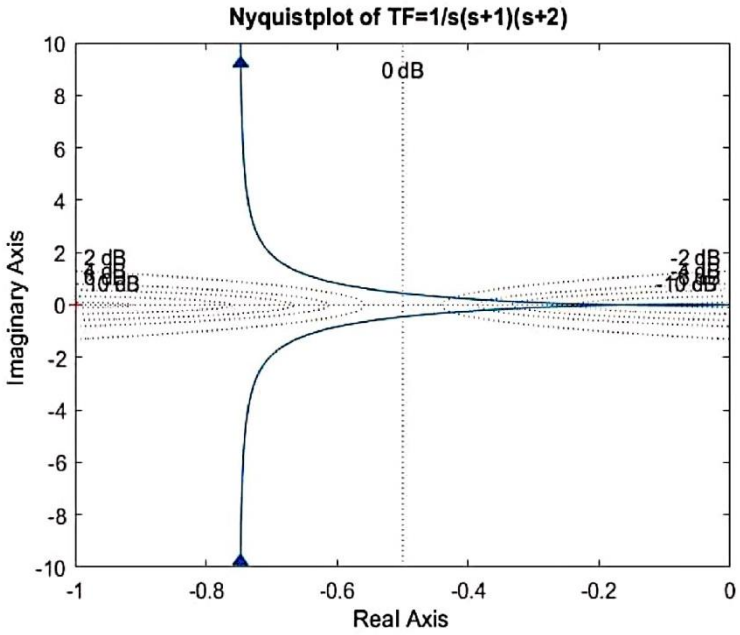


2.TF=1/s(s+1)(s+2)

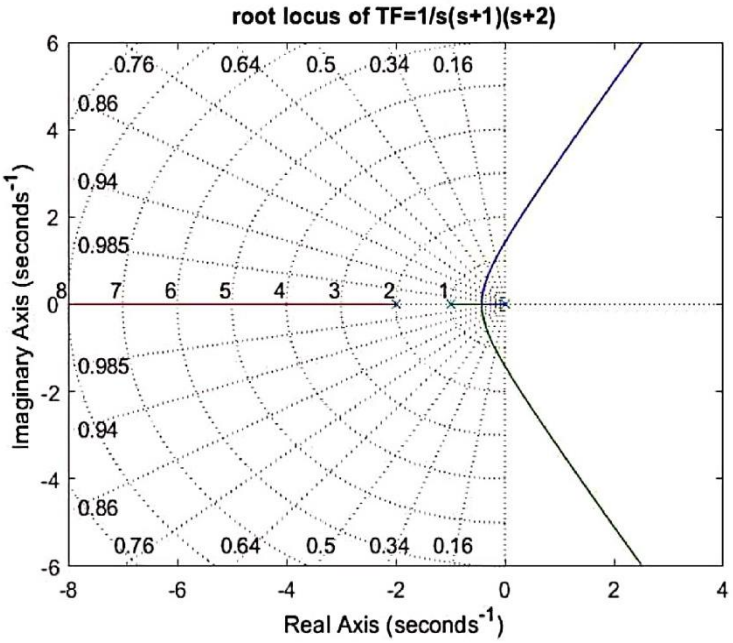
```
num=[1];
den=[1 3 2 0];
g3=tf(num,den);
figure(11);
margin(g3);
```




```
figure(12);
nyquist(g3);
grid
title('Nyquistplot of TF=1/s(s+1)(s+2) ');
```

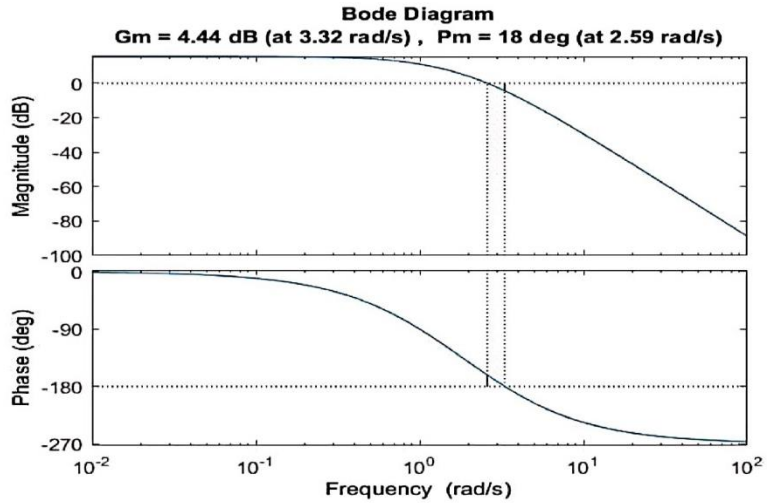



```
figure(13);
rlocus(g3);
grid
title('root locus of TF=1/s(s+1)(s+2)');
```

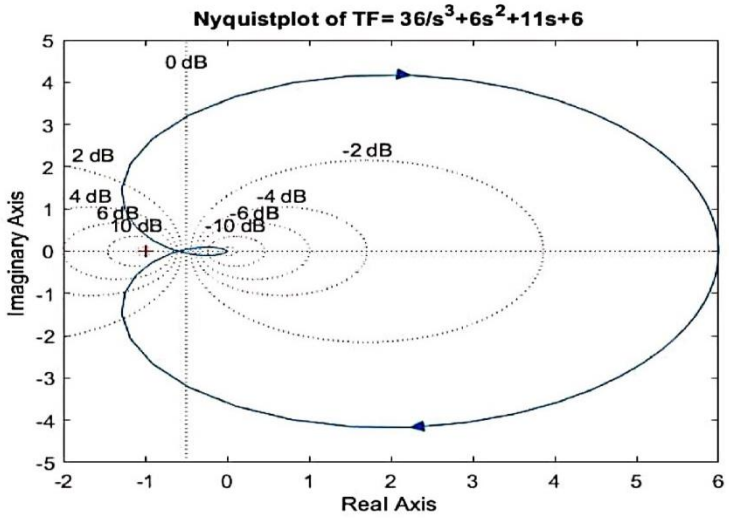


3.TF=36/s^3+6s^2+11s+6

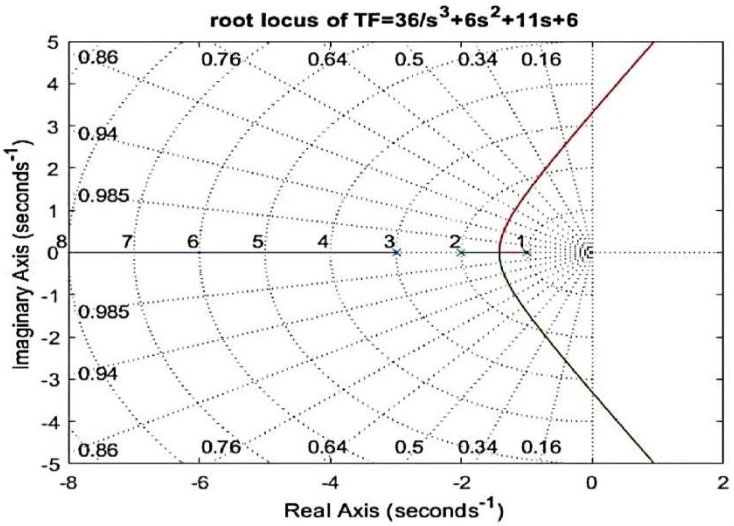
```
num=[36];
den=[1,6,11,6];
g2=tf(num,den);
figure(8);
margin(g2);
```




```
figure(9);
nyquist(g2);
grid
title('Nyquistplot of TF= 36/s^3+6s^2+11s+6');
```




```
figure(10);
rlocus(g2);
grid
title('root locus of TF=36/s^3+6s^2+11s+6');
```



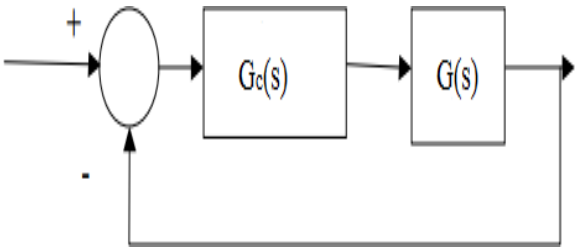
Experiment No.07

Design and study of lag, lead and Lag-lead compensator networks

AIM: To Design and study of lag, lead and Lag-lead compensator networks.

Apparatus: PC with MATLAB software.

Theory:



Procedure:

1. Consider uncompensated system G(s).
2. Design the lead, lag and lag-lead compensators for given specifications.
3. Plot the time response and frequency responses for compensated and uncompensated systems.
4. Compare the theoretical values with practical values.

Design of Lead Compensator:

Design a lead compensator for a unity feedback system with an open-loop transfer function

$$G(s) = \frac{K}{s(s+1)}$$

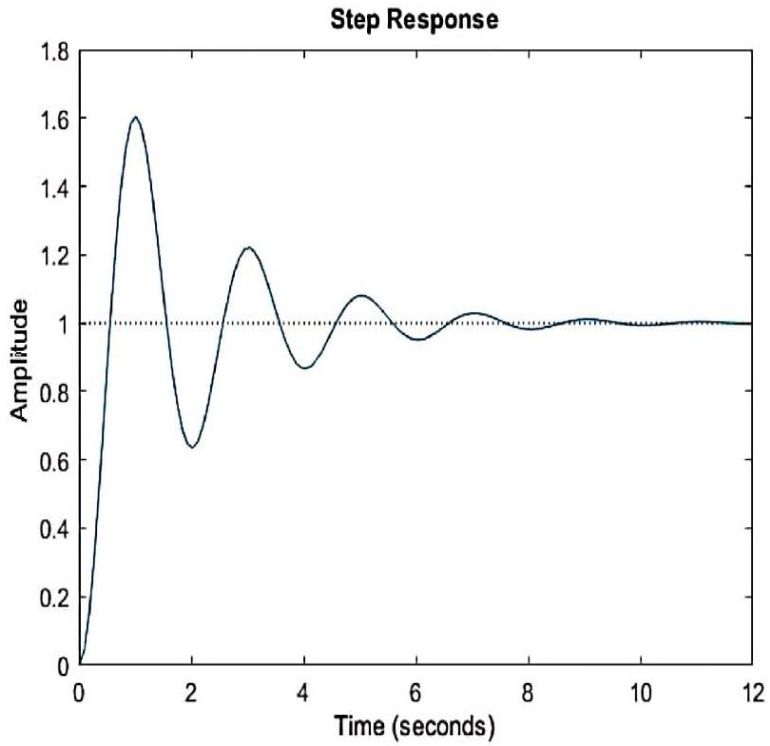
For the specifications of Kv = 10 s^{-1} and φ = 35°

Lead compensator

```
clc;
clear;
close all;

s=tf('s');
G_F=10/(s^2+s)
```

```
G_F =

    10
  -------
  s^2 + s

Continuous-time transfer function.
```

```
figure(1)
step(feedback(G_F,1))
```

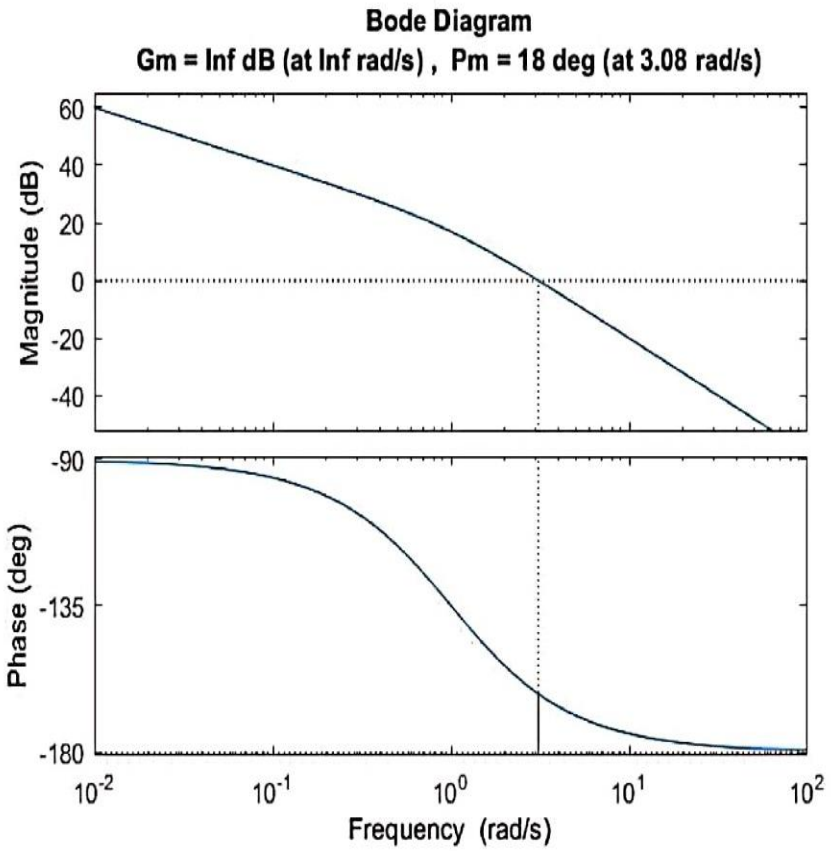



```
stepinfo(feedback(G_F,1))
```

```
ans = struct with fields:
        RiseTime: 0.3738
    SettlingTime: 7.3148
     SettlingMin: 0.6347
     SettlingMax: 1.6045
       Overshoot: 60.4530
```

```
      Undershoot: 0
            Peak: 1.6045
        PeakTime: 1.0131
```

```
figure(2)
margin(G_F)
```




```
[m_r,w_r]=getPeakGain(feedback(G_F,1))
```

```
m_r = 3.2026
w_r = 3.0818
```

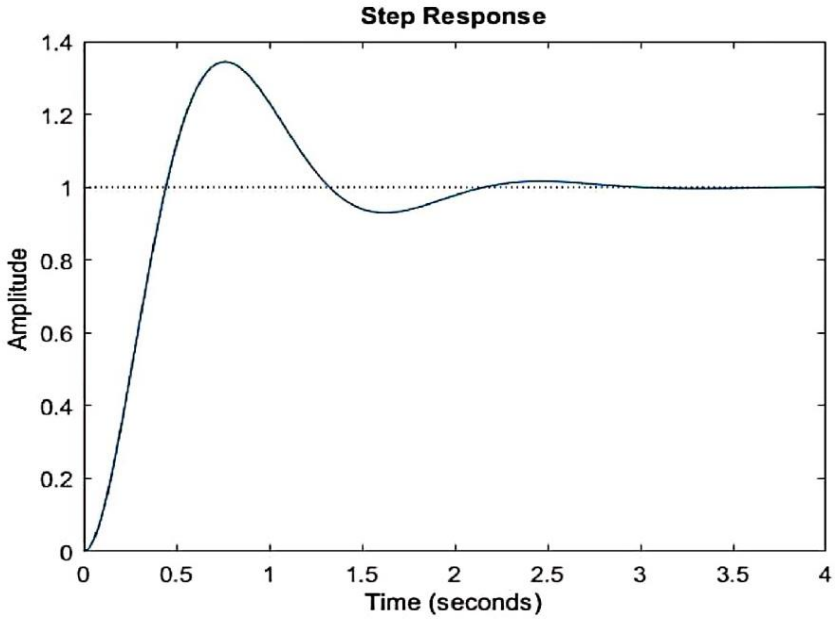
```
G_C=(1+0.385*s)/(1+0.162*s)
```

```
G_C =

  0.385 s + 1
  -----------
  0.162 s + 1

Continuous-time transfer function.
```

```
G=G_F*G_C
```

```
G =

         3.85 s + 10
  ------------------------
  0.162 s^3 + 1.162 s^2 + s

Continuous-time transfer function.
```

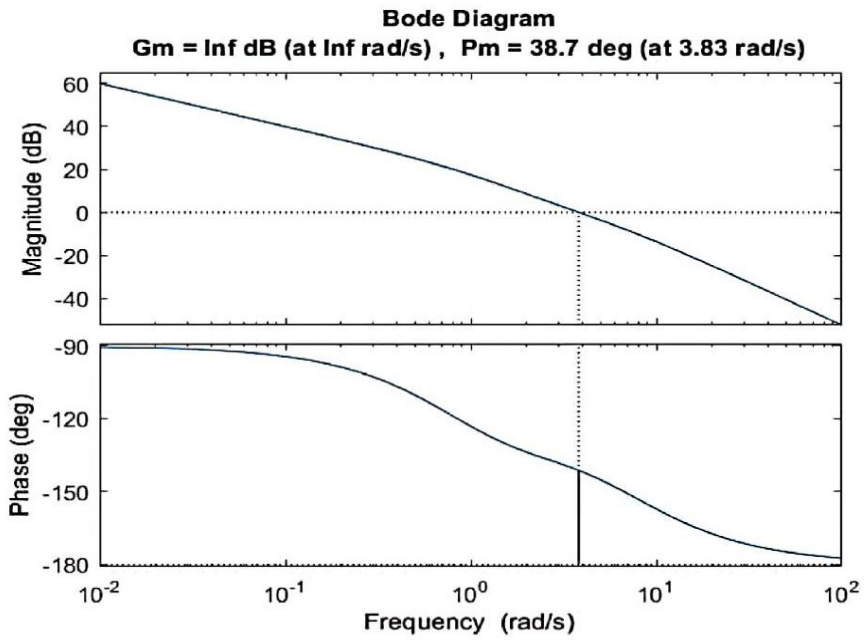
```
figure(3)
step(feedback(G,1))
```




```
stepinfo(feedback(G,1))
```

```
ans = struct with fields:
        RiseTime: 0.3020
    SettlingTime: 2.0151
     SettlingMin: 0.9164
     SettlingMax: 1.3445
       Overshoot: 34.4478
      Undershoot: 0
            Peak: 1.3445
        PeakTime: 0.7644
```

```
figure(4)
margin(G)
```




```
[m_r_cont,w_r_cont]=getPeakGain(feedback(G,1))
```

```
m_r_cont = 1.5418
w_r_cont = 3.4600
```

Design of Lag Compensator:

Compensate the system with the open – loop transfer function

$$G(s) = \frac{K}{s(s+1)(s+5)}$$

To meet the following specifications:

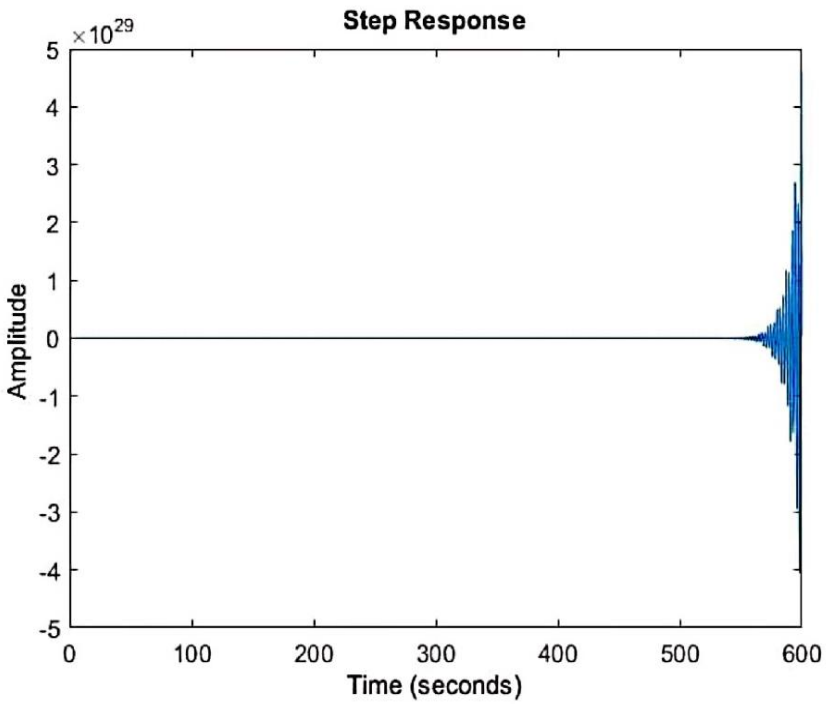
- **Damping Ratio g = 0.3**
- **Settling time t_s = 12s**
- **Velocity error constant $K_v \geq 8\ s^{-1}$**

Lag compensator

```
clc;
clear;
close all;

s=tf('s');
G_F=8/((s)*(s+1)*(1+0.2*s))
```

```
G_F =

            8
  ---------------------
  0.2 s^3 + 1.2 s^2 + s

Continuous-time transfer function.
```

```
figure(1)
step(feedback(G_F,1))
```

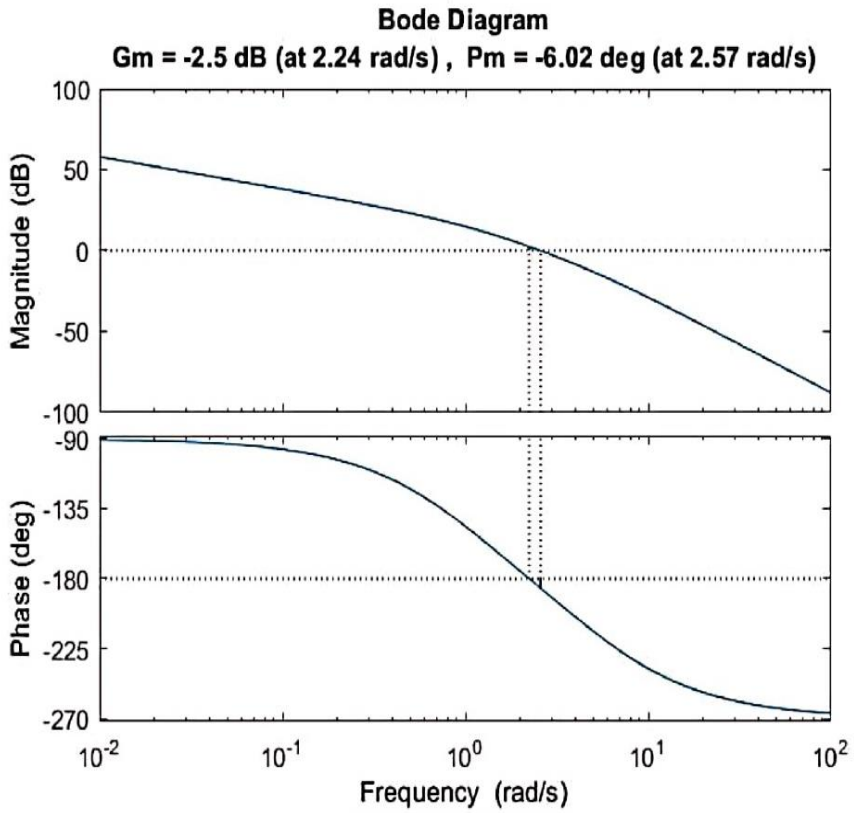



```
stepinfo(feedback(G_F,1))
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
```

```
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
figure(2)
margin(G_F)
```




```
[m_r,w_r]=getPeakGain(feedback(G_F,1))
```

```
m_r = 10.2932
w_r = 2.5284
```

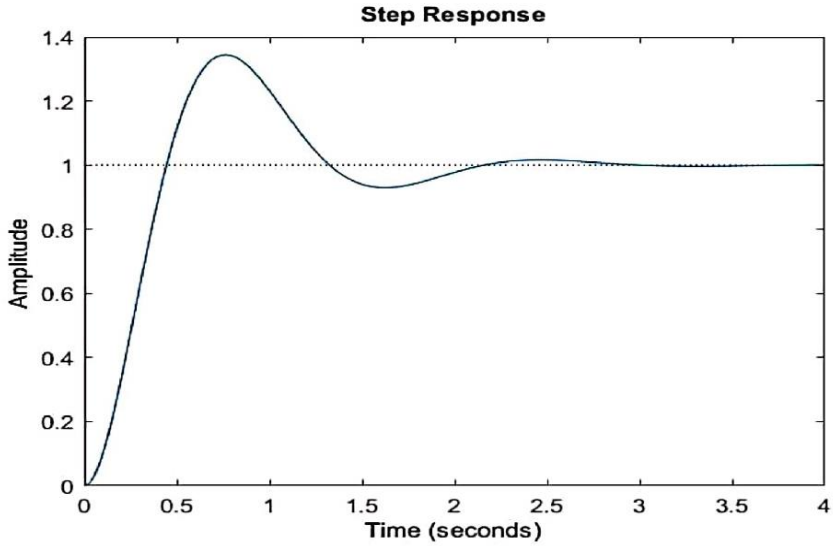
```
G_C=(1+5*s)/(1+40*s)
```

```
G_C =

  5 s + 1
  --------
  40 s + 1

Continuous-time transfer function.
```

```
G=G_F*G_C
```

```
G =

              40 s + 8
  -------------------------------
  8 s^4 + 48.2 s^3 + 41.2 s^2 + s

Continuous-time transfer function.
```

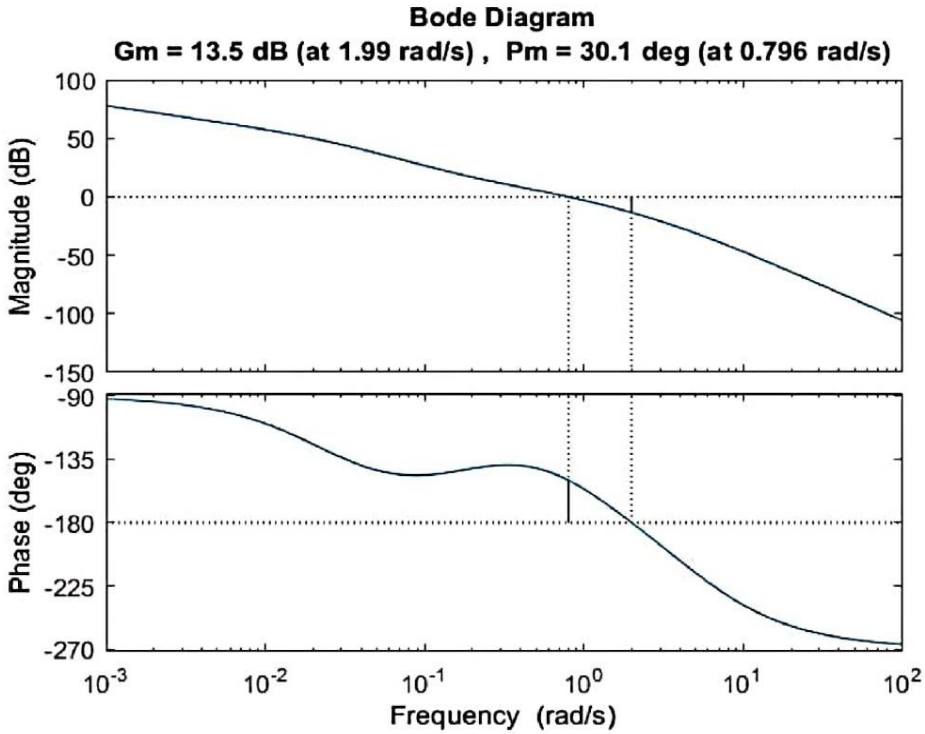
```
figure(3)
step(feedback(G,1))
```




```
stepinfo(feedback(G,1))
```

```
ans = struct with fields:
        RiseTime: 0.3020
    SettlingTime: 2.0151
     SettlingMin: 0.9164
     SettlingMax: 1.3445
       Overshoot: 34.4478
      Undershoot: 0
            Peak: 1.3445
        PeakTime: 0.7644
```

```
figure(4)
margin(G)
```




```
[m_r_cont,w_r_cont]=getPeakGain(feedback(G,1))
```

```
m_r_cont = 1.9250
w_r_cont = 0.8062
```

Design of Lag – Lead Compensator:

The open – loop transfer function of a unity feedback control system is a given by

$$G(s) = \frac{K}{s(1+0.5s)(1+0.1s)}$$

Compensate the system to meet the following specifications:

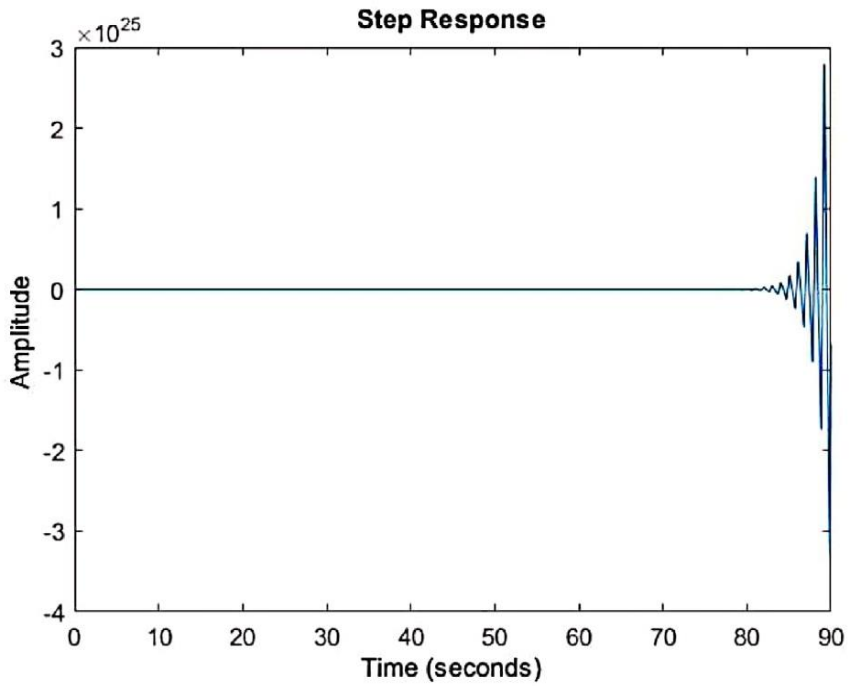
- **Velocity error constant $K_v \geq 25\ s^{-1}$**
- **Phase Margin $\varphi_s \geq 60^{o}$**
- **Band width $w_b = 10$ rad/s**

Lag-Lead compensator

```
clc;
clear;
close all;

s=tf('s');
G_F=(25)/((s)*(1+0.5*s)*(1+0.1*s))
```

```
G_F =

            25
  ----------------------
  0.05 s^3 + 0.6 s^2 + s

Continuous-time transfer function.
```

```
figure(1)
step(feedback(G_F,1))
```

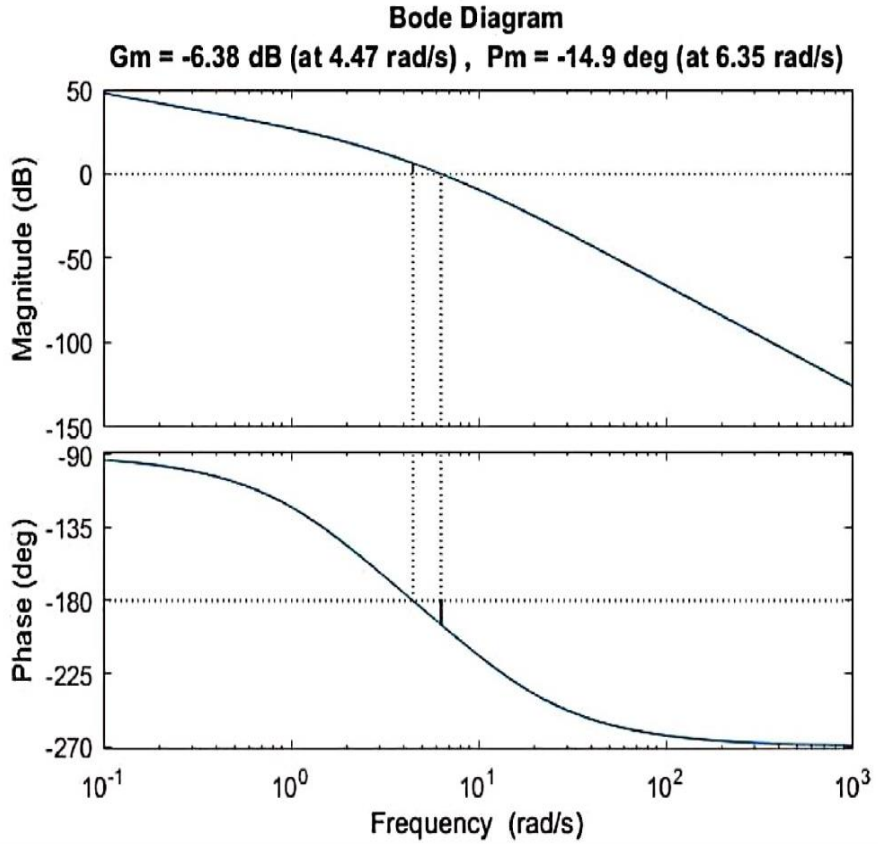



```
stepinfo(feedback(G_F,1))
```

```
ans = struct with fields:
        RiseTime: NaN
    SettlingTime: NaN
     SettlingMin: NaN
     SettlingMax: NaN
       Overshoot: NaN
```

```
      Undershoot: NaN
            Peak: Inf
        PeakTime: Inf
```

```
figure(2)
margin(G_F)
```




```
[m_r,w_r]=getPeakGain(feedback(G_F,1))
```

```
m_r = 4.2592
w_r = 6.0439
```

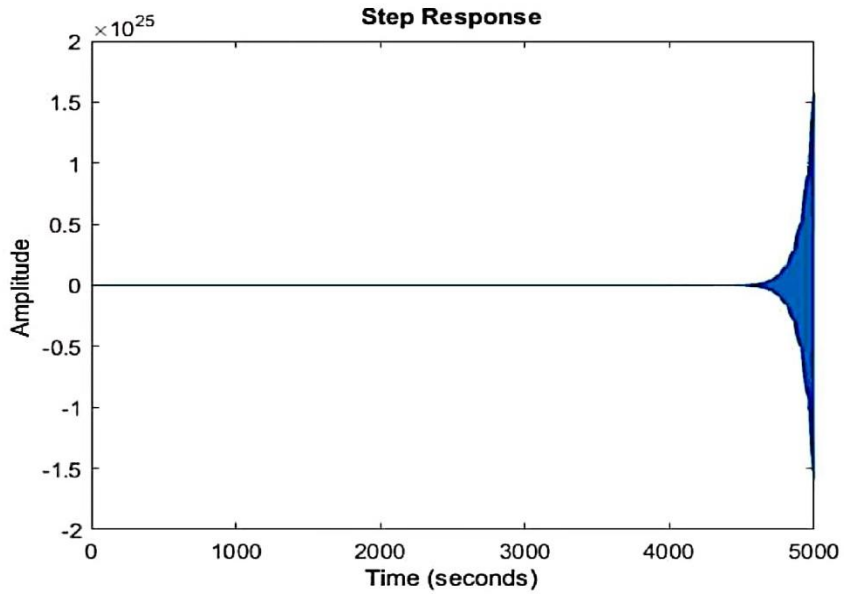
```
G_C=(1+0.746*s)/((1+16*s)*(1+0.093*s))
```

```
G_C =

        0.746 s + 1
  ----------------------
  1.488 s^2 + 16.09 s + 1

Continuous-time transfer function.
```

```
G=G_F*G_C
```

```
G =

                    18.65 s + 25
  ------------------------------------------------
  0.0744 s^5 + 1.697 s^4 + 11.19 s^3 + 16.69 s^2 + s

Continuous-time transfer function.
```

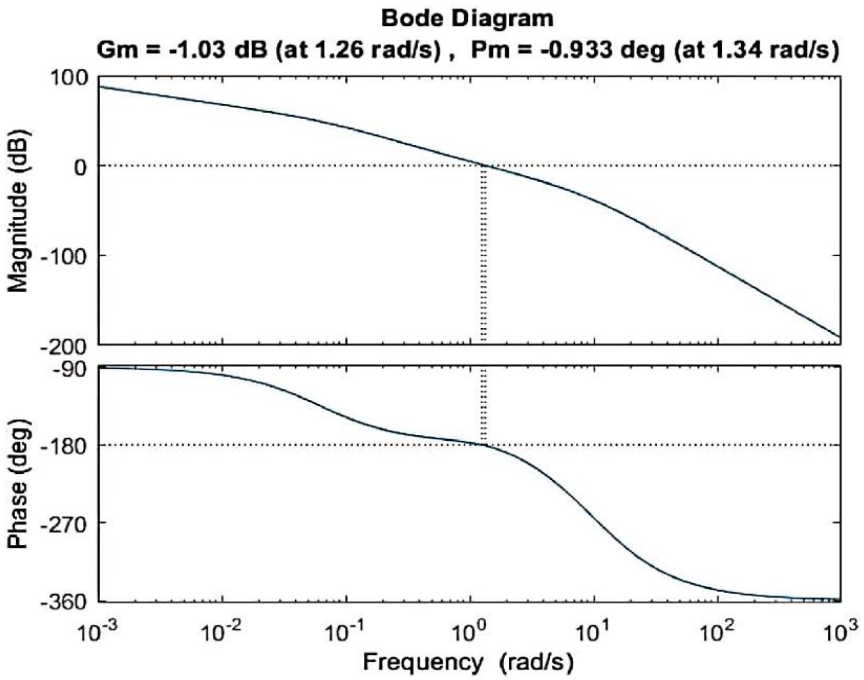
```
figure(3)
step(feedback(G,1))
```




```
stepinfo(feedback(G,1))
```

```
ans = struct with fields:
         RiseTime: NaN
     SettlingTime: NaN
      SettlingMin: NaN
      SettlingMax: NaN
        Overshoot: NaN
       Undershoot: NaN
             Peak: Inf
         PeakTime: Inf
```

```
figure(4)
margin(G)
```




```
[m_r_cont,w_r_cont]=getPeakGain(feedback(G,1))
```

```
m_r_cont = 62.1395
w_r_cont = 1.3413
```

Result:

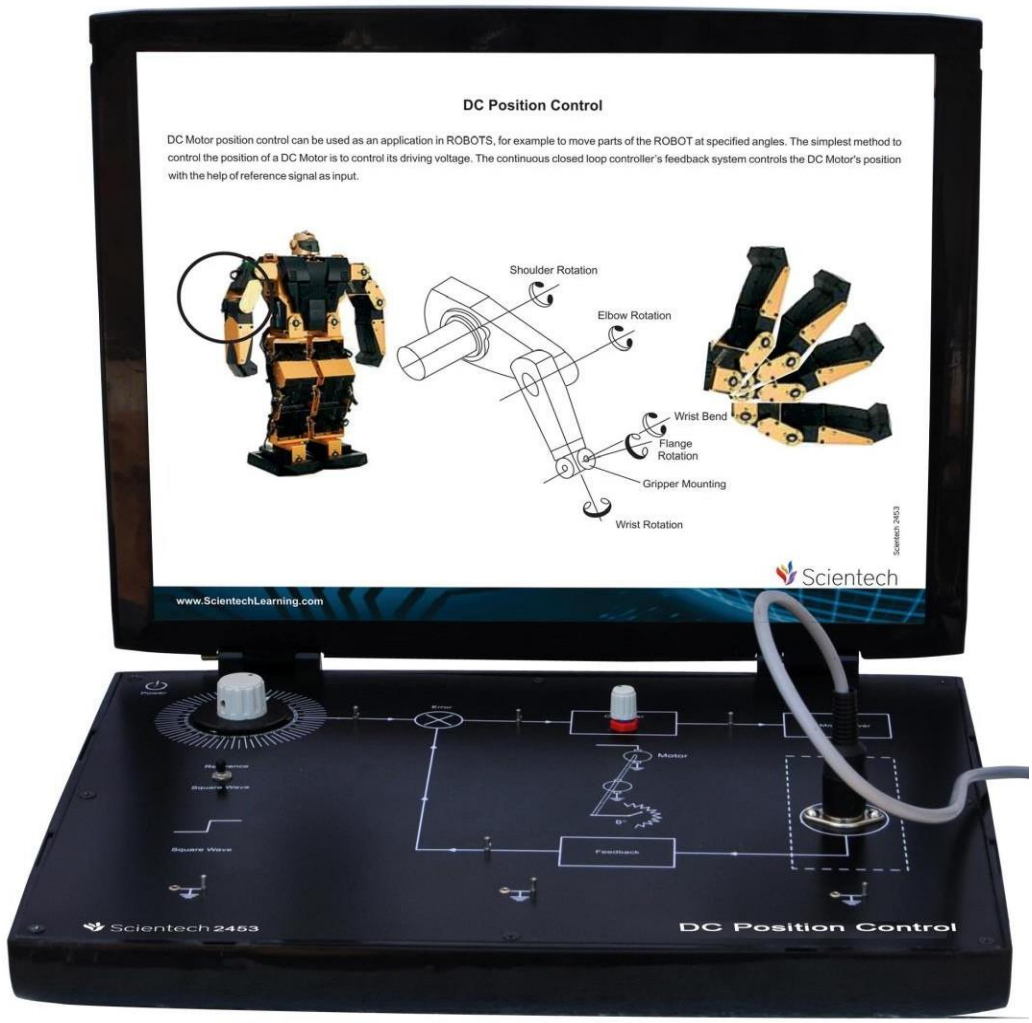
Experiment No - 08

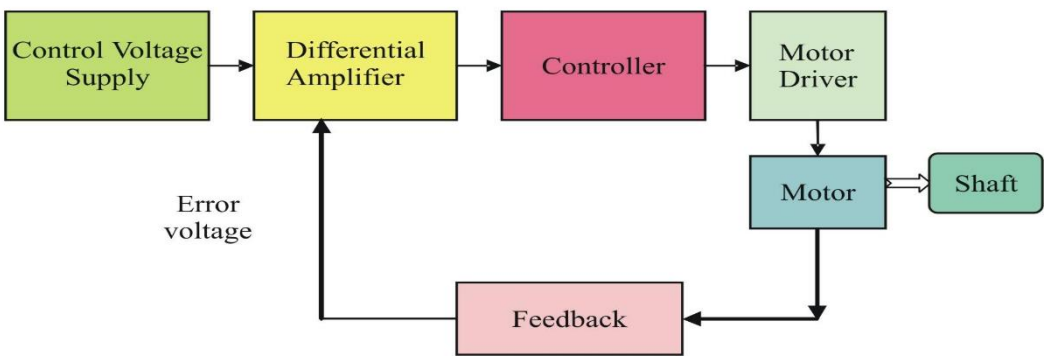
Position Control

Objective: To observe the position control system for different values of the forwardgain at different values of angular position commands.

Equipments needed:

1. Scientech 2453 DC Position Control
2. Additional unit Motor Cabinet for Scientech 2453.
3. DIN Cable.





Procedure:

1. Connect the Motor Cabinet to the Scientech 2453 DC Position Control throughDIN cable at the connector provided in the motor block on the board.
2. Connect the power supply to Scientech 2453 DC Position Control.
3. Switch "On" the supply.
4. Set Toggle switch to upwards i.e. to Reference input.
5. Set the angular positioning command through the pointer as shown on the boardthat will be the reference angle for the DC motor.
6. Observe the voltage value at the reference input before Error block.
7. Now set the forward gain through the knob provided in the controller block on theboard at maximum position.
8. Observe the response at different values of angular command and feedback in thefeedback block on the board.

NOTE: Error in command and output calibrated dial of ± 5 degree is permissible.

Observations:

S.No	Reference Angular position	Actual Angular position + Correction Factor
1.	30	25+5
2.	60	55+5
3.	90	85+5
4.	150	145+5
5.	180	175+5
6.	210	205+5
7.	240	235+5
8.	270	265+5
9.	300	295+5
10.	330	325+5
11.	350	345+5

Result:

Experiment No - 09

DC Motor Speed Control

Experiment No – 09(A)

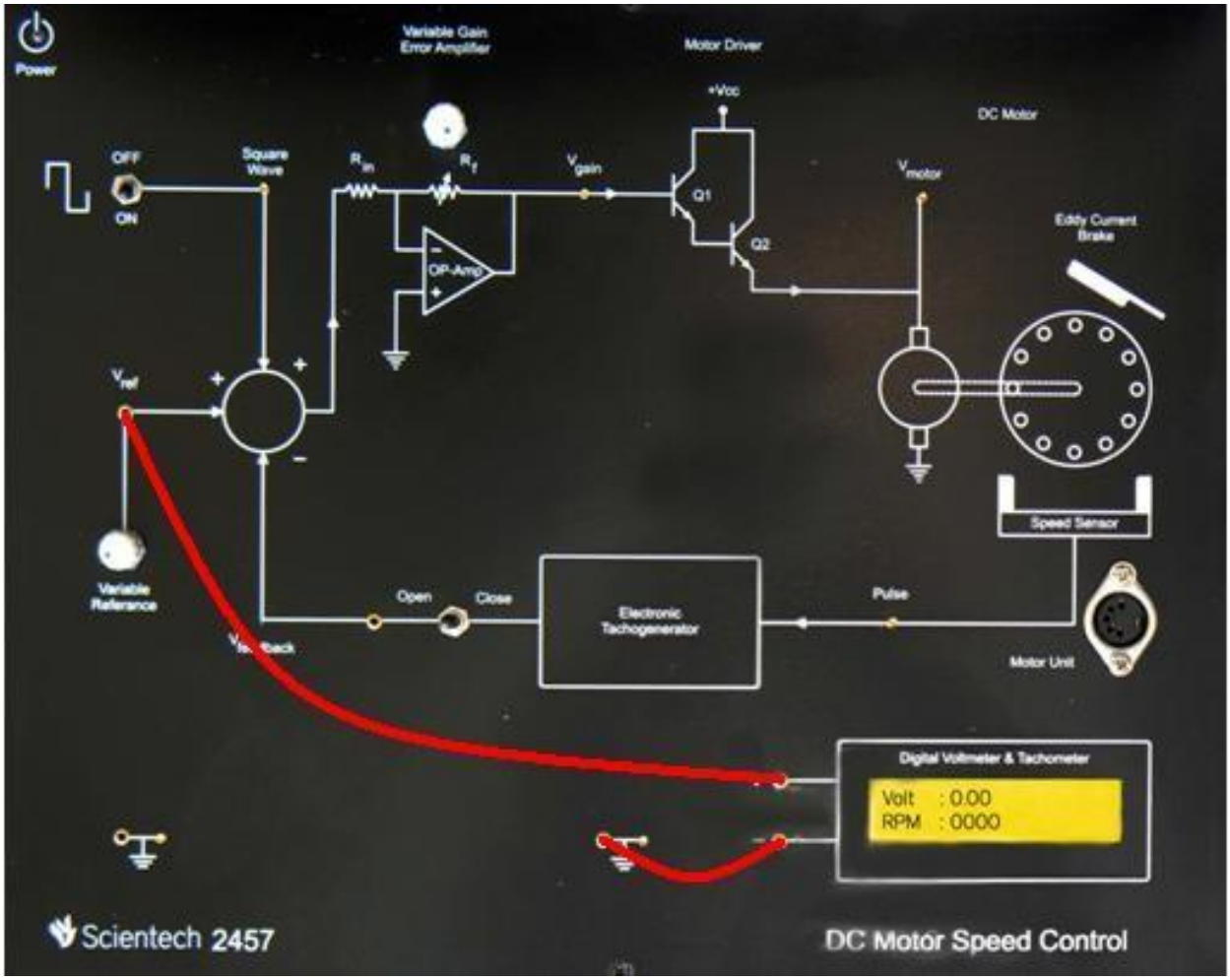
Objective: Study of effect of loading on the speed of the Motor in the open loop(Eddy Current Brake)

Equipments Required:

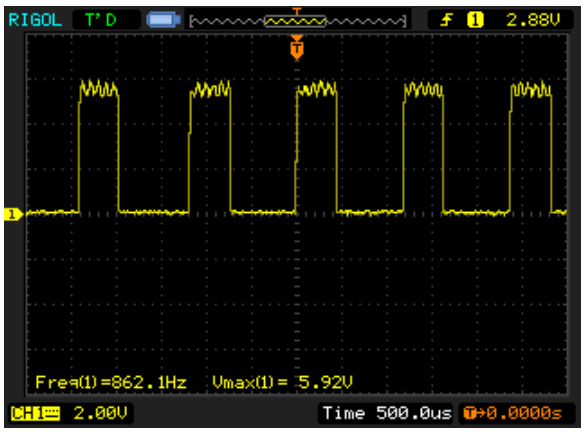
1. Scientech 2457 Techbook
2. Techbook power supply
3. Mains cord
4. Additional unit Motor Cabinet for Scientech 2457.
5. Digital Storage oscilloscope (DSO)
6. DIN Connector Cable.

Procedure:

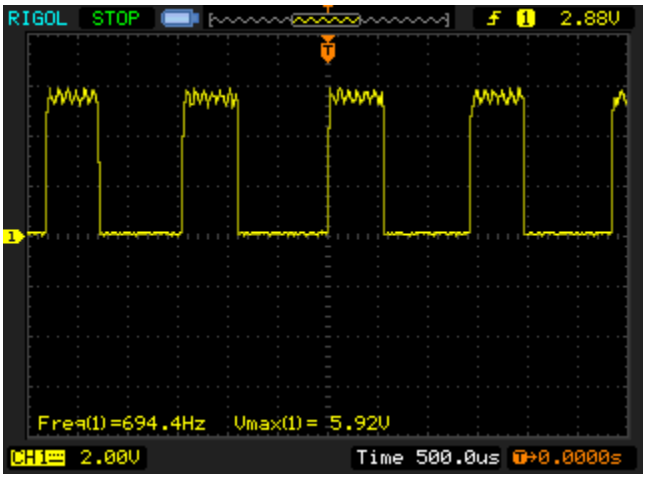
- Connect the Motor Cabinet to the Scientech 2457 DC Motor Speed Controlthrough DIN Connector cable.
- Turn the square wave toggle switch in “OFF” position.
- Turn the Feedback toggle switch in “Open” position.
- Connect the Techbook power supply to the Scientech 2457. Touch over thepower LED to turn ON the power.
- Connect a Digital Storage oscilloscope (DSO) between the test point “Pulse”and GND.



- Rotate the "Variable Reference" pot at the maximum position and observe the waveform at the "Pulse" test point on the Digital Storage oscilloscope (DSO). The waveform is as shown in the figure.



- Rotate the knob on the Motor cabinet and bring the magnet closer to the rotatingwheel of the DC Motor.
- Observe the waveform on the Digital Storage oscilloscope (DSO). The frequency at the "Pulse" test point will start to decrease and the speed of the Motor will also start to reduce.



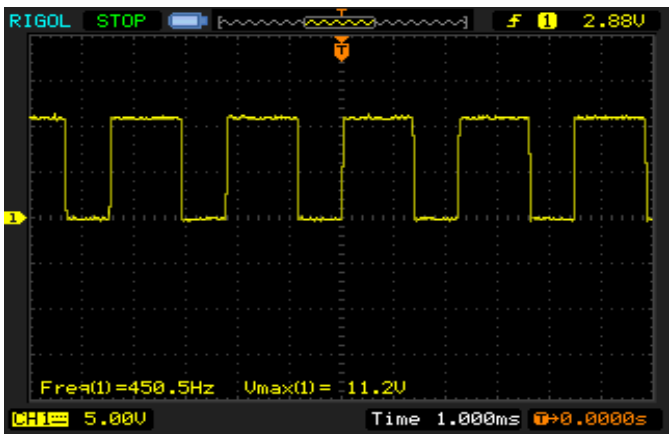
- This Phenomenon is known as Eddy Current Brake.

Experiment 9(B)

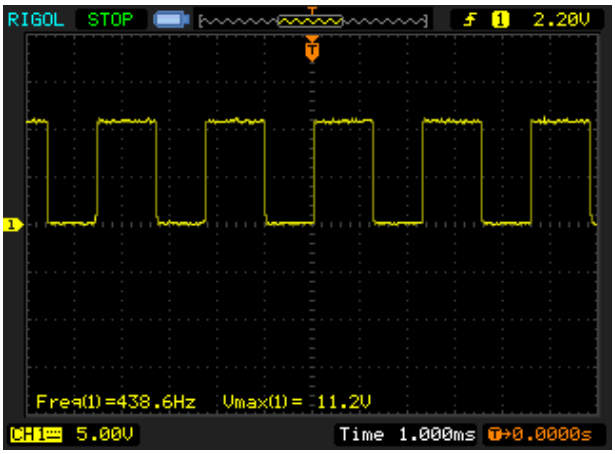
Objective: Effect of loading on the speed of the Motor in the closed loop

Procedure:

- Connect the Motor Cabinet to the Scientech 2457 DC Motor Speed Control through DIN Connector cable.
- Turn the square wave toggle switch in OFF position.
- Turn the Feedback toggle switch in Close position.
- Connect the Techbook power supply to the Scientech 2457. Touch over thepower LED to turn ON the power.
- Connect a Digital Storage oscilloscope (DSO) between the point VMotor andGND.
- Rotate the “Variable Reference” pot at some middle position so that the waveform viewed in the Digital Storage oscilloscope (DSO) is as shown in figure.



- Rotate the knob on the Motor cabinet and bring the magnet closer to the rotatingwheel of the DC Motor.



- The speed of the DC Motor gets reduced when the magnet is brought near the wheel. This is due to the eddy current effect.

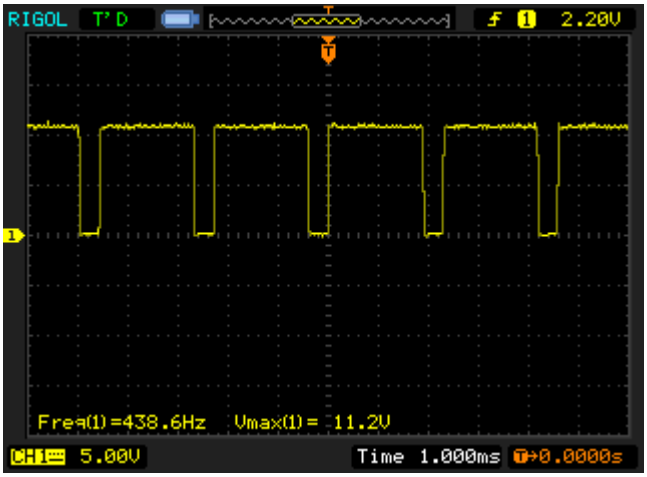
- Observe the waveform on the Digital Storage oscilloscope (DSO). As the magnet is brought near the wheel, the duty cycle of the pulse increases indicating that the controller is trying to maintain the speed of the Motor that is reduced by the eddy current effect. Repeat the above procedure for different “Variable Reference” input.

Experiment 9(C)

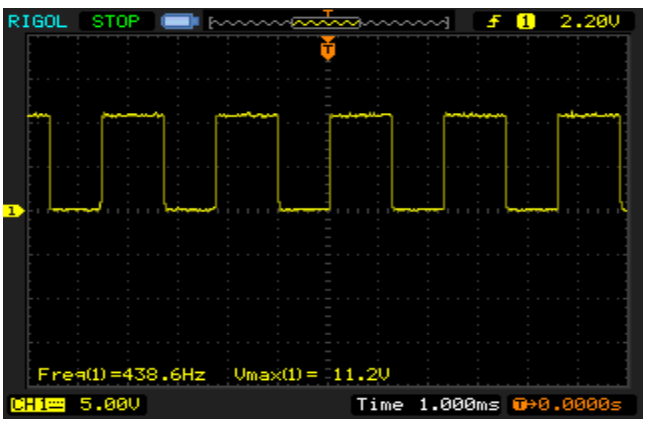
Objective: Study the speed control of a DC Motor.

Procedure:

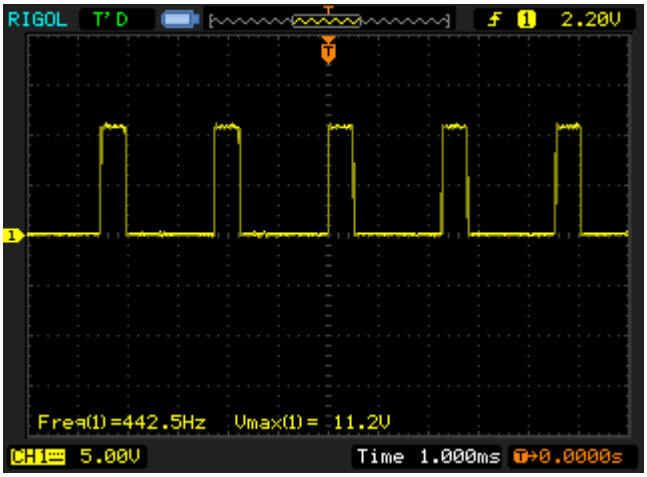
- Connect the Motor Cabinet to the Scientech 2457 DC Motor Speed Controlthrough DIN Connector cable.
- Turn the square wave toggle switch in OFF position.
- Turn the Feedback toggle switch in Close position.
- Connect the Techbook power supply to the Scientech 2457. Touch over thepower LED to turn ON the power.
- Connect a Digital Storage oscilloscope (DSO) between the point VMotor andGND.
- Now vary the "Variable Reference" potentiometer and observe the speedvariation of the Motor.
- A pulse appears on the Digital Storage oscilloscope (DSO) whose duty cyclevaries as the variable reference pot is moved.
- When the pot is at maximum position, the pulse is as shown in figure below. Atthis position the Motor is at its maximum speed.



- Now rotate the pot towards minimum position. The duty cycle of the pulse willalso decrease. This is as shown by the figure.



- When the pot is almost at the minimum, the speed of the Motor is too low andthe signal on Digital Storage oscilloscope (DSO) is as shown in Fig.



- Beyond this point the Motor will stop and there will be no signal to drive the Motor.

S.No	Voltage Reading in Volts	Speed in RPM
1.	0.62	382
2.	0.85	735
3.	1.25	1290
4.	1.56	1690
5.	1.88	2012
6.	2.43	2430

Result:

Experiment No – 10(A)

Block diagram reduction technique using MATLAB

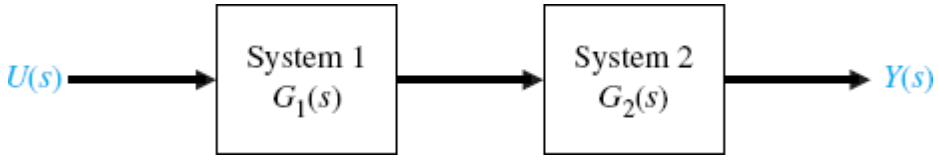
Aim: simulation of multi feedback system

SOFTWARE REQUIRED: MATLAB – Personal Computer with MATLAB

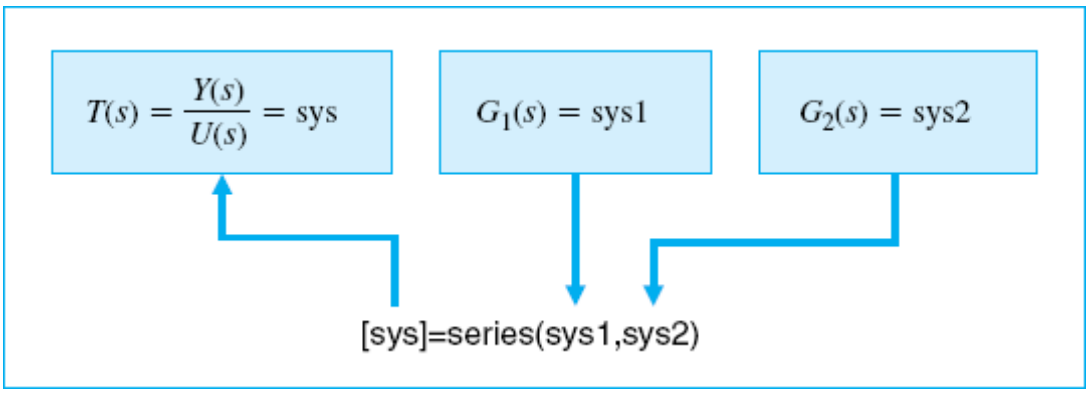
Theory:

Series configuration: If the two blocks are connected as shown below then the blocks are said

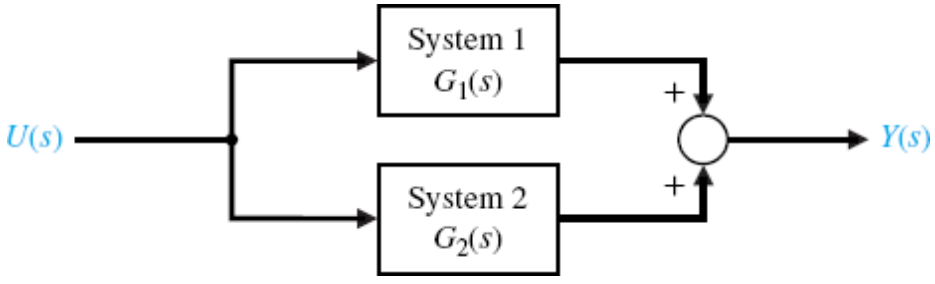
to be in series. It would like multiplying two transfer functions. The MATLAB command for the suchconfiguration is "series".



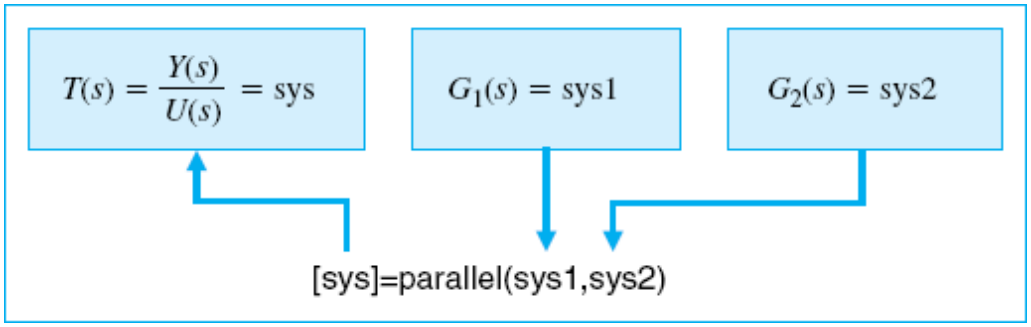
The series command is implemented as shown below



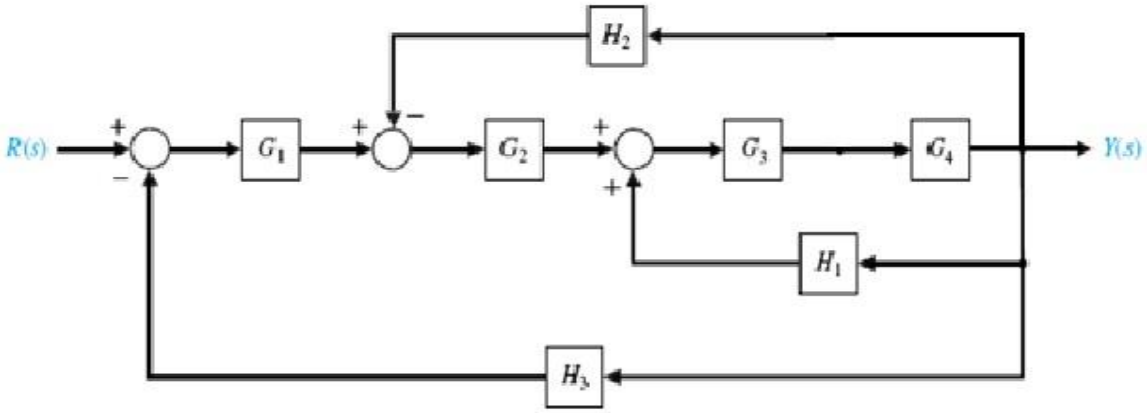
Parallel configuration: If the two blocks are connected as shown below then the blocks are said tobe in parallel. It would like adding two transfer functions



The MATLAB command for implementing a parallel configuration is "parallel" as shown below:



Block diagram:



Program:

```
ng1=[1]; dg1=[1 0]; sysg1=tf(ng1, dg1)
ng2=[1]; dg2=[1 0]; sysg2=tf(ng2, dg2)
ng3=[1]; dg3=[1 0]; sysg3=tf(ng3, dg3)
ng4=[1]; dg4=[1 0]; sysg4=tf(ng4, dg4)
nh1=[1]; dh1=[1]; syshl=tf(nh1,dh1)
nh2=[1]; dh2=[1]; sysh2=tf(nh2,dh2)
nh3=[1]; dh3=[1]; sysh3=tf(nh3,dh3)
sys1=series(sysg3,sysg4)
sys2=feedback(sys1,syshl,+1)
sys3=series(sysg2,sys2)
sys4=feedback(sys3,sysh2,-1)
sys5=series(sysg1,sys4);
sys=feedback(sys5,[1])
```

Theoretical calculations:

Compare the theoretical calculations for the above block diagram with MATLAB

All the transfer functions are given in the program.

	Theoretical Calculations	MATLAB
Block Diagram	T.F=	T.F=

Result:

Experiment No – 10(B)

STATE MODEL FOR CLASSICAL TRANSFER FUNCTION & VICE VERSA USING MATLAB

AIM: To find state model for classical transfer function and transfer from state model using MATLAB

APPARATUS: Computer with MATLAB software.

PROCEDURE:

1. Write the programme in MATLAB text editor using mat lab instructions for state model of classical transfer function and for tranfer function from state model.
2. Run the programs.
3. Note down the outputs.

PROGRAM 1:

$\boldsymbol{A} = \begin{bmatrix} \mathbf{0} & \mathbf{2} \\ \mathbf{1} & \mathbf{-1} \end{bmatrix}$ B=$\begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}$ C= $[\mathbf{1} \quad \mathbf{0}]$

```
a=input ('Enter the values of a matrix');
b=input('Enter the values of b matrix');
c=input('Enter the values of c matrix');
d=input('Enter the values of d matrix');
[num,den]=SS2 tf (a,b,c,d,1)
Transfer function=tf (num,den);
```

PROGRAM 2:

$$\boldsymbol{T.F} = \frac{\mathbf{2}}{(\mathbf{s}-\mathbf{1})(\mathbf{s}+\mathbf{2})}$$

```
num = input('Entre numerator polynomial values in the form of matrix array');
den1=input('Enter denominator 1 values');
den2=input('enter denominator 2 values');
den=conv(den1,den2);
H=tf(num,den);
P=SS(H);
[a,b,c,d]=SS data(P);
```

PROGRAM3: Controllability and Observability

```
% State Space Representation
% x' = Ax + Bu
% y = Cx + Du
% Check Controllability and Observability of a 3rd order System
% Given ----------------------------------------------------------------
MatrixA = [-3 -1 0;2 0 0;0 -1 -1];
MatrixB = [1;0;1];
MatrixC = [1 0 1];
MatrixD = 0;
% Objective ------------------------------------------------------------
% 1) To Find Controllable Matrix Qc, its rank and check controllability
% 2) To Find Observable Matrix Qb, its rank and check observability
% Controllable Matrix --------------------------------------------------
Qc = ctrb(MatrixA,MatrixB);
rankQc = rank(Qc);
disp('Controllable Matrix is Qc = ');
disp(Qc);
if(rankQc == rank(MatrixA))
disp('Given System is Controllable.');
else
disp('Given System is Uncontrollable');
end
% Observable Matrix ----------------------------------------------------
Qb = obsv(MatrixA, MatrixC);
rankQb = rank(Qb);
disp('Observable Matrix is Qb = ');
disp(Qb);
if(rankQb == rank(MatrixA))
disp('Given System is Observable.');
```

```
else
disp('Given System is Unobservable');
end
% End of Program ----------------------------------------------------
```

Observations:

Compare the Theoretical calculations with MATLAB.

	Theoretical Calculations	MATLAB
Program 1	T.F=	T.F=
Program 2	A= B= C= D=	A= B= C= D=
Program 3	Controllability matrix = Observable Matrix=	Controllability matrix = Observable Matrix=

RESULT: