1. AIM – PROGRAM TO FIND BODE PLOT

CODE –

```
clc
num=input('Enter the value numerator:')
den=input('Enter the value denominator:')
fun=tf(num,den)
bode(fun)
gridon
```

INPUT –

Enter the value numerator: [1 4]

num =

1 4

Enter the value denominator: [4 6 8]

den =

4 6 8

OUTPUT –

fun =

$$\frac{s + 4}{4\,s^2 + 6\,s + 8}$$

Continuous-time transfer function.

GRAPH –

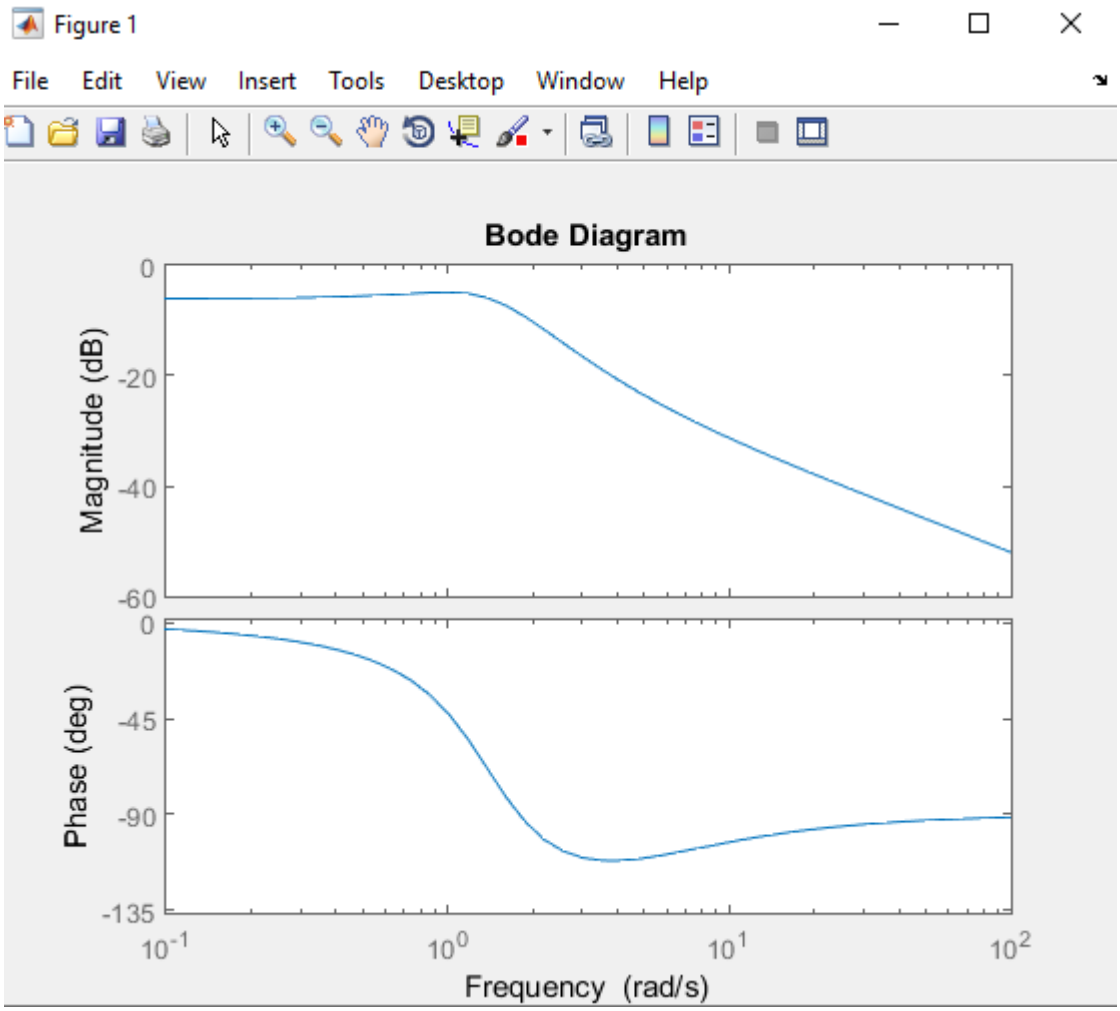


Figure 1
File Edit View Insert Tools Desktop Window Help
Bode Diagram
Magnitude (dB)
0
-20
-40
-60
Phase (deg)
0
-45
-90
-135
10^-1
10^0
10^1
10^2
Frequency (rad/s)

2. AIM – PROGRAM FOR UNIT STEP RESPONSE TO FIND RISE TIME AND DELAY TIME

CODE –

```
clc
sys=tf([8 18 32],[1 6 14 24])
S=stepinfo(sys,'RiseTimeLimits',[0.05,0.95])
step(sys)
gridon
```

OUTPUT –

sys =

$$\frac{8\,s^2 + 18\,s + 32}{s^3 + 6\,s^2 + 14\,s + 24}$$

Continuous-time transfer function.

S =

RiseTime: 0.2393

SettlingTime: 3.4972

SettlingMin: 1.1956

SettlingMax: 1.6871

Overshoot: 26.5302

Undershoot: 0

Peak: 1.6871

PeakTime: 0.5987

GRAPH –

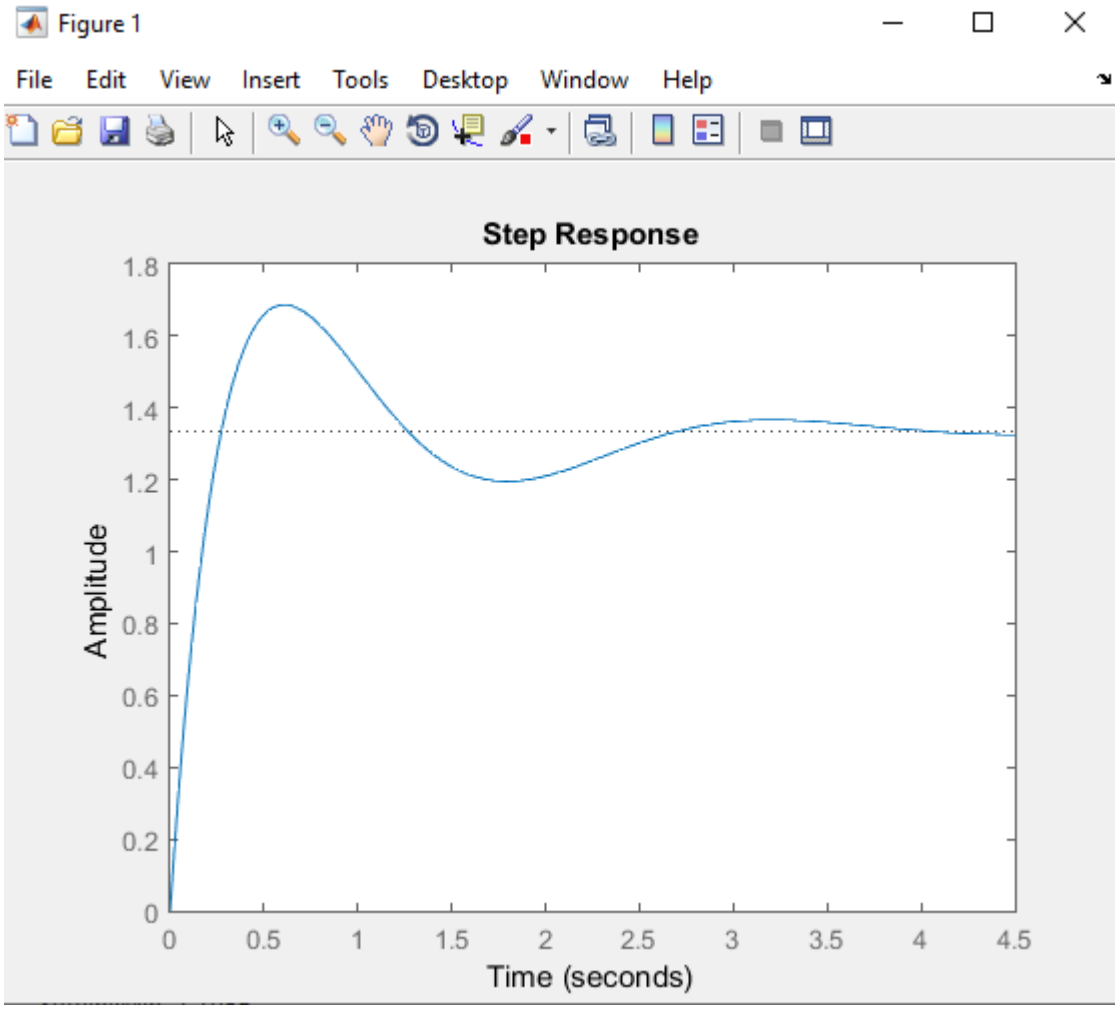


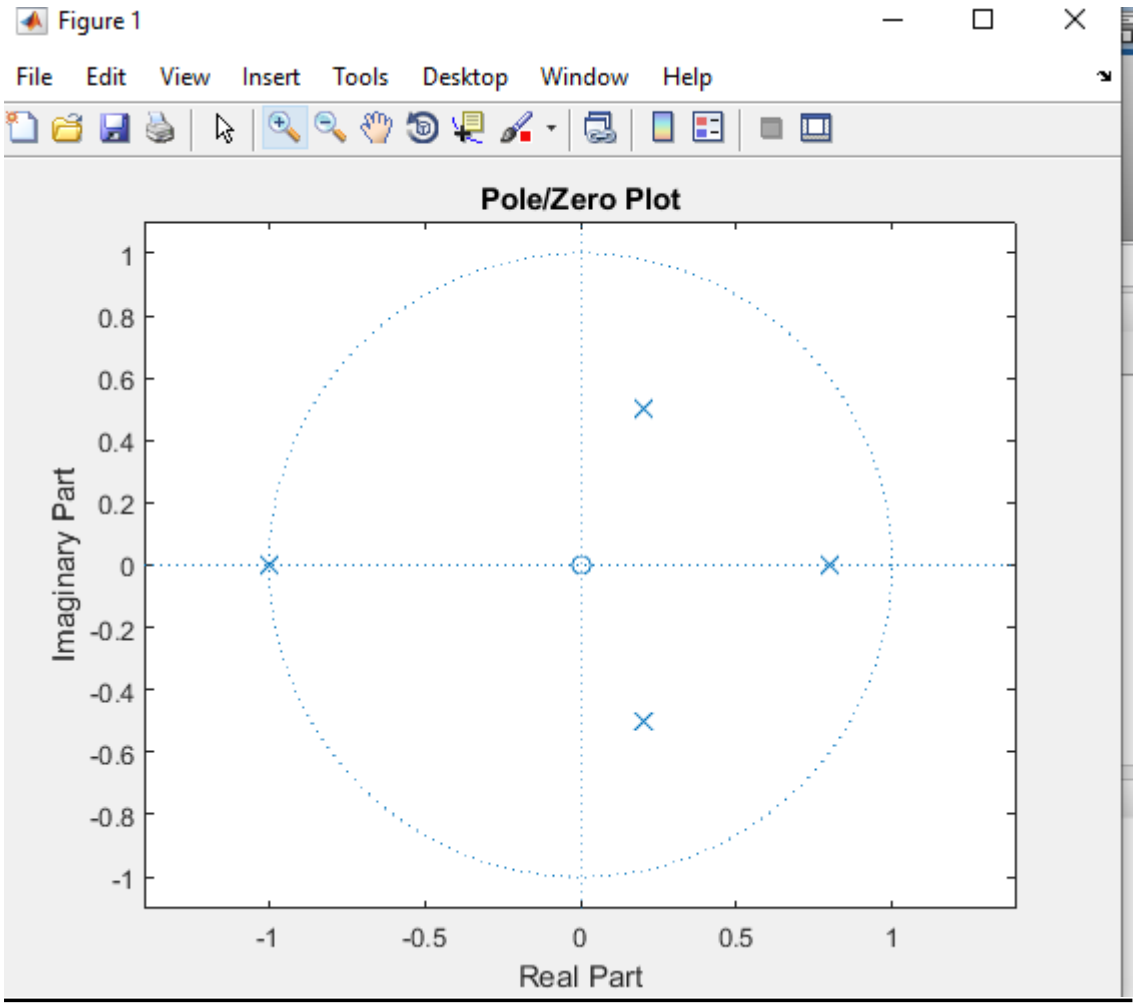
Figure 1
File Edit View Insert Tools Desktop Window Help
Step Response
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5

3. AIM – PROGRAM FOR POLE ZERO CONFIGURATION IN S-PLANE

CODE –

```
clc
z=[j -j]
p=[-1;0.8;0.2+0.5j;0.2-0.5j];
figure(1);
zplane(z,p);
title('Pole/Zero Plot')
```

OUTPUT –

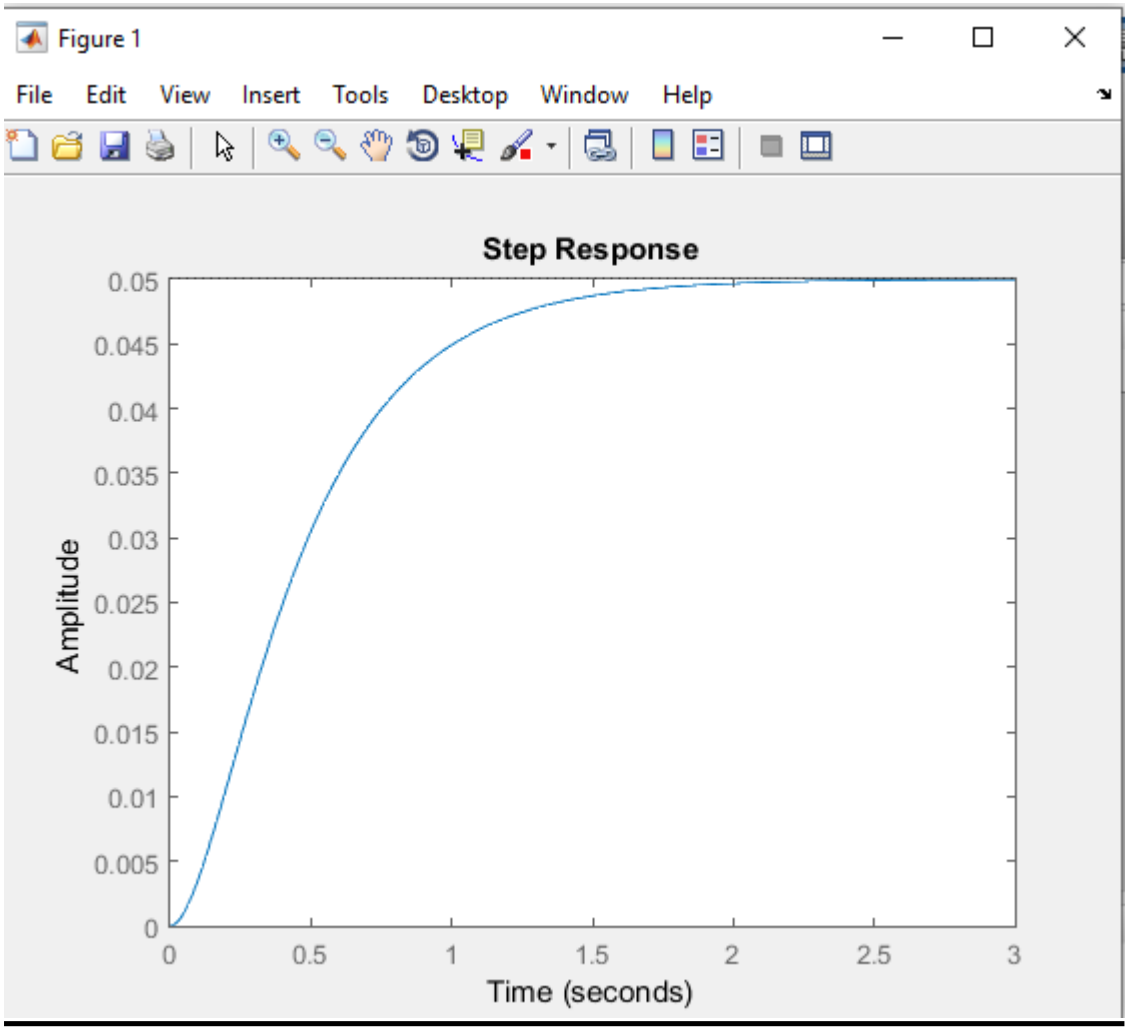


4. AIM – PROPORTIONAL CONTROL

CODE –

```
num=1;
den=[1 10 20];
plant=tf(num,den);
step(plant)
```

OUTPUT –

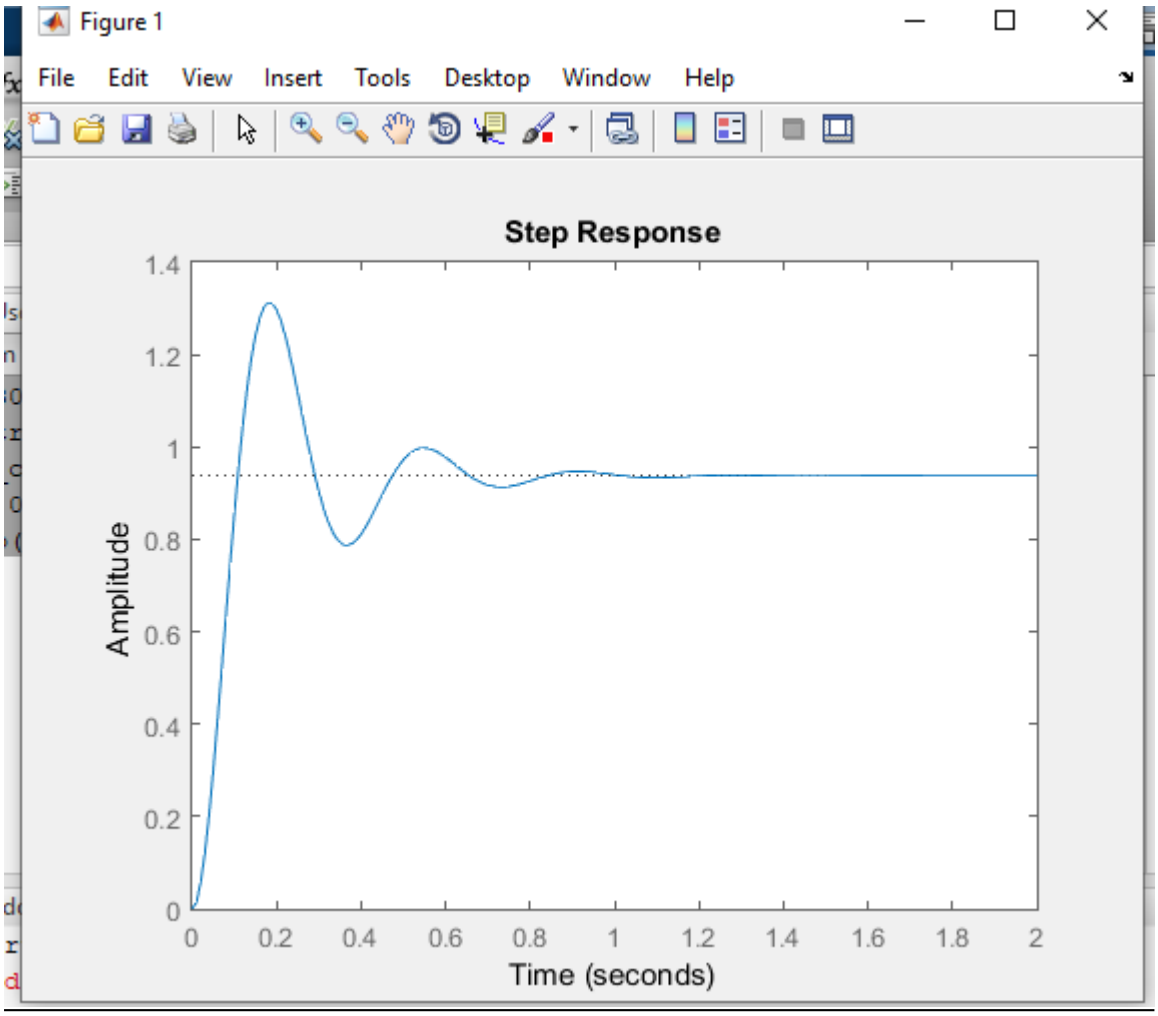


PROPORTIONAL CONTROL –

CODE –

```
kp=300;
contr=kp;
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

OUTPUT –

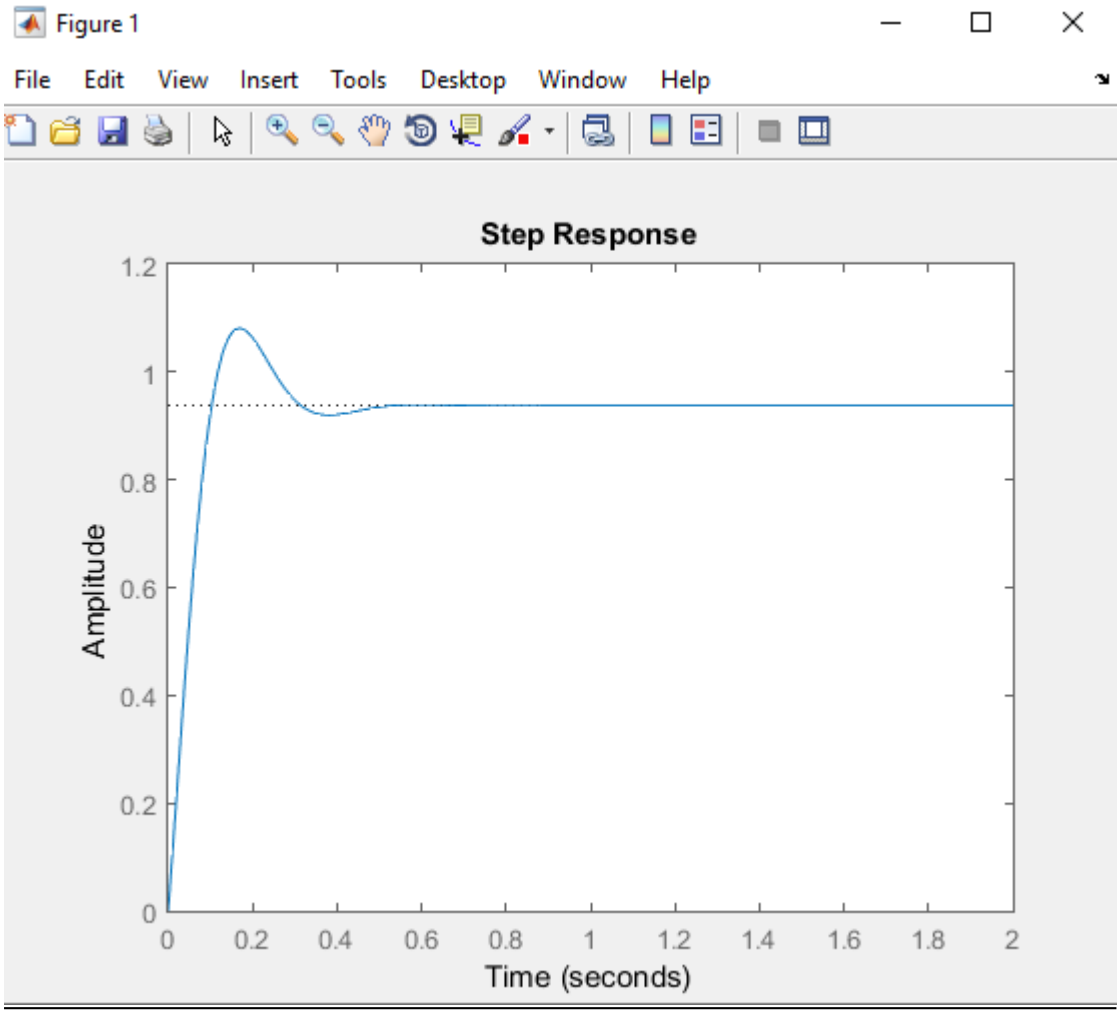


5. AIM – PROPORTIONAL-DERIVATIVE CONTROL

Code –

```
Kp=300;
Kd=10;
contr=tf([Kd Kp],1);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

output –

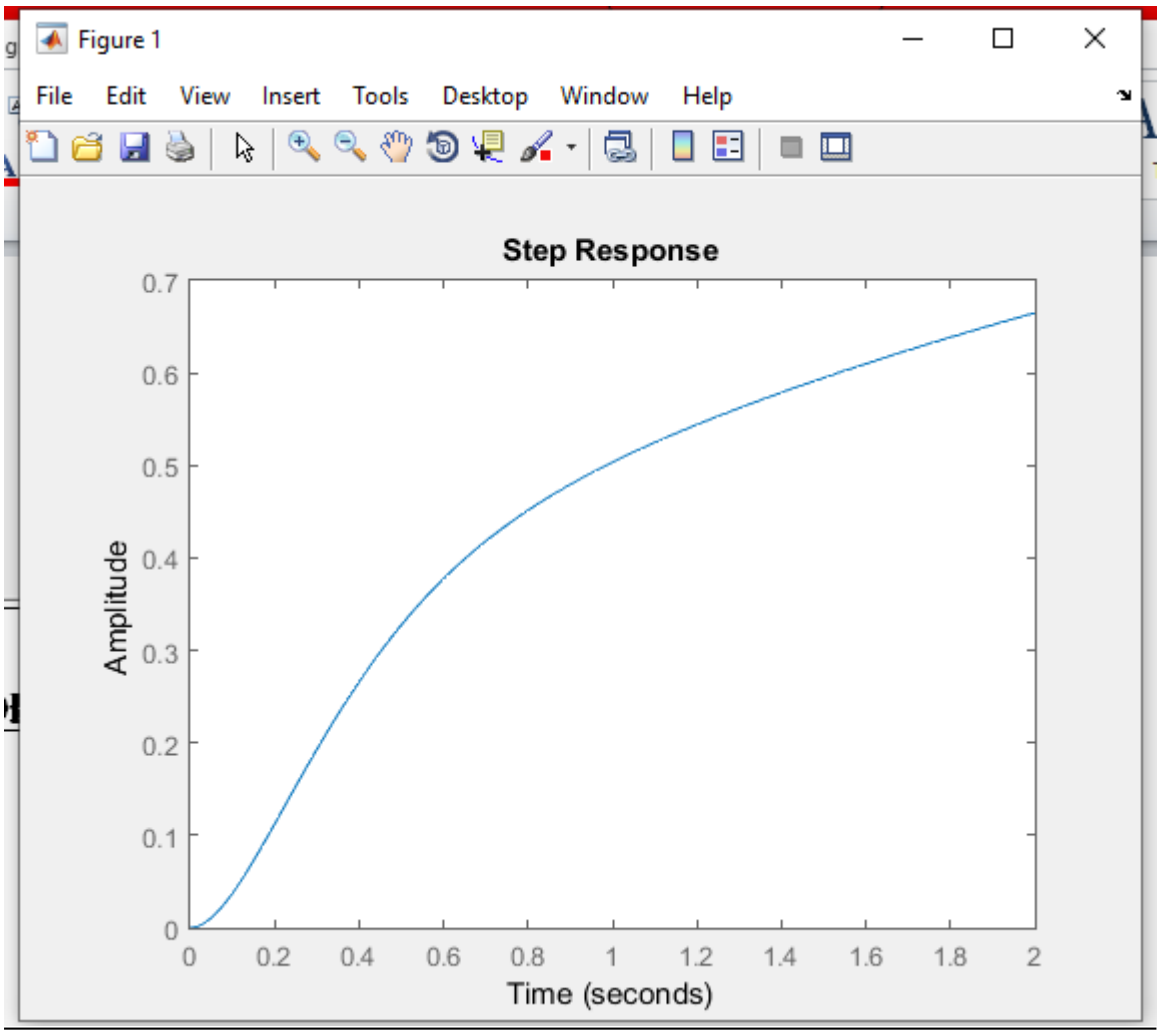


6. AIM – PROPORTIONAL - INTEGRAL CONTROL

Code –

```
Kp=300;
Ki=10;
contr=tf([Kd Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

output –



7. AIM – STUDY OF P, PI, PD AND PID CONTROLLER.

Apparatus Required – MATLAB Software

Theory –

When the plant's performance isn't as per the system's requirements, a closed loop control system can be handy, instead of a new plant. The error between the actual and the required outputs derived the controller. This controller in turn produces the necessary correcting actions upon the plant pulling the actual output towards the required one. The easiest approach is to produce the actuating signal u proportional to the error, the more is the error, the higher will be the actuating signal magnitude. This allows the actual output to follow the required output. This is the Proportional controller. This controller is useful when the error is under some limit.

PD Controller

Practically, no single controller can serve the purpose. It needs multiple controllers work together to meet the requirements. Whenever, it's needed to speed up the transient period, we add proportional control and the overshoot and the oscillatory problems need the derivative control. This calls for Proportional- Derivative controller (PD Controller). Special care must be taken to ensure that the signals are noiseless, else the noise can get amplified by the derivative actions and creates stability problems to the plant.

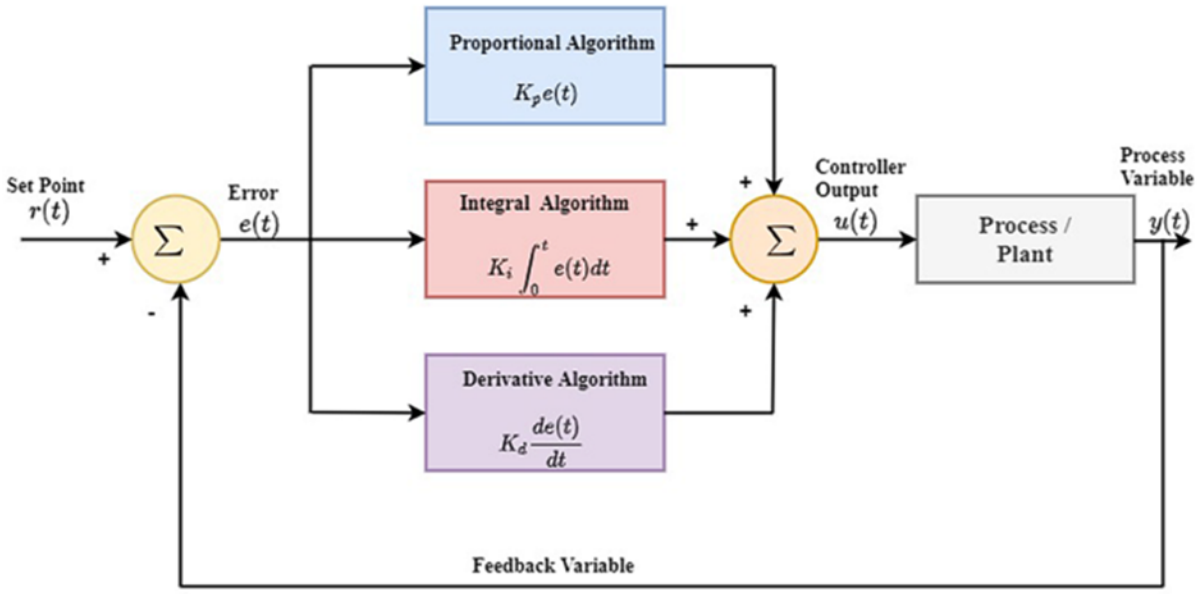
PI Controller

This helps in the situation where proportional control is necessary to speed up the settling and integral control to reduce the error that is constant over time. This is Proportional Integral controller (PI controller).

PID Controller

More general case would be to address all the above discussed problems with a single solution. In this case all the three controllers are used in parallel will appropriately chosen gains. Adjusting these gains, any combination of P,I,D controllers can be obtained and hence a more robust one.

The three gains — Kp, Ki and Kd for the Proportional, Integral and Derivative controller actions can decide the contribution of each one in the final controller action.



Code –

```
% Open loop step response
num = 1;
den = [1 10 20];
plant = tf(num,den);
step(plant)
title('open loop step response')
```

% Proportional control

```
Kp= 300;
contr = Kp;
sys_cl = feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
title('Poportional Control')
```

% Proportional-Derivative control

```
Kp=300;
Kd=10;
```

```
contr=tf([Kd Kp],1);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
title('Poportional-Derivative Control')
```

% Proportional-Integral control

```
Kp=30;
Ki=70;
contr=tf([Ki Kp],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
title('Proportional-Integral control')
```

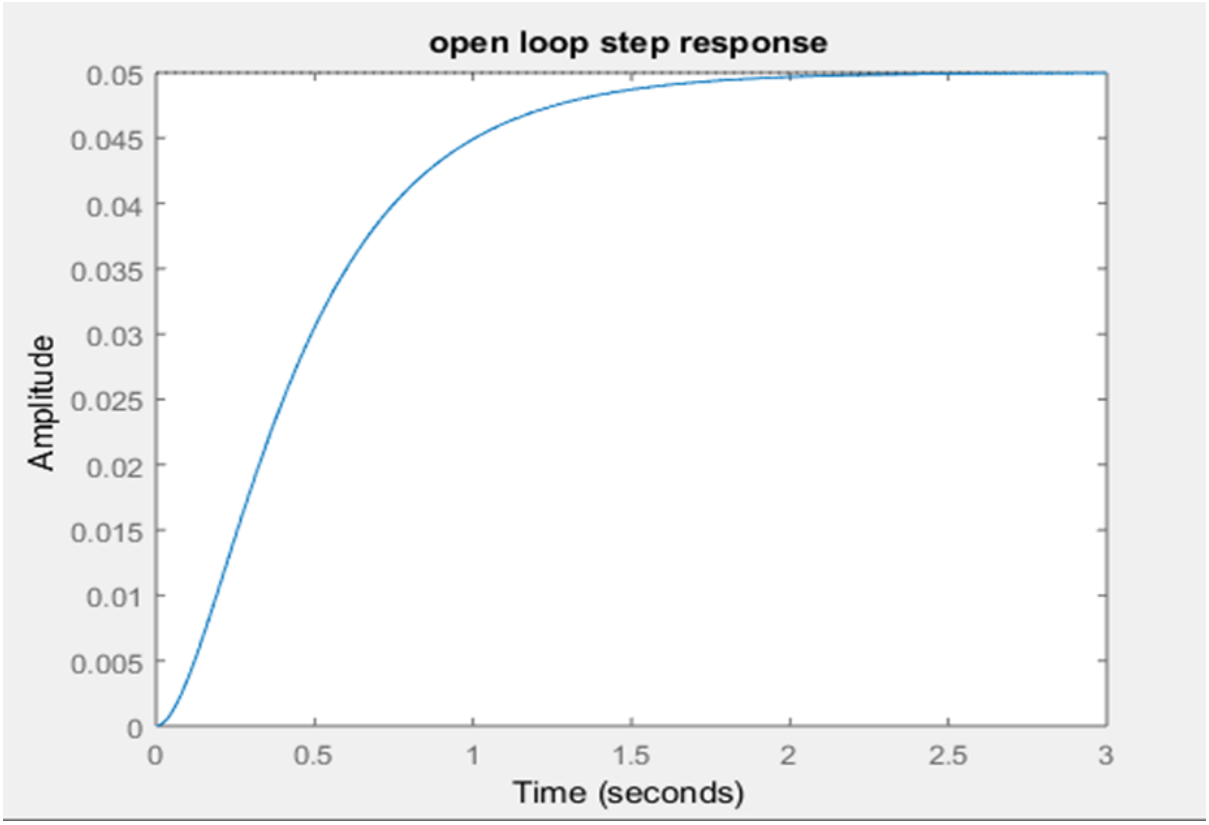
% Proportional-Integral-Derivative control

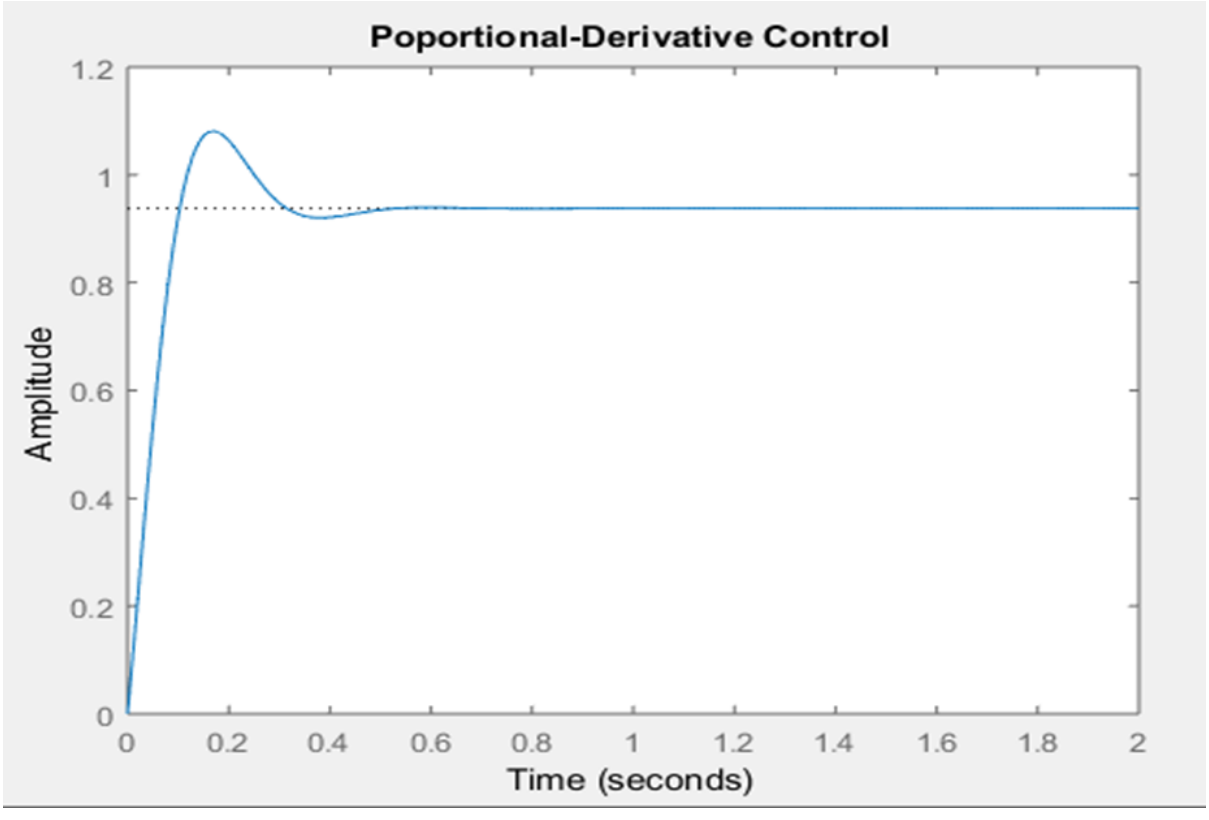
```
Kp=350;
Ki=300;
Kd=50;
contr=tf([Kd Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
title('Proportional-Integral-Derivative
control')
```

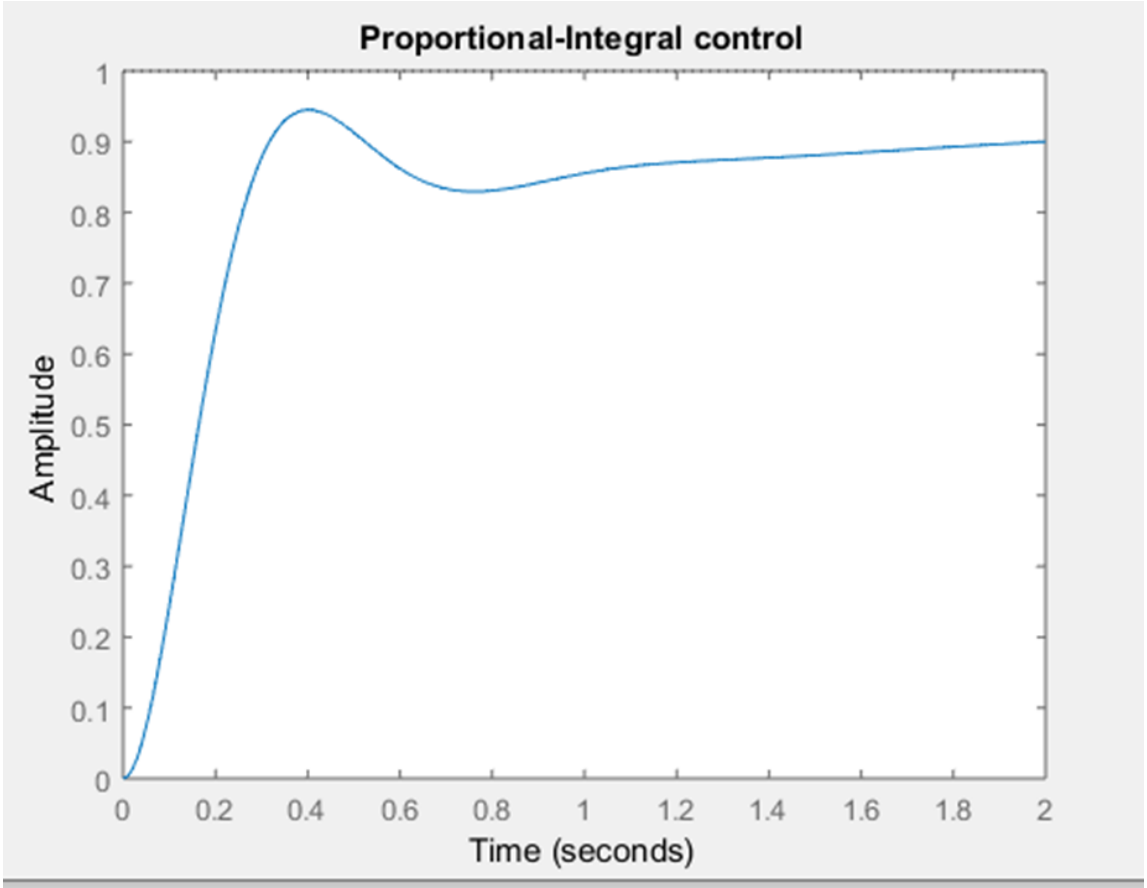
Output –

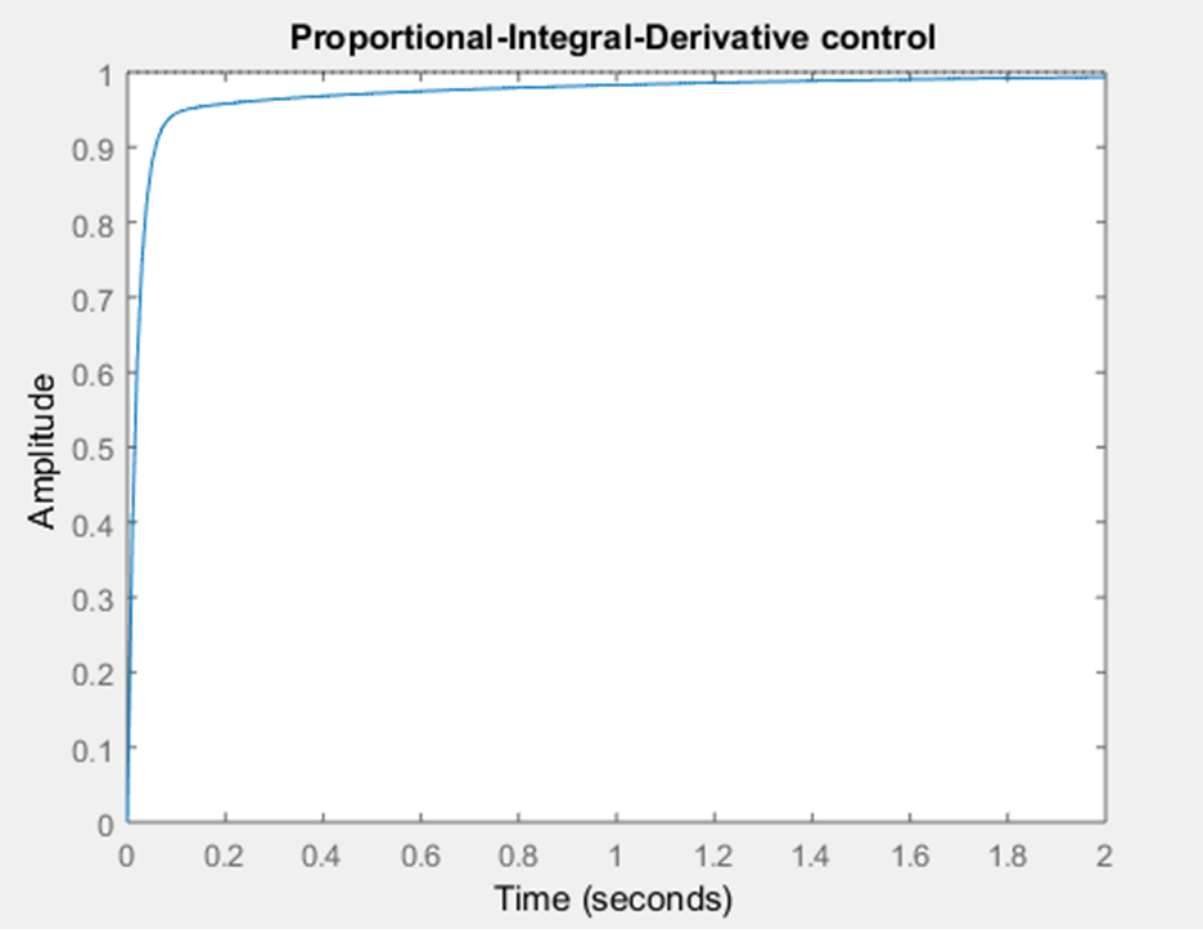
```
Kp=350;
Ki=300;
kd=50;
contr=tf([Kd Kp Ki],[1 0]);
sys_cl=feedback(contr*plant,1);
t=0:0.01:2;
step(sys_cl,t)
```

output –


open loop step response
Amplitude
Time (seconds)
0.05
0.045
0.04
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
0
0.5
1
1.5
2
2.5
3


Poportional-Derivative Control
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Proportional-Integral control
Amplitude
Time (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


Proportional-Integral-Derivative control
Amplitude
Time (seconds)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2