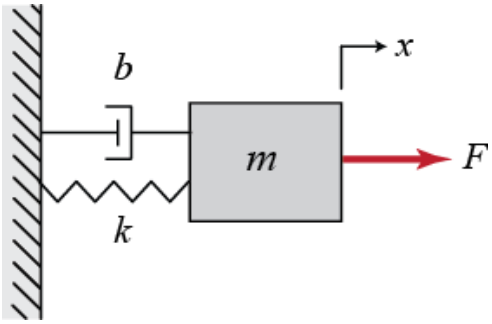
# Lab # 1

## PID Controller

### Objectives:

- Study the effect of $K_p$, $K_i$ & $K_D$ on feedback loop response.
- Investigate the characteristic of proportional, integral & derivative.
- How to use to obtain desire response?




### Theory:

**Modeling:**

$$f(t) = m\frac{d^2x}{dt^2} + b\frac{dx}{dt} + kx(t)$$

$$F(s) = Ms^2X(s) + bs\,X(s) + K\,X(s)$$

$$\frac{X(s)}{F(s)} = \frac{1}{Ms^2+bs+K}$$

Let

M=1kg
b=10Ns/m
K=20N/m

$$\frac{X(s)}{F(s)} = \frac{1}{s^2+10s+20}$$

To find DC gain put s=0 in equation

$$\text{DC gain} = \frac{1}{20} = 0.05$$

### Goal:

Show how can $K_p$, $K_i$ & $K_D$ contributes to obtain

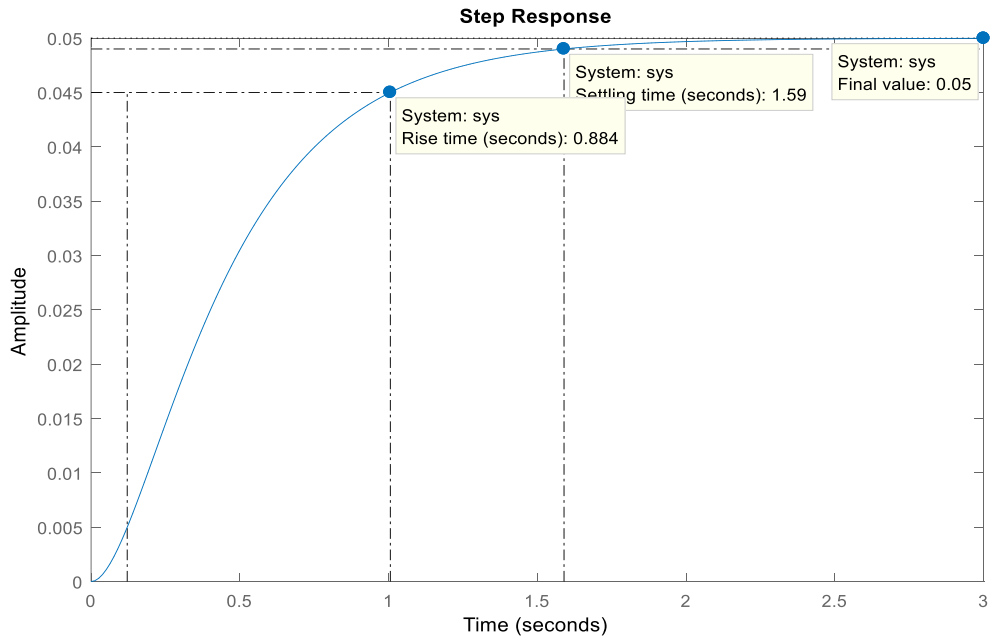
1. Fast Rise Time
2. Min Overshoot
3. No Steady State Error

## Model representation and step response in M-file (without $K_p$, $K_i$ & $K_D$):

```
clear all;
close all;
num=[1];
den=[1 10 20];
sys=tf(num,den);
step(sys)
stepinfo(sys)
```

**Results:**

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



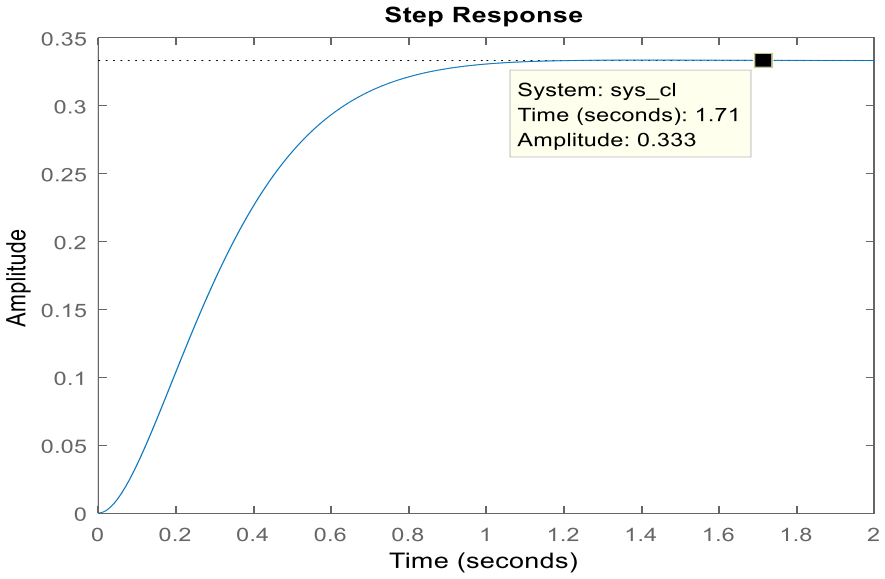
## Model representation and step response in M-file (with $K_p$):

### $K_p = 10$:

```
clear all;
close all;
num=[1];
den=[1 10 20];
sys=tf(num,den);
Kp=10;
contr=Kp;
G=contr*sys;
sys_cl=feedback(G,1);
t=0:0.01:2;
step(sys_cl,t)
stepinfo(sys_cl)
```

**Results:**

```
     RiseTime: 0.5371
 SettlingTime: 0.8838
  SettlingMin: 0.3005
  SettlingMax: 0.3336
    Overshoot: 0.0889
   Undershoot: 0
         Peak: 0.3336
     PeakTime: 1.4092
```



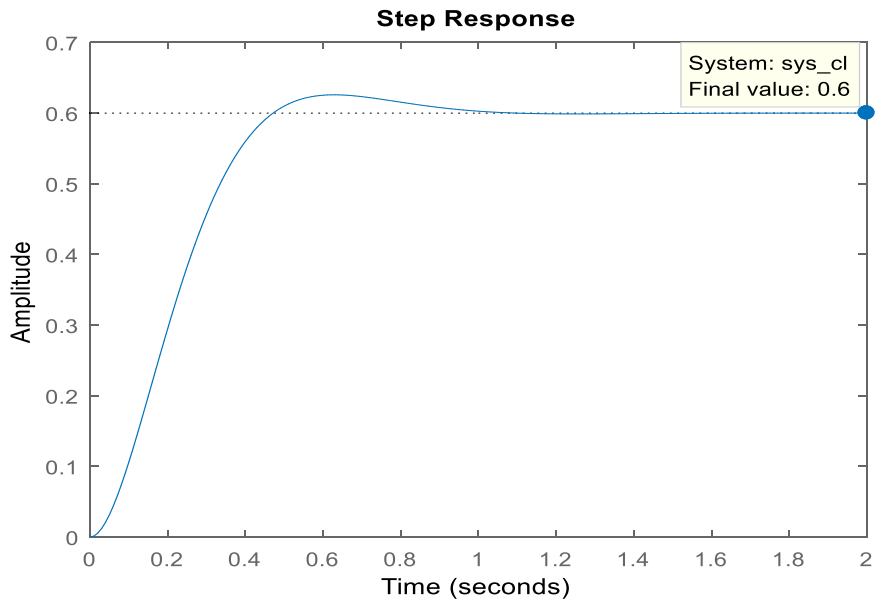
Steady state error = (1-0.333) / 0.333 = 200%

### $K_p = 30$:

```
clear all;
close all;
num=[1];
den=[1 10 20];
sys=tf(num,den);
Kp=30;
contr=Kp;
G=contr*sys;
sys_cl=feedback(G,1);
t=0:0.01:2;
step(sys_cl,t)
stepinfo(sys_cl)
```

**Results:**

```
     RiseTime: 0.3039
 SettlingTime: 0.8433
  SettlingMin: 0.5421
  SettlingMax: 0.6259
    Overshoot: 4.3210
   Undershoot: 0
         Peak: 0.6259
     PeakTime: 0.6263
```

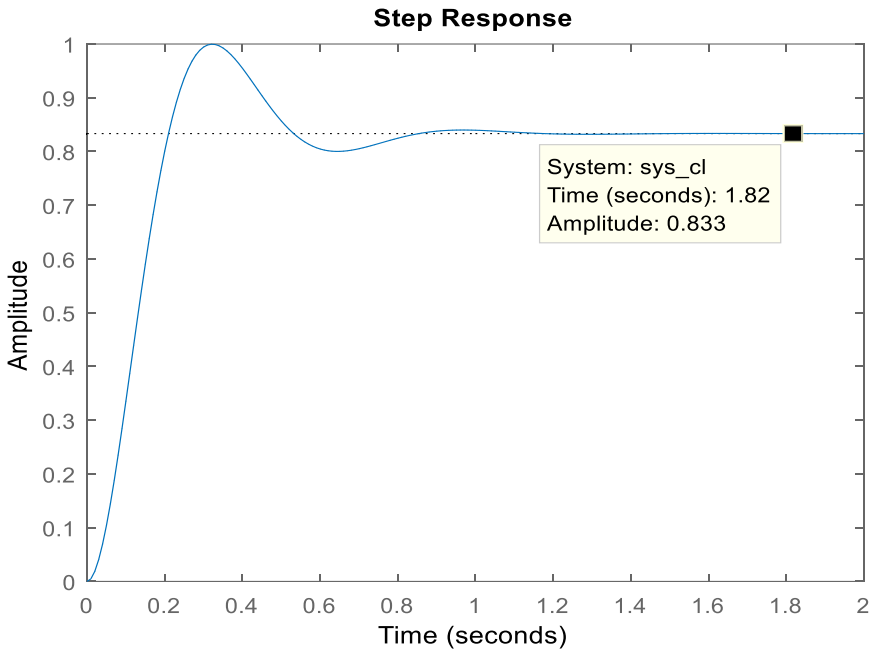


Steady state error = (1-0.6) / 0.6 = 66.67%

### $K_p = 100$:

**Results:**

```
     RiseTime: 0.1423
 SettlingTime: 0.7600
  SettlingMin: 0.7767
  SettlingMax: 0.9996
    Overshoot: 19.9567
   Undershoot: 0
         Peak: 0.9996
     PeakTime: 0.3224
```




Steady state error = (1-0.833) / 0.833 = 20%

### Characteristics for increases $K_p$:

- Settling time increasing with small change.
- Rise time decreased.
- Steady state error decreased.
- Overshoot increased.

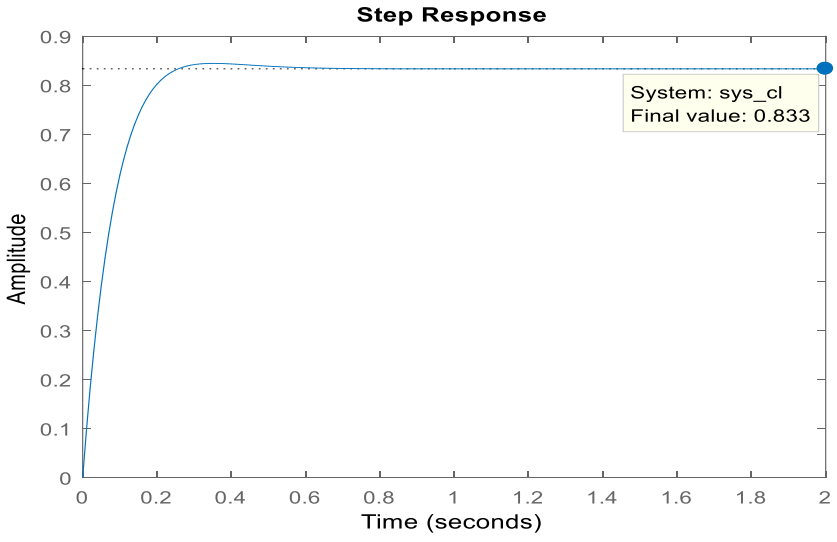
## Model representation and step response in M-file (with $K_P$=100 & $K_D$):

### $K_D$ = 10:

```
clear all;
close all;
num=[1];
den=[1 10 20];
sys=tf(num,den);
Kp=100;
Kd=10
num1=[Kd Kp];
den1=[1];
contr=tf(num1,den1);
G=contr*sys;
sys_cl=feedback(G,1);
t=0:0.01:2;
step(sys_cl,t)
stepinfo(sys_cl)
```

**Results:**

```
     RiseTime: 0.1476
 SettlingTime: 0.2202
  SettlingMin: 0.7504
  SettlingMax: 0.8444
    Overshoot: 1.3337
   Undershoot: 0
         Peak: 0.8444
     PeakTime: 0.3500
```

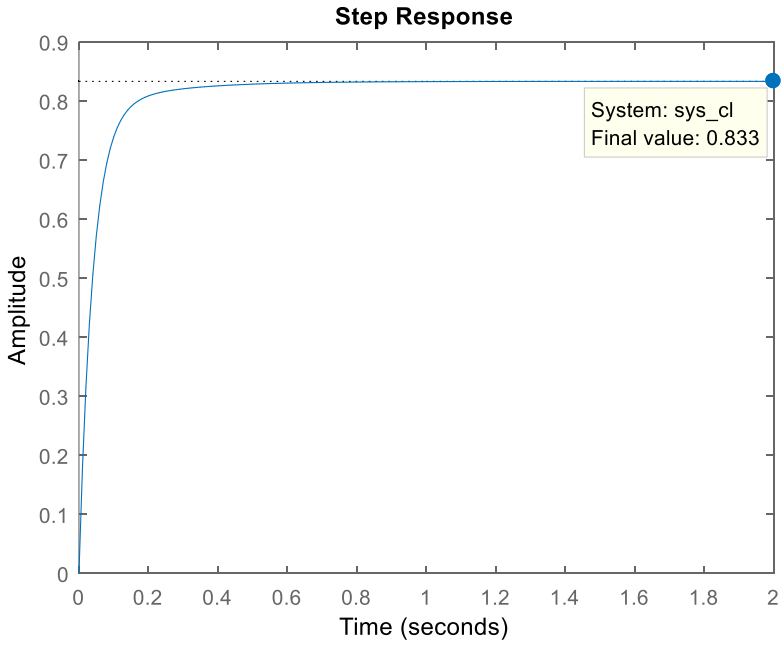


Steady state error = (1-0.833) / 0.833 = 20%

### $K_D = 20$:

**Results:**

```
      RiseTime: 0.1024
  SettlingTime: 0.2519
   SettlingMin: 0.7512
   SettlingMax: 0.8295
     Overshoot: 0
    Undershoot: 0
          Peak: 0.8295
      PeakTime: 0.5454
```

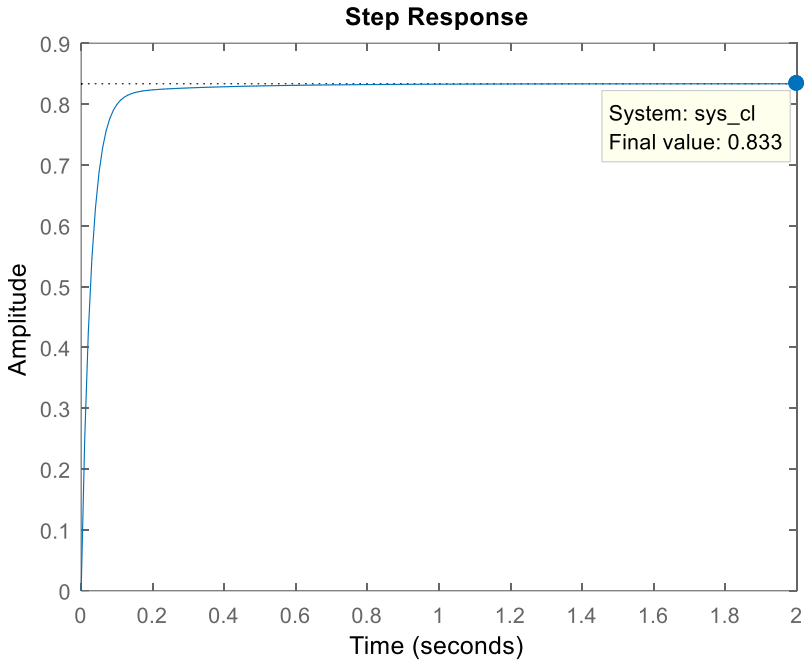



Steady state error = (1-0.833) / 0.833 = 20%

### $K_D = 30$:

**Results:**

```
     RiseTime: 0.0644
 SettlingTime: 0.1384
  SettlingMin: 0.7509
  SettlingMax: 0.8322
    Overshoot: 0
   Undershoot: 0
         Peak: 0.8322
     PeakTime: 0.8638
```




Steady state error = (1-0.833) / 0.833 = 20%

## Characteristics for increases $K_D$:

- Settling time decreased.
- Rise time decreased with small change.
- Steady state error remained same.
- Overshoot decreased (main factor).

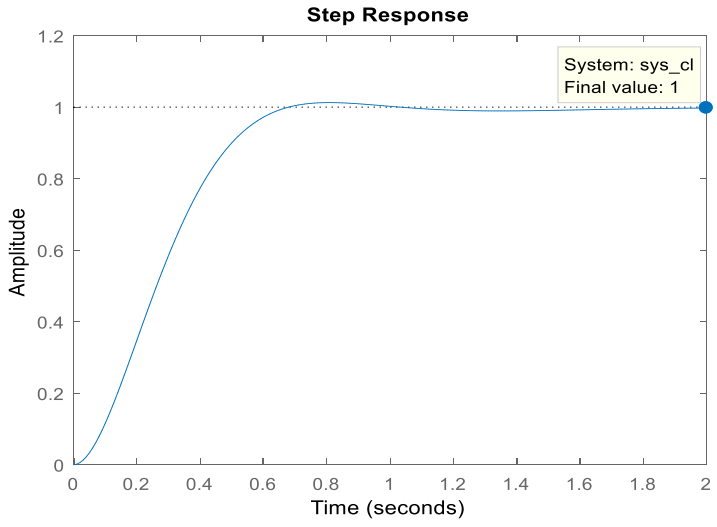
## Model representation and step response in M-file (with $K_p$=30 & $K_i$):

### $K_i$ = 70:

```
clear all;
close all;
num=[1];
den=[1 10 20];
sys=tf(num,den);
Kp=30;
Ki=70;
num1=[Kp Ki];
den1=[1 0];
contr=tf(num1,den1);
G=contr*sys;
sys_cl=feedback(G,1);
t=0:0.01:2;
step(sys_cl,t)
stepinfo(sys_cl)
```

**Results:**

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

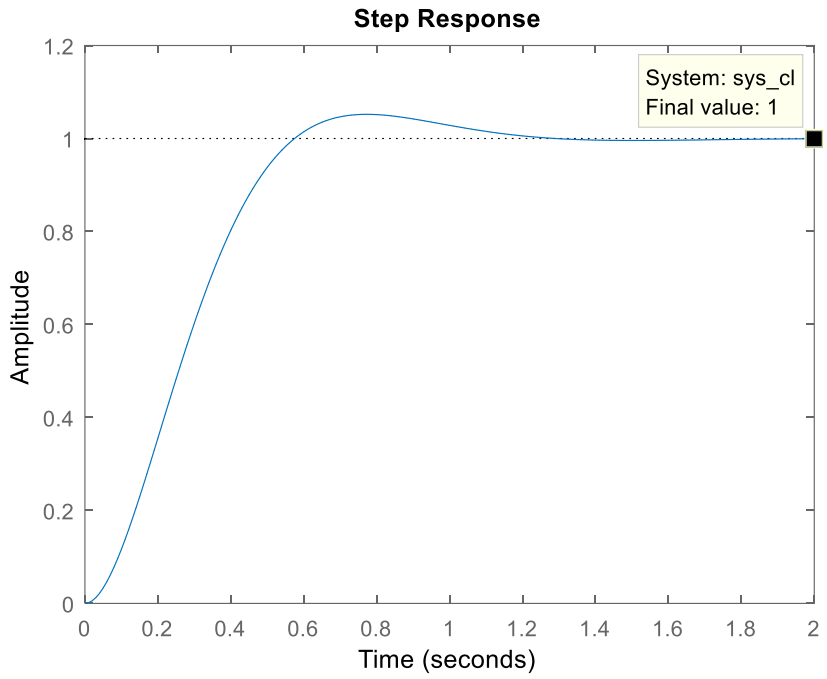


Steady state error = (1-1) / 1 = 0%

### $K_i$ = 80:

**Results:**

```
      RiseTime: 0.3752
  SettlingTime: 1.0613
   SettlingMin: 0.9078
   SettlingMax: 1.0520
     Overshoot: 5.1966
    Undershoot: 0
          Peak: 1.0520
      PeakTime: 0.7718
```




Steady state error = (1-1) / 1 = 0%

### Characteristics for increases $K_i$:

- Settling time increased.
- Rise time decreased.
- Steady state error eliminated.
- Overshoot increased.

## Statement:

These characteristics may not be accurate because of $K_p$, $K_i$ & $K_D$ depend on each-other. Changing two values at a time can change the results.

## Task:

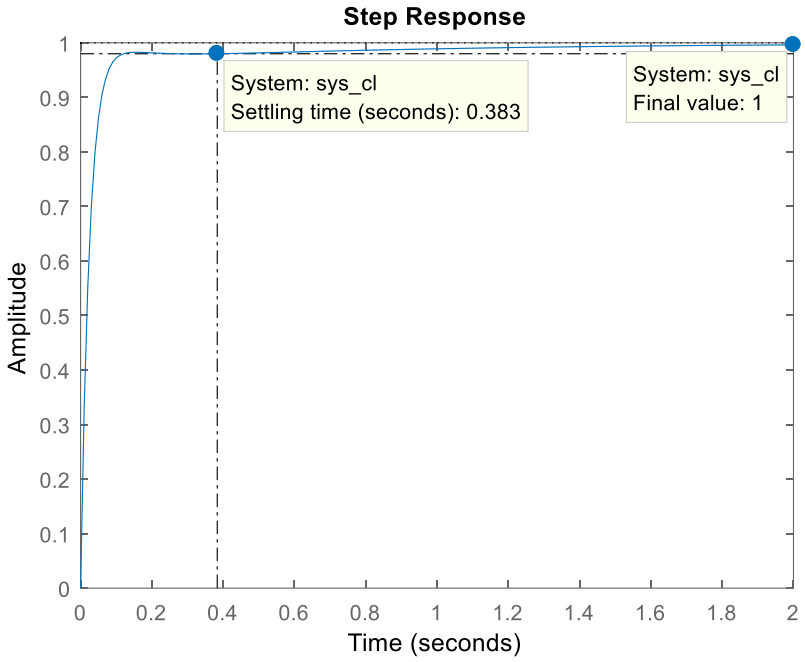
- Implement PID for desired output.

## Model representation and step response in M-file (with $K_p$, $K_i$ & $K_D$):

```
clear all;
close all;
num=[1];
den=[1 10 20];
sys=tf(num,den);
Kp=400;
Ki=400;
Kd=40;
num1=[Kd Kp Ki];
den1=[1 0];
contr=tf(num1,den1);
G=contr*sys;
sys_cl=feedback(G,1);
t=0:0.01:2;
step(sys_cl,t)
stepinfo(sys_cl)
```

**Results:**

```
    RiseTime: 0.0559
SettlingTime: 0.3835
 SettlingMin: 0.9023
 SettlingMax: 0.9848
   Overshoot: 0
  Undershoot: 0
        Peak: 0.9848
    PeakTime: 0.6954
```




Steady state error = (1-1) / 1 = 0%

## Lab # 2

### Objectives:

- Introduction to SISOT TOOL & Lead compensator design using root locus in sisotool.

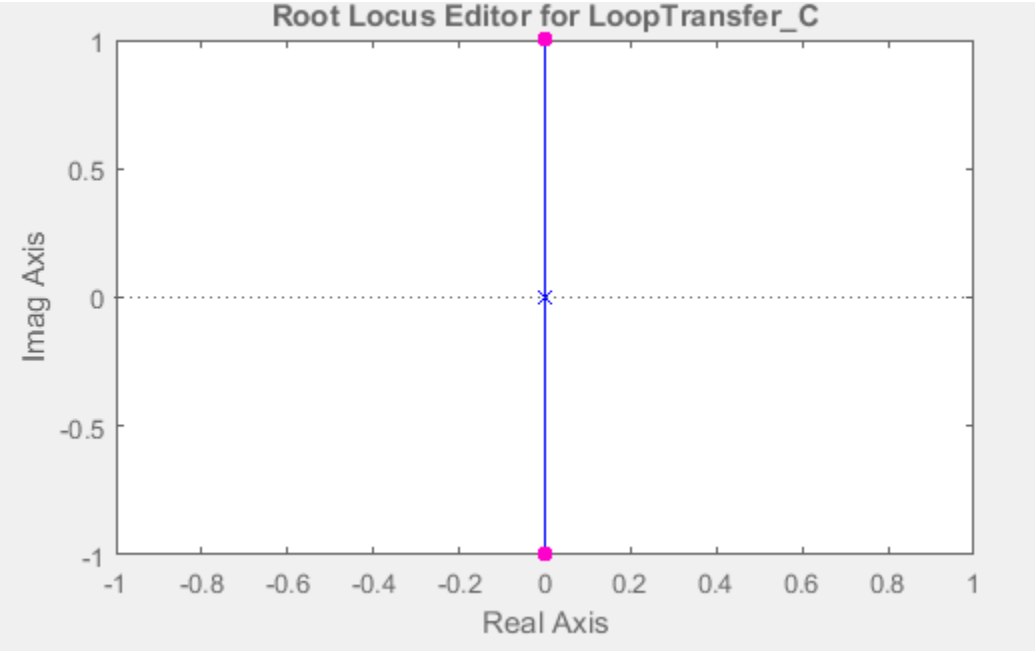
1. $G(s) = \frac{K}{s^2}$ Required damping ratio = 0.45 and $T_s$ = 4s. (Lab work)
2. $G(s) = \frac{K}{s(s+2)}$ Required damping ratio = 0.45 and $K_v$ = 20. (Task)

**$G(s) = \frac{K}{s^2}$ Required damping ratio = 0.45 and $T_s$ = 4s**

### Model representation and step response in M-file:

```
clear all;
close all;
num=[1];
den=[1 0 0];
g=tf(num,den)
sisotool('rlocus',g)
```

**SISO-TOOL:**




$\omega_d = \omega_n$ because damping ratio = 0

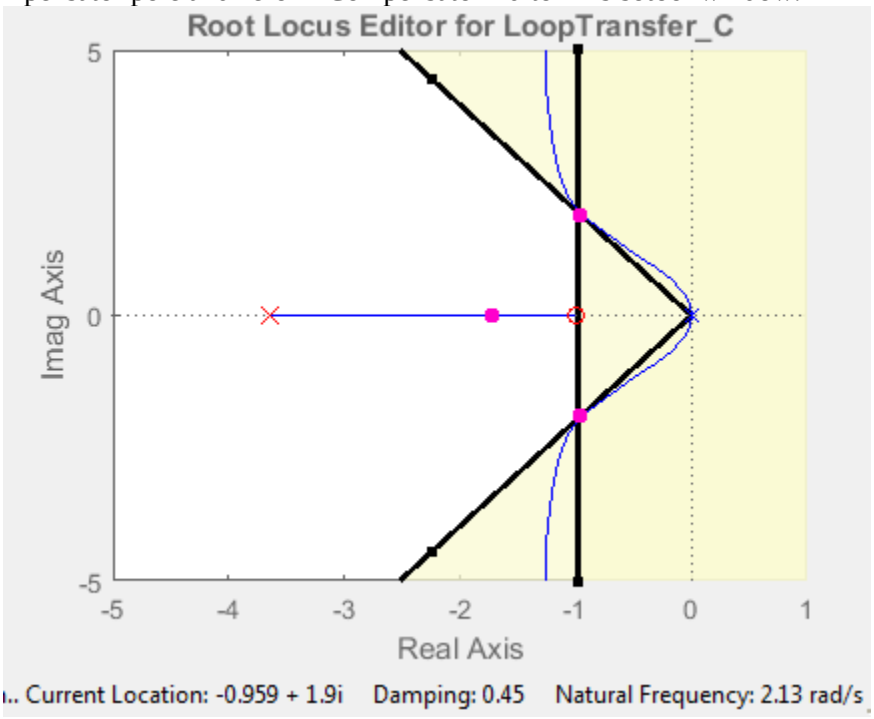
- Add values of Ts = 4s and damping ratio =0.45 in Design Requirement in sisotool window.

## Compensator:

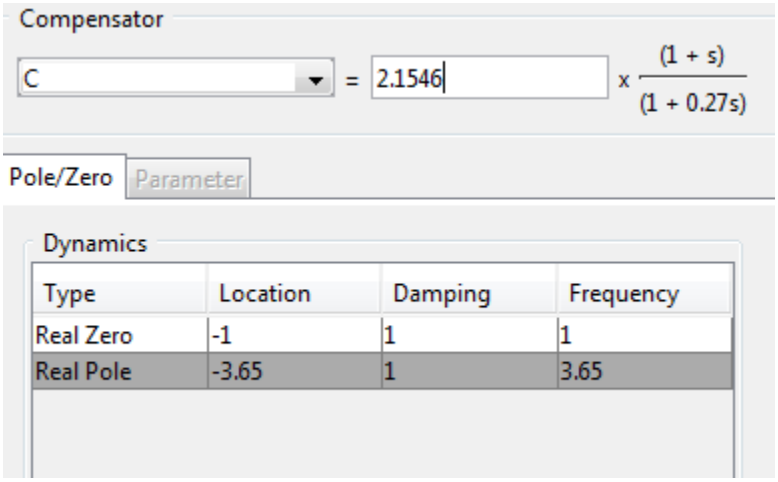
Suppose zero at -1

$P_c$ = -3.65 get value after angle contribution

- Add Compensator pole and zero in Compensator Editor in sisotool window.




Dominant Pole = -0.959 ± 1.9j




K = 2.1546 × (1/0.27)

K = 8

$$G_c(s) = \frac{8\,(S+1)}{(S+3.65)}$$

$L(s) = G_c(s) \times G(s)$

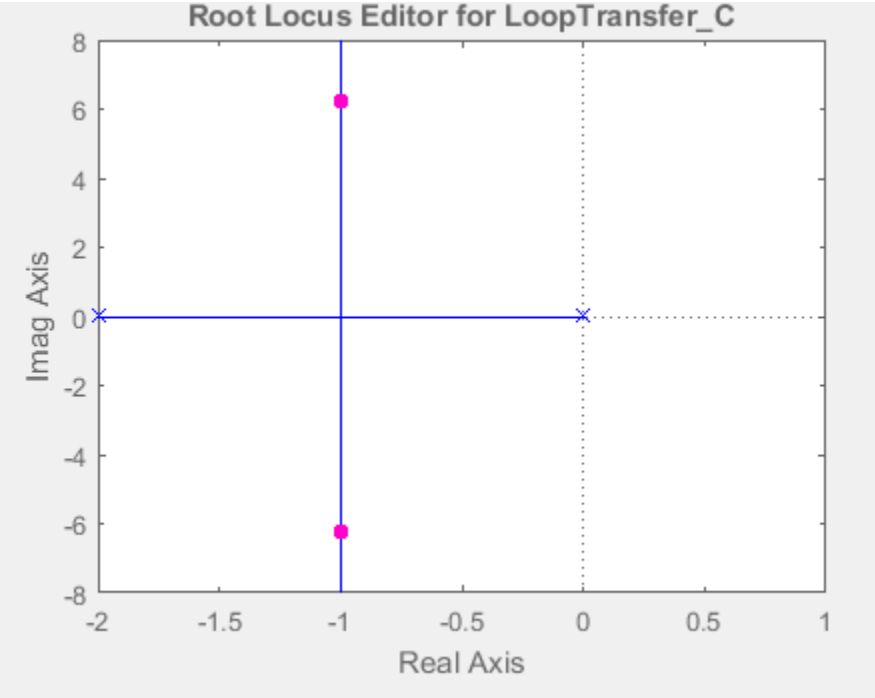
$$\boldsymbol{L(s) = 8\,\frac{(S+1)}{S^2\,(S+3.65)}}$$

$\boldsymbol{G(s) = \frac{K}{s(s+2)}}$ **Required damping ratio = 0.45 and $K_v$ = 20**

## Model representation and step response in M-file:

```
clear all;
close all;
num=[40];
den=[1 2 0];
g=tf(num,den);
sisotool('rlocus',g)
```

**SISO-TOOL:**




K = 40

$\omega_n = \sqrt{40}$

$$T(s) = \frac{40}{s(s+2)+40}$$

$$\zeta = 0.16$$

Ts =1s

- Add values of Ts = 1s and damping ratio =0.45 in Design Requirement in sisotool window.

## Compensator:

$$\zeta\, \omega_n = 4$$

$$\omega_n = \frac{4}{0.45}$$

$$\omega_n = 9$$

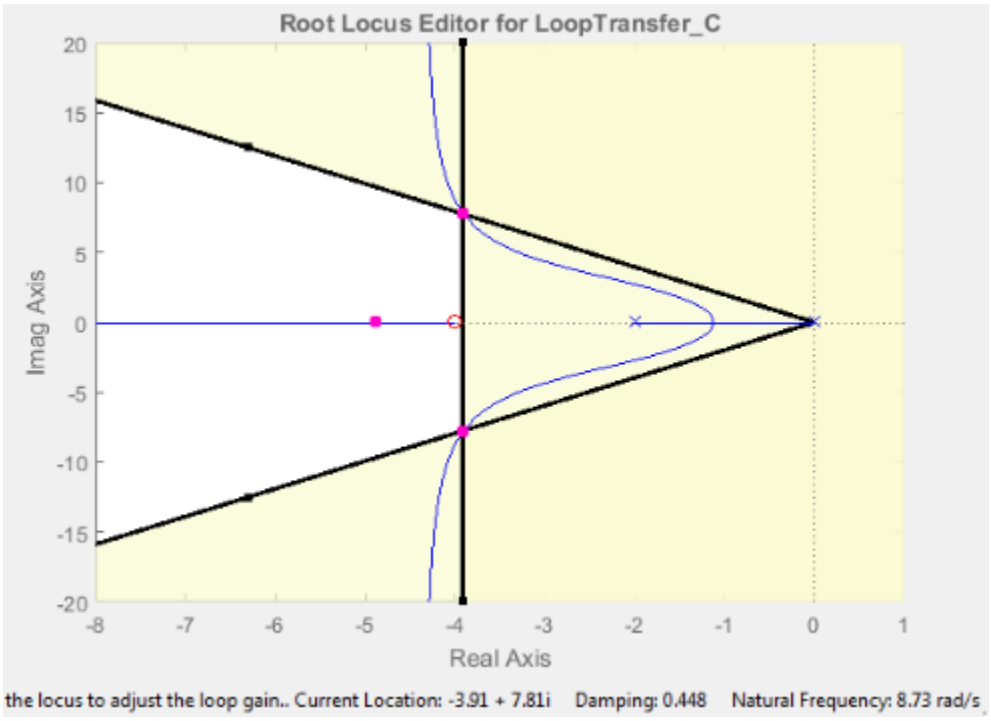
Suppose zero at -4

**Angle Contribution:**

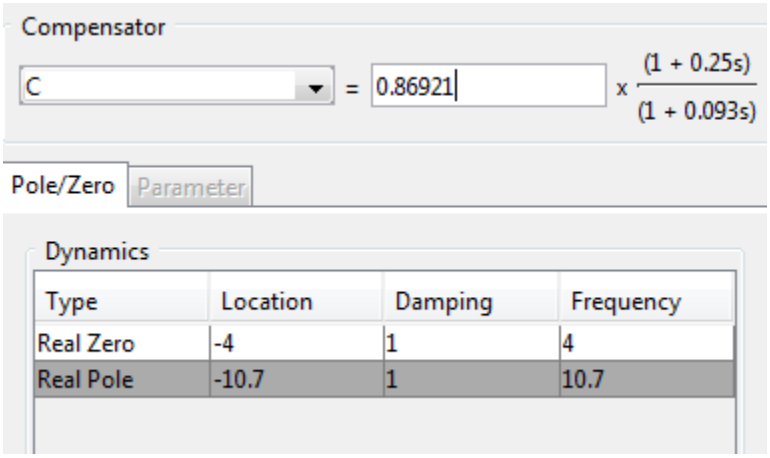
$$\Phi_p = 50 \text{ (deg.)}$$

$$P_c = -10.7$$

- Add Compensator pole and zero in Compensator Editor in sisotool window.



Dominant Pole = -3.91 ± 7.81j




K = 0.86921 (0.25/0.093)

K = 2.337

$$G_c(s) = \frac{2.337\ (S+4)}{(S+10.7)}$$

$L(s) = G_c(s) \times G(s)$

$$\boldsymbol{L(s) = 93.5\ \frac{(S+4)}{S(S+2)(S+10.7)}}$$

## Characteristic of pole and zero addition:

- Root locus goes on right side when pole added.
- Root locus goes on left side when zero added.

## Lab # 3

### Objectives:

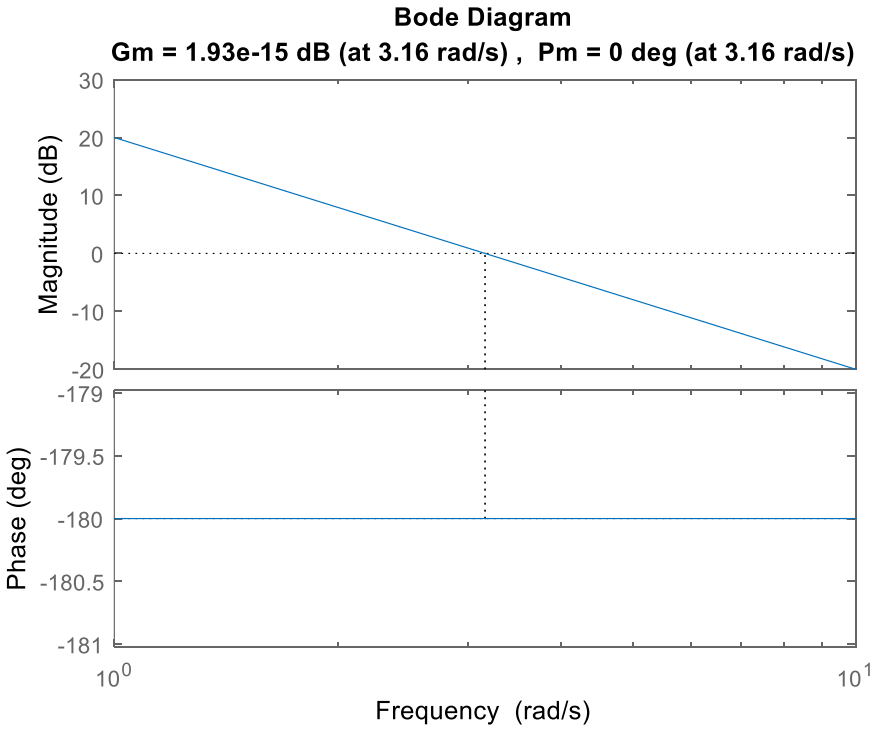
- Lead Compensator using Bode Plot $G(s) = \frac{10}{s^2}$ with the help of

1. bode command
2. sisotool.

## bode command

### Model representation and step response in M-file (without Compensator):

```
clear close all;
num=[10];
den=[1 0 0];
g=tf(num,den);
bode(g)
margin(g)
```




### Lead Compensator:

$$G_{c(s)} = \frac{1}{a}\frac{(1+a\tau s)}{(1+\tau s)}$$

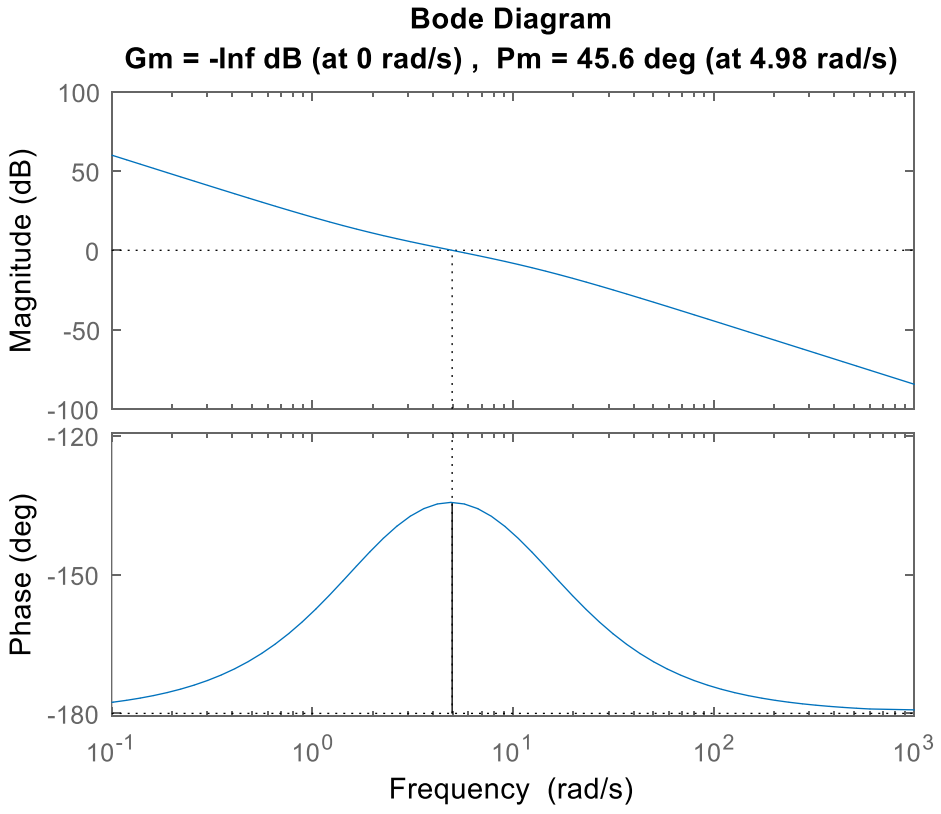
$$a = 6,\ a\tau = \frac{1}{2},\ \tau = \frac{1}{12}$$

$$G_{c(S)} = \frac{(s+2)}{(s+12)}$$

$$G_{T(S)} = a \times G_{c(S)} \times G(s) = 6 \times \frac{10(s+2)}{s^2(s+12)}$$

## Model representation and step response in M-file (with Lead Compensator):

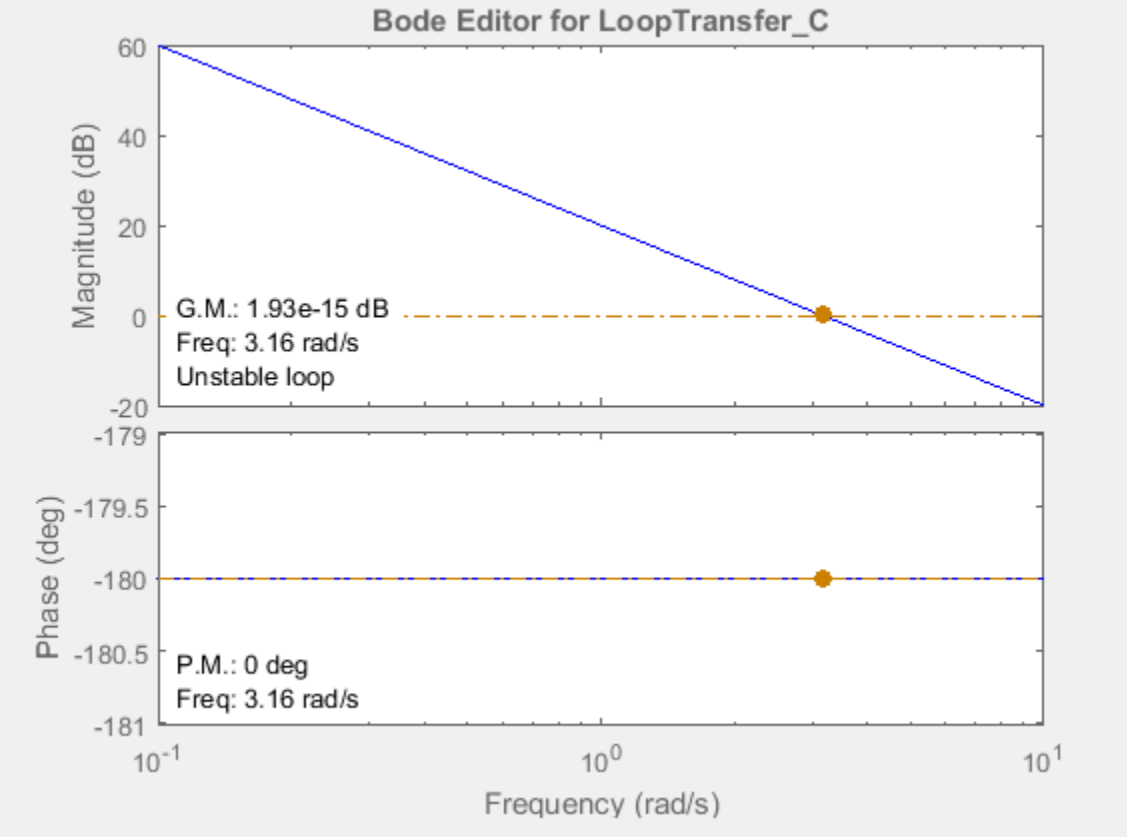
```
clear close all;
num=[10];
den=[1 0 0];
g=tf(num,den);
num1=[1 2];
den1=[1 12];
contr=tf(num1,den1);
G=g*contr;
G=6*G;
bode(G)
margin(G)
```



## sisotool command

### Model representation and step response in M-file :

```
clear close all;
num=[10];
den=[1 0 0];
g=tf(num,den);
sisotool('bode',g)
```




- Add Phase Margin > 45° and unchecked Gain margin in Design Requirement in sisotool window.
- Add Lead Compensator (Phase = 45° at frequency 4.98 rad/sec) in Compensator Editor in sisotool window.

Compensator

C = 1 x $\frac{(1 + 0.48s)}{(1 + 0.083s)}$

Pole/Zero | Parameter

Dynamics

| Type | Location | Damping | Frequency |
|---|---|---|---|
| Lead | -2.06, -12 | 1 | 2.06, 12 |

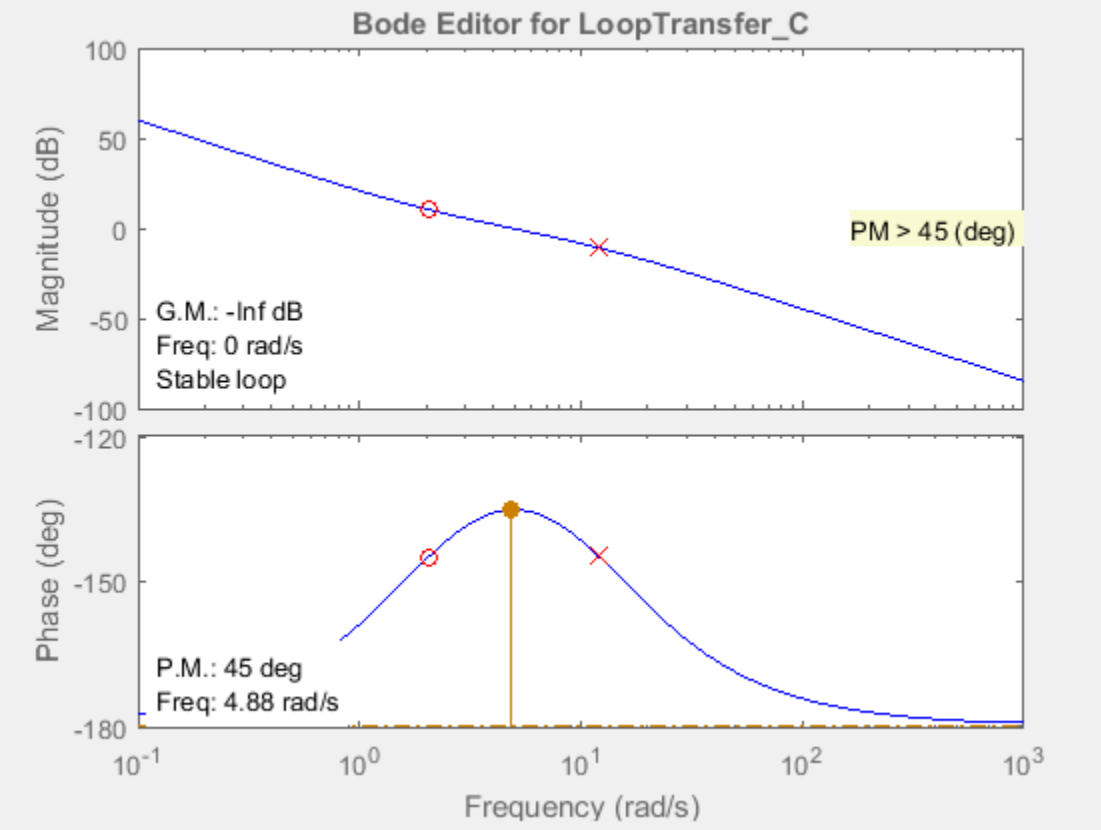
Edit Selected Dynamics

Real Zero: -2.0628

Real Pole: -12.023

Max Delta Phase (deg): 45

at Frequency: 4.98



## Statement:

Gain margin is –inf dB because it's phase didn't cut -180°.

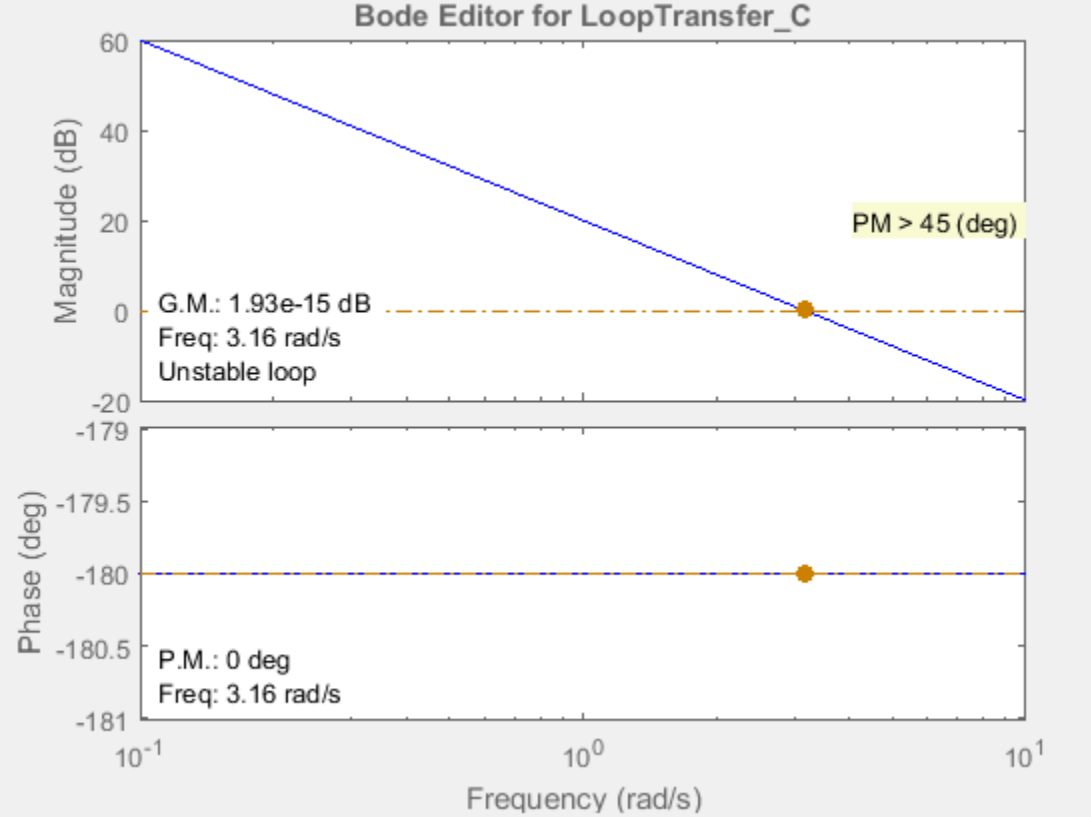
## Task:

- If Phase margin is given (45°) but frequency that point isn't given then how to find frequency that point?

### Model representation and step response in M-file :

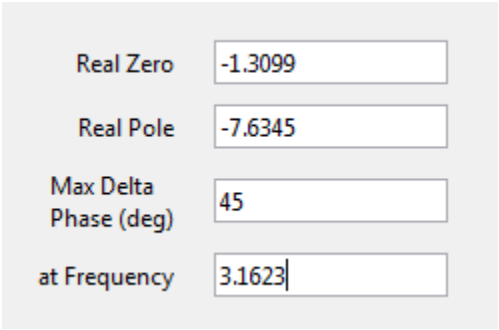
```
clear close all;
num=[10];
den=[1 0 0];
g=tf(num,den);
sisotool('bode',g)
```

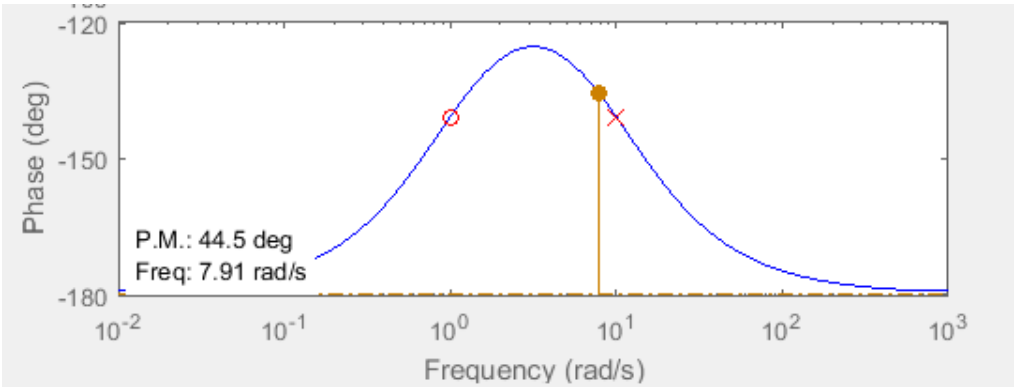
1) Add Phase Margin > 45° and unchecked Gain margin in Design Requirement in sisotool window.

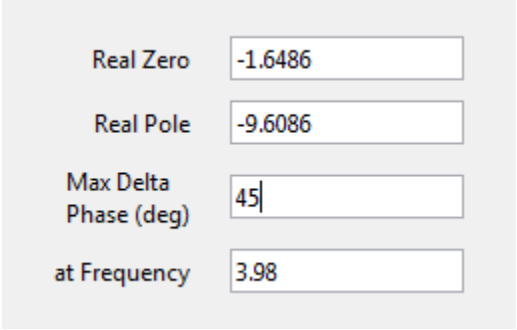


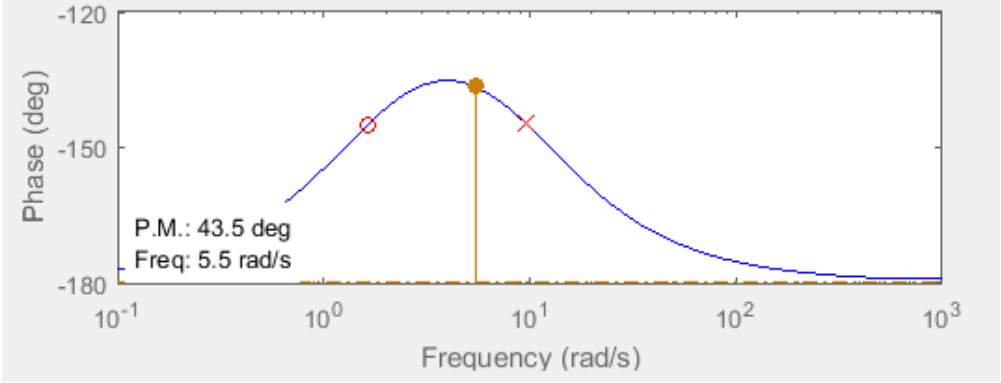

- 0dB occur at frequency 3.16 rad/sec. So, our desired frequency should greater than 3.16 rad/sec.

- Add Lead Compensator (change frequency (>3.16 rad/sec) until you get desired Phasemargin which 45° ) in Compensator Editor in sisotool window.

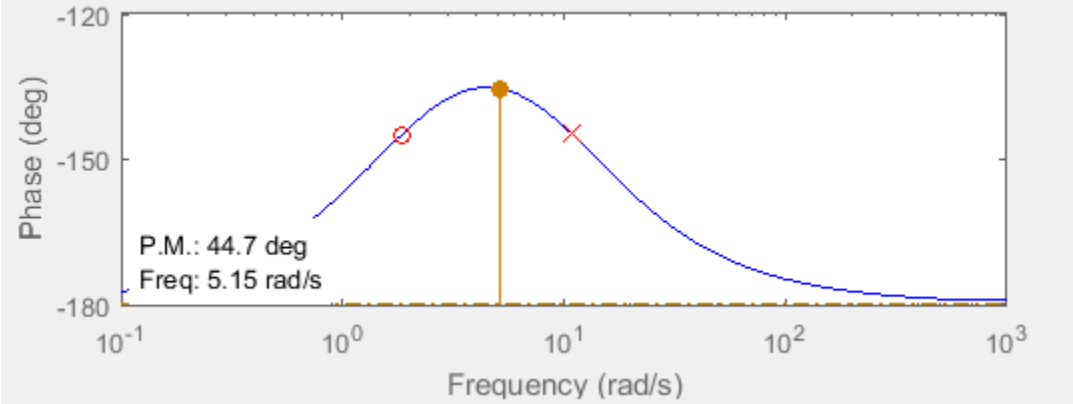


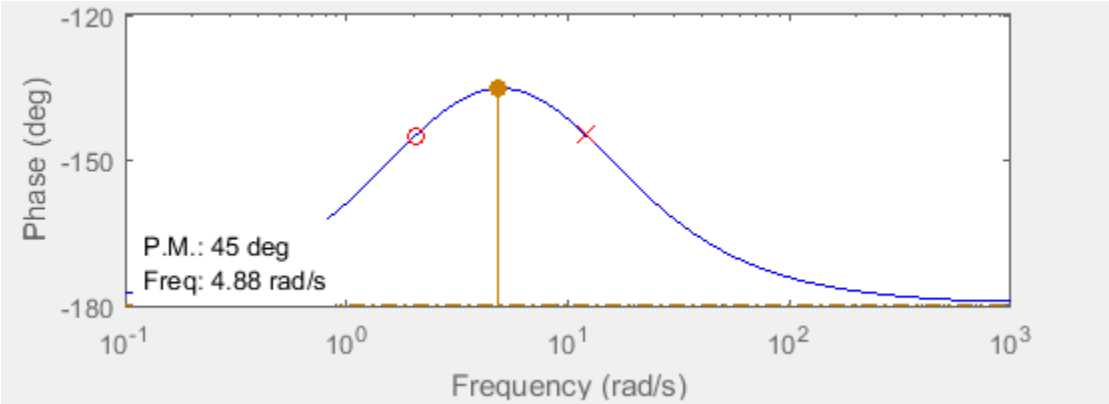



| | |
|---|---|
| Real Zero | -1.864 |
| Real Pole | -10.864 |
| Max Delta Phase (deg) | 45 |
| at Frequency | 4.5 |




| | |
|---|---|
| Real Zero | -2.0628 |
| Real Pole | -12.023 |
| Max Delta Phase (deg) | 45 |
| at Frequency | 4.98 |



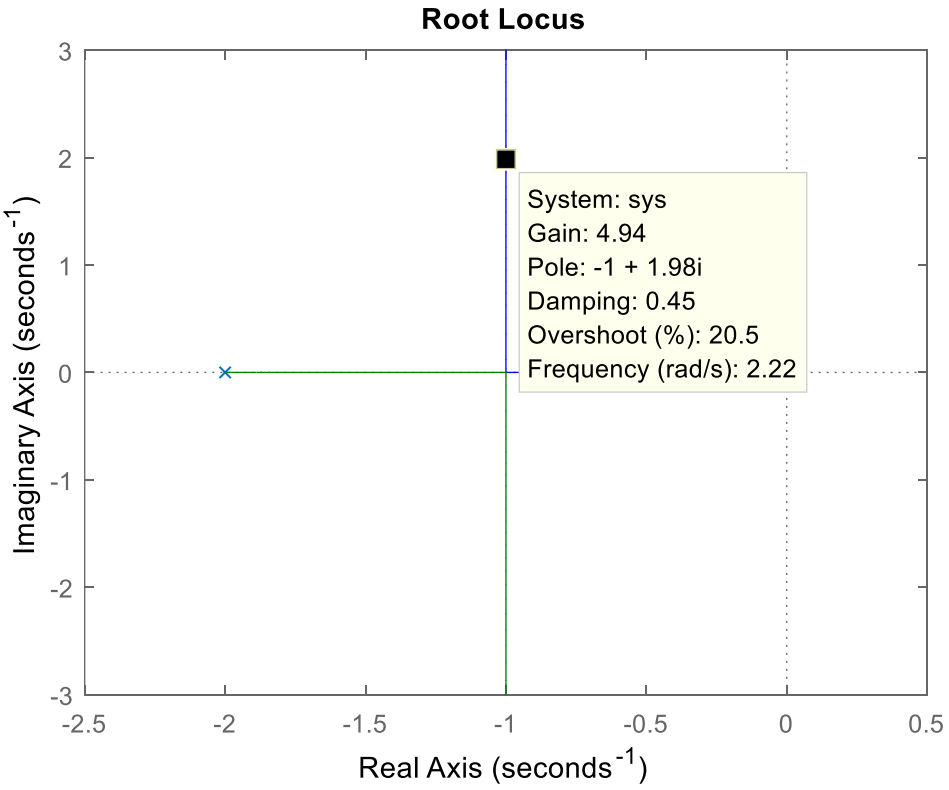
## Lab # 4

### Objectives:

- Lag Compensator using Root locus $G(s) = \frac{K}{s(s+2)}$.
- Design Specifications:
  1. ʒ = 0.45
  2. $K_V$ = 20

### Model representation and step response in M-file (k=1):

```
clear close all;
num=[1];
den=[1 2 0];
sys=tf(num,den);
sys_cl=feedback(sys,1)
stepinfo(sys_cl)
rlocus(sys)
```




- It is only used to check whether desired damping ratio exist in its pole trajectory or not.

### Model representation and step response in M-file (k=4.94):

```
clear close all;
num=[4.94];
den=[1 2 0];
sys=tf(num,den);
sys_cl=feedback(sys,1)
stepinfo(sys_cl)
rlocus(sys)
```

- By changing gain, we only can achieve the desire damping ratio and we can't achieve desired $K_V$.

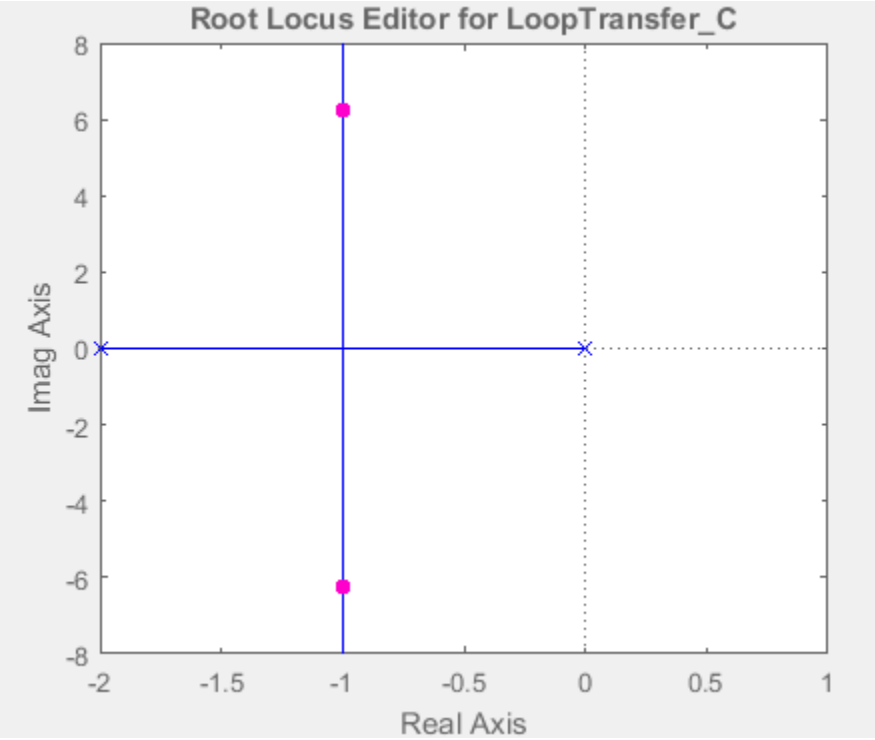
$$K_V = 20 = \lim_{s\to 0} sG(s) = k/2$$

K=40

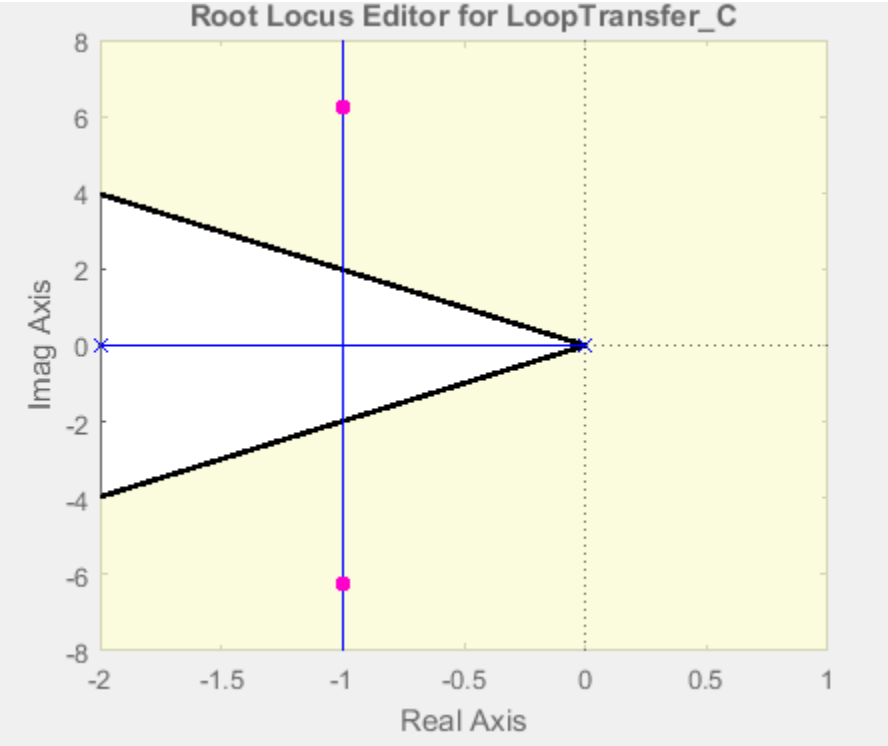
## With the help of sisotool

### Model representation and step response in M-file (k=40):

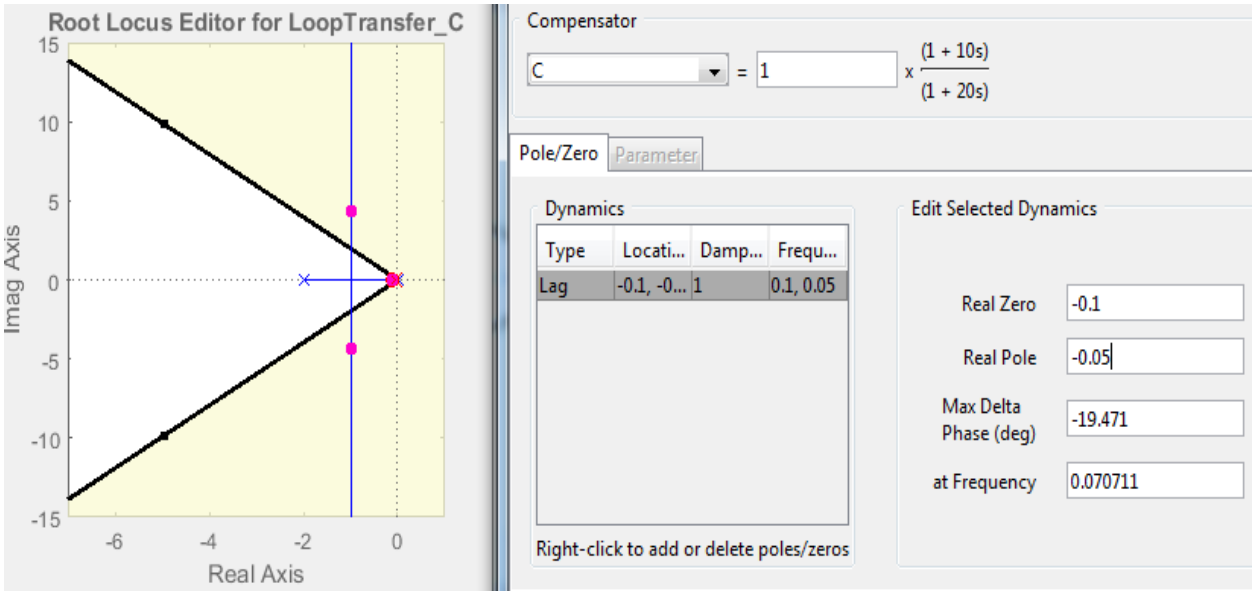
```
clear close all;
num=[40];
den=[1 2 0];
sys=tf(num,den);
sisotool('rlocus',sys)
```



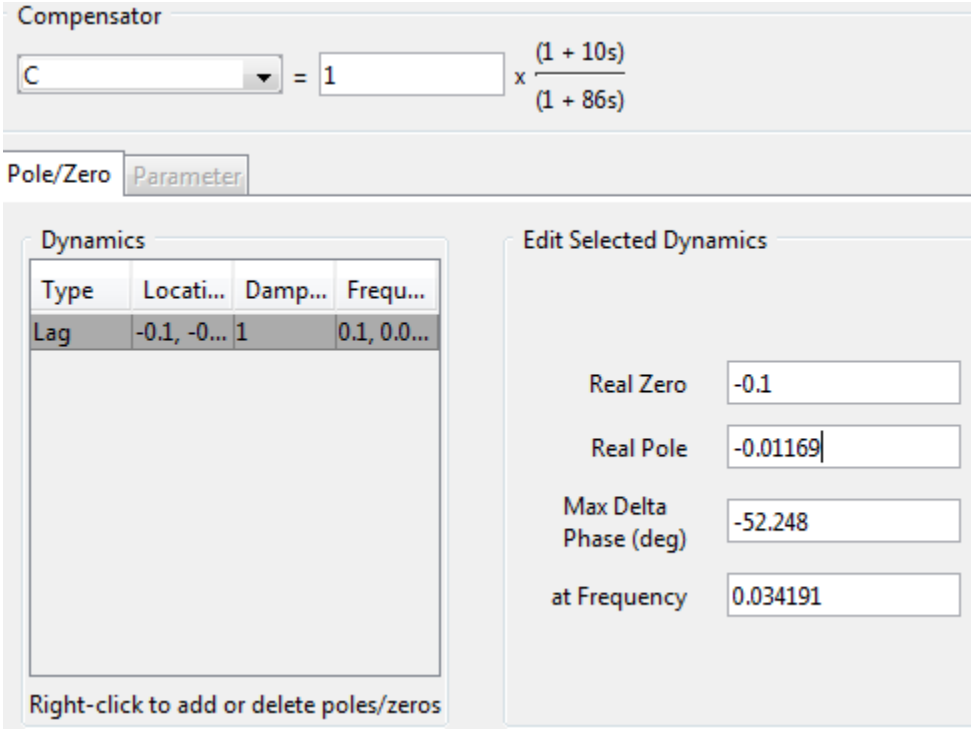

- Add damping ratio = 0.45 in design Requirements in sisotool window.

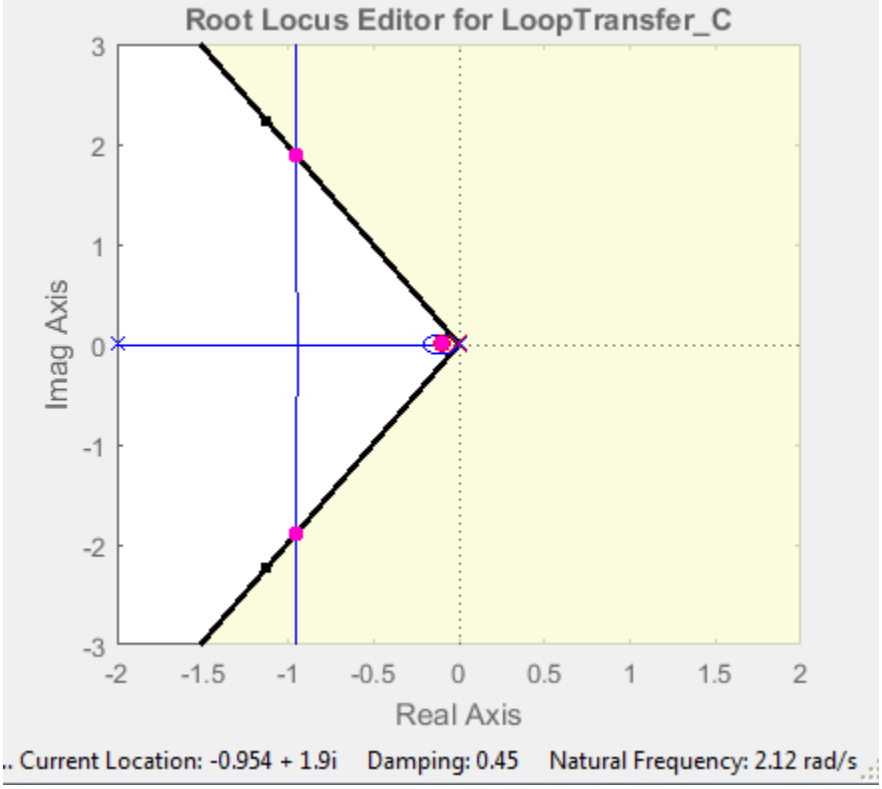



- Add Lag Compensator (suppose Zero $Z_C$ = -0.1 and **abs($P_C$)<abs($Z_C$)**) in Compensator Editor in sisotool window.




- Change the values of pole until point movie on intersection point and that is desired gain.




- Desired pole is at -0.01169 where $K_V = 20$ and $\zeta = 0.45$.

$$G_C(s) = \frac{(1+10s)}{(1+86s)} = \frac{1}{8.6}\frac{(s+\frac{1}{10})}{(s+\frac{1}{86})}$$

$$G_C(s) \times G(s) = \frac{40}{8.6}\frac{(s+\frac{1}{10})}{s(s+2)(s+\frac{1}{86})}$$

$$\boldsymbol{G_C(s) \times G(s) = \frac{4.7(s+\frac{1}{10})}{s(s+2)(s+\frac{1}{86})}}$$

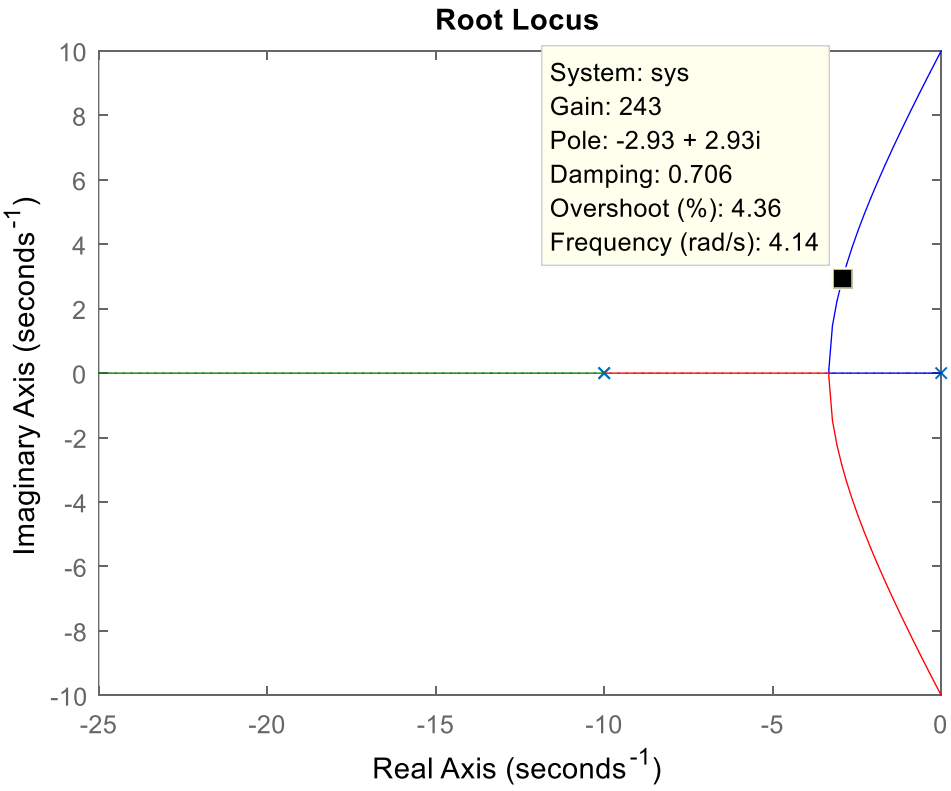
$$\boldsymbol{gain \approx 5}$$

## Task:

- Lag Compensator using Root locus $G(s) = \frac{K}{s(s+10)^2}$.
- Design Specifications:
  1. $\zeta = 0.707$
  2. $K_V = 20$

## Model representation and step response in M-file (k=1):

```
clear close all;
num=[1];
den=[1 20 100 0];
sys=tf(num,den);
sys_cl=feedback(sys,1)
stepinfo(sys_cl)
rlocus(sys)
```



$K_V = 20 = \lim_{s\to 0} sG(s) = k/100$

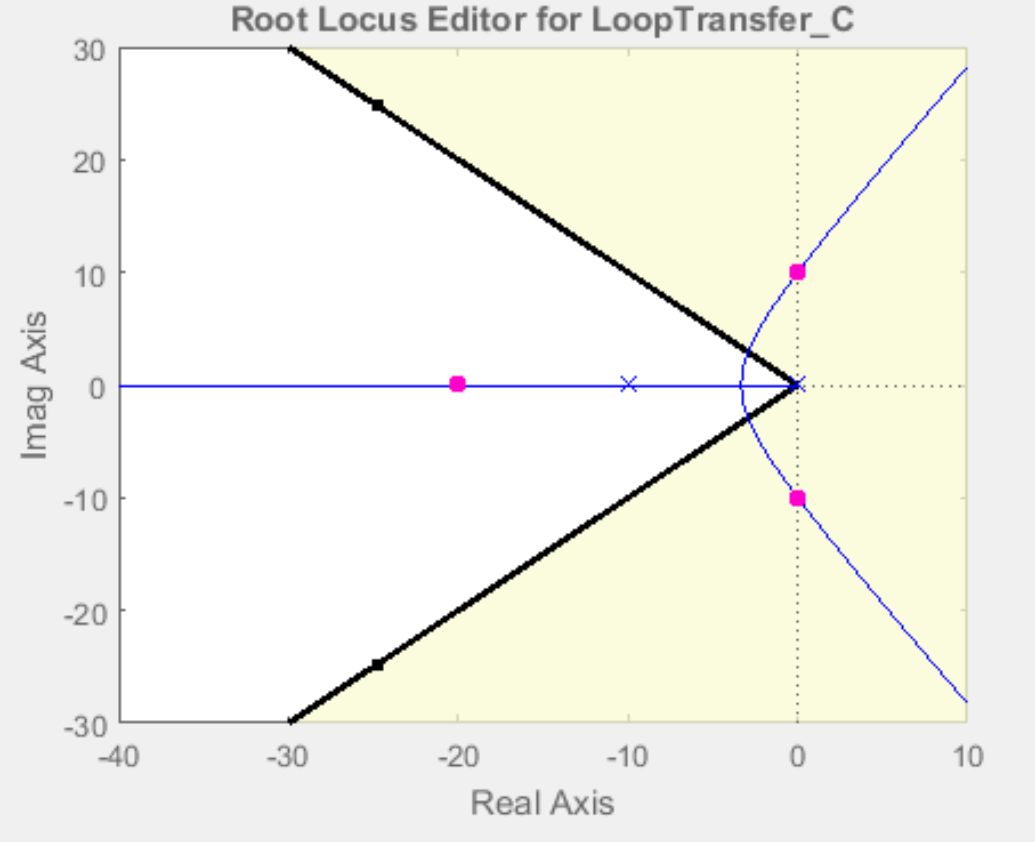
K=2000

## With the help of sisotool

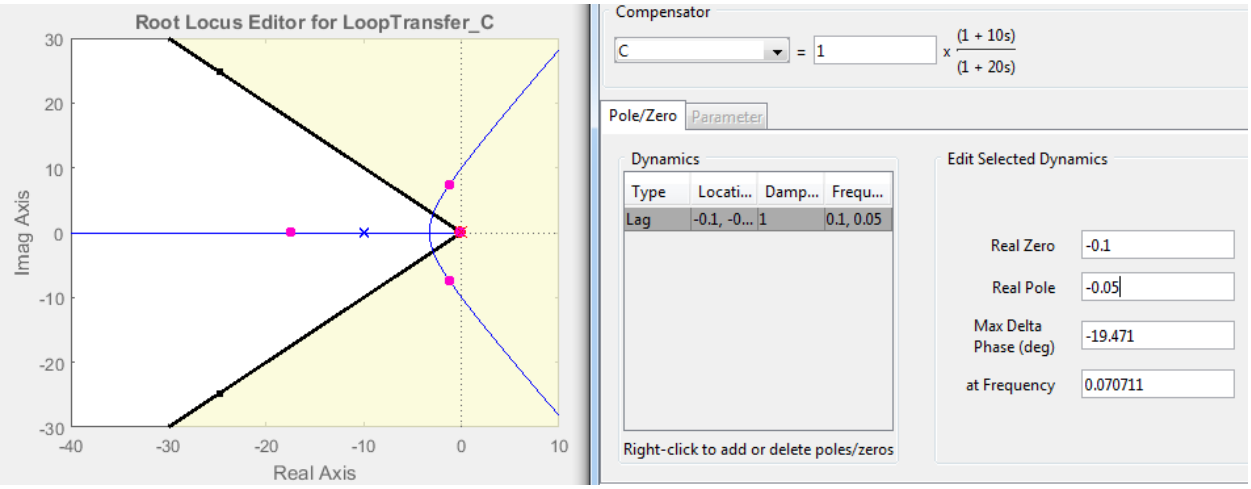
### Model representation and step response in M-file (k=2000):

```
clear close all;
num=[2000];
den=[1 20 100 0];
sys=tf(num,den);
sisotool('rlocus',sys)
```

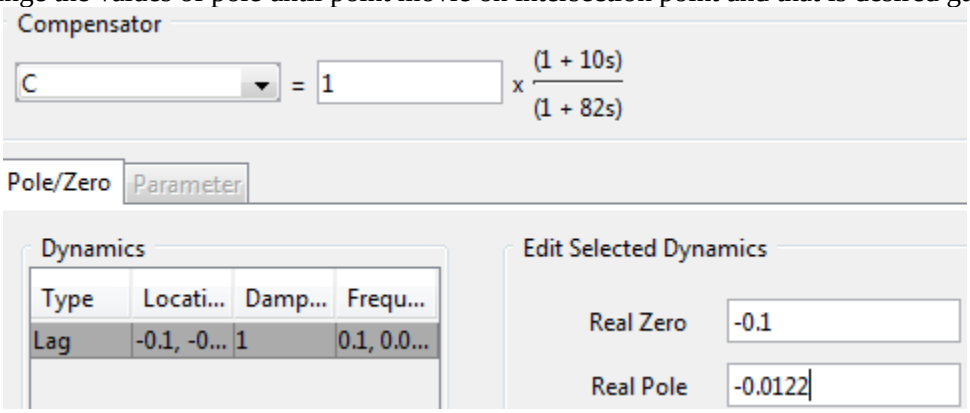
- Add damping ratio = 0.707 in design Requirements in sisotool window.

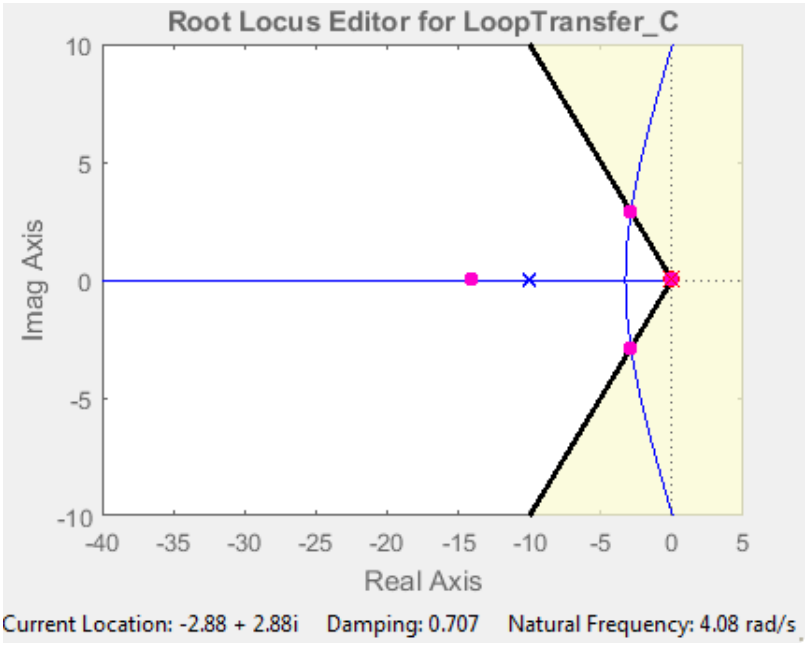



- Add Lag Compensator (suppose Zero $Z_C$ = -0.1 and **abs($P_C$)<abs($Z_C$)**) in Compensator Editor in sisotool window.




- Change the values of pole until point movie on intersection point and that is desired gain.



- Desired pole is at -0.0122where $K_V = 20$ and $\zeta = 0.707$.

$$G_C(s) = \frac{(1+10s)}{(1+82s)} = \frac{1}{8.2}\frac{(s+\frac{1}{10})}{(s+\frac{1}{82})}$$

$$G_C(s) \times G(s) = \frac{2000}{8.2}\frac{(s+\frac{1}{10})}{s(S+10)^2(s+\frac{1}{82})}$$

$$\boldsymbol{G_C(s) \times G(s) = \frac{243.9(s+\frac{1}{10})}{s(s+2)(s+\frac{1}{82})}}$$

$$\boldsymbol{gain \approx 244}$$

# Lab # 5

## Objectives:

- System design in sisotool using Integration Network

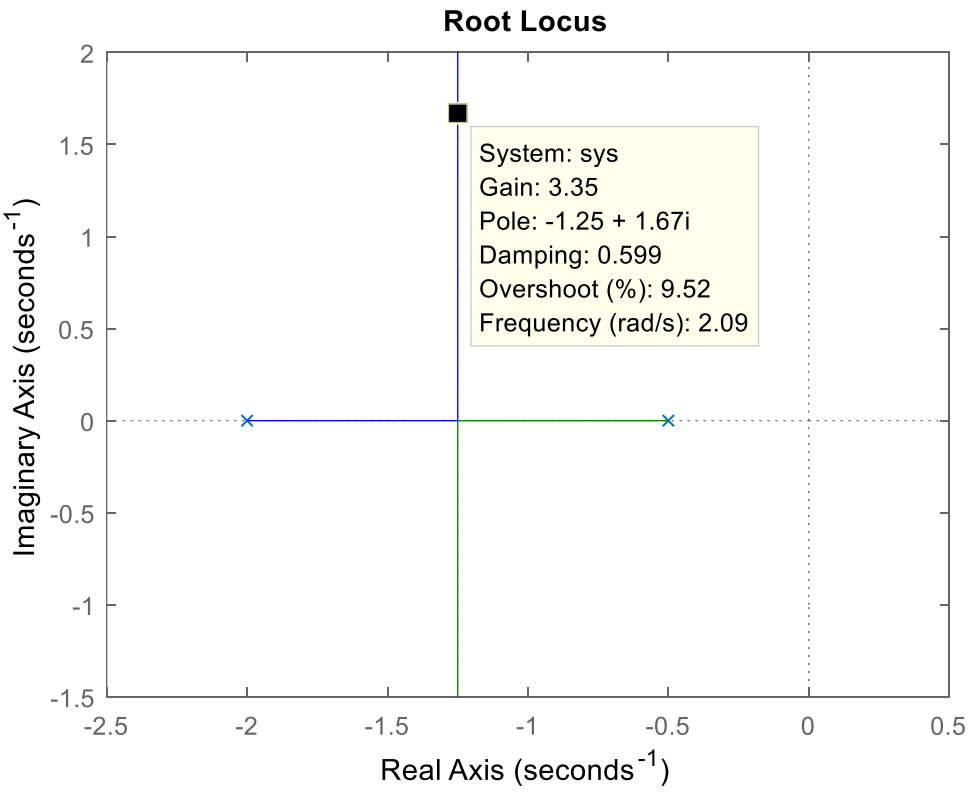
$$\text{Plant } G(s) = \frac{K}{(s+0.5)(s+2)}$$

$$G_c(s) = \frac{K_p\,(s + \frac{K_I}{K_P})}{s}$$

- Design specifications are damping ratio = 0.6 and real part of dominant pole = -0.75.

## Model representation and step response in M-file (k=1):

```
clear close all;
num=[1];
den=[1 2.5 1];
sys=tf(num,den);
stepinfo(sys)
rlocus(sys)
```




$\zeta \times \omega_n = 0.75$

$$\omega_n = \frac{0.75}{0.6} = 1.25$$

$$T_s = \frac{4}{0.75} = 5.33s$$

### Question:

Is it possible to achieve damping ratio = 0.6 without reshaping the root locus (without changing the design)?

### Answer:

Yes, we can achieve damping ratio = 0.6 without changing root locus at gain 3.35.

## With the help of sisotool

### Model representation and step response in M-file (k=1):

```
1 -    clear close all;
2 -    num=[1];
3 -    den=[1 2.5 1];
4 -    sys=tf(num,den);
5 -    sisotool('rlocus',sys)
```

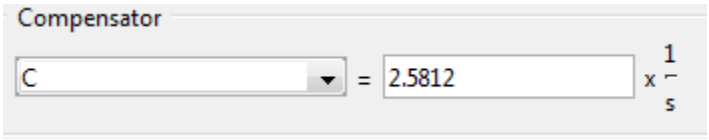
- Add damping ratio = 0.6 and Settling time = 5.33s in design Requirements in sisotool window.
- Add Integrator at location 0 (because compensator pole is at 0 location) in Compensator Editor in sisotool window.

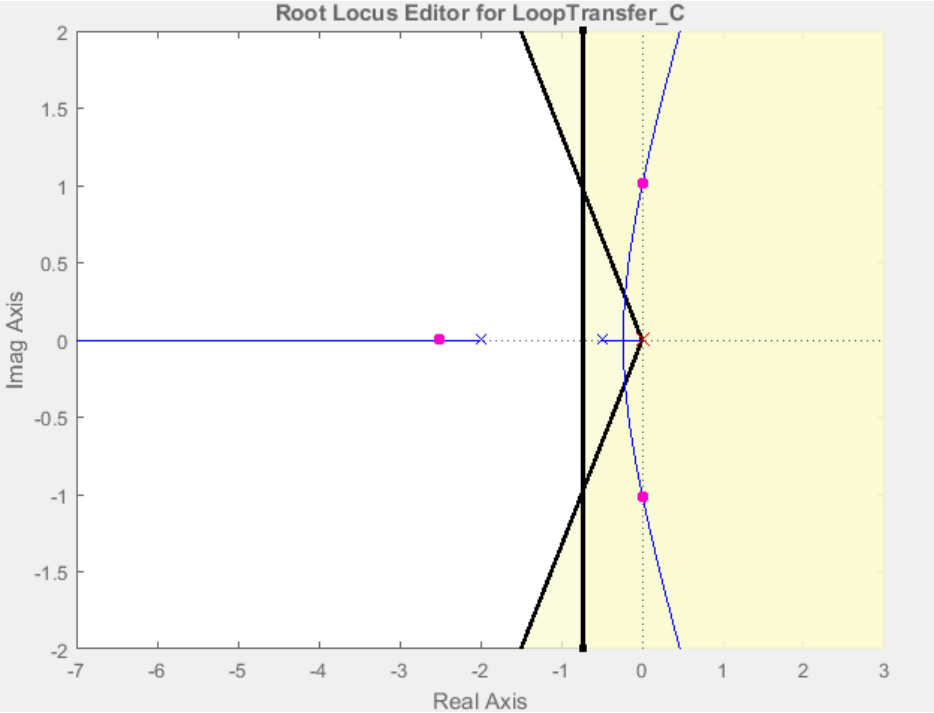
### Question:

What is the value of gain which makes system marginally stable oblige undamped?

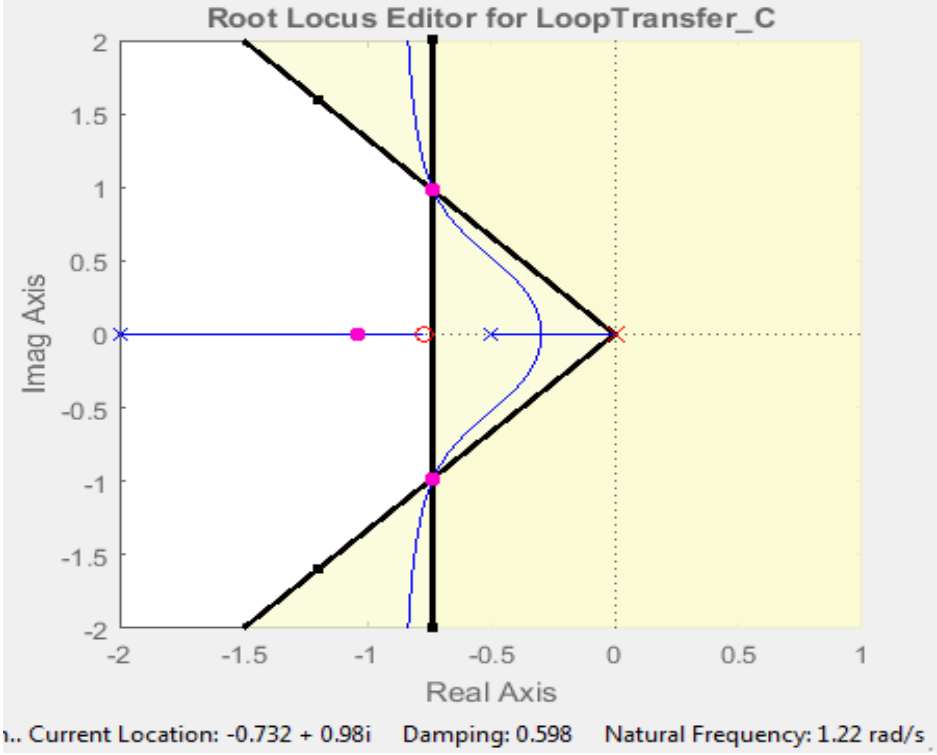
### Answer:

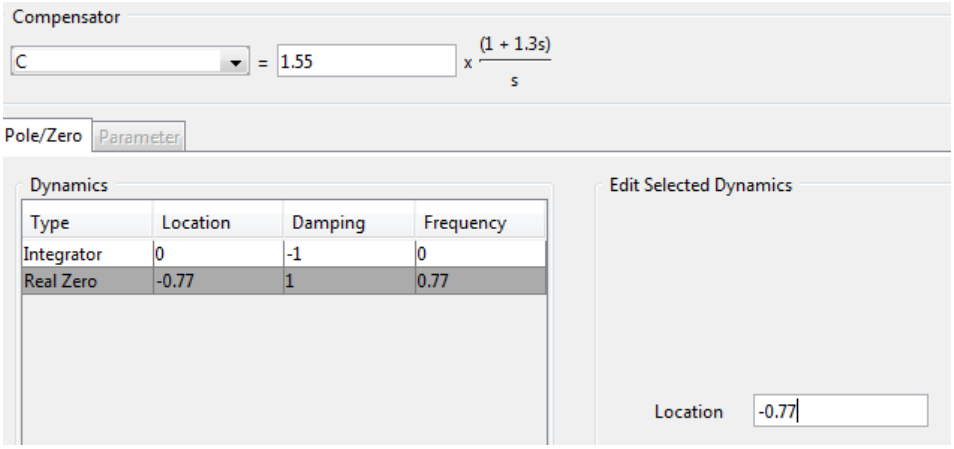
At gain 2.812, system becomes marginally stable.






- Add real zero in Compensator Editor in sisotool window. And changed the value where you get desired response.






Desired dominant poles = $-0.732 \pm 0.98i$

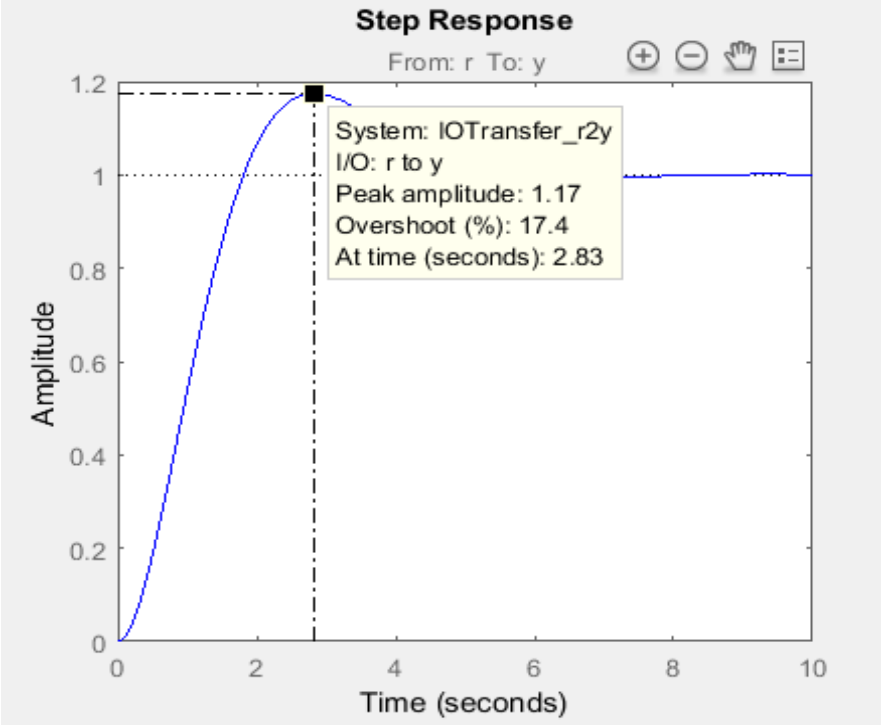
$$G_c(s) = 1.55\ \frac{1.3(s+\frac{1}{1.3})}{s}$$

$$G_c(s) = 2.015\ \frac{s+0.769}{s}$$

$$T(s) = G(s) \times G_c(s)$$

$$T(s) = \frac{2.002s+1.541}{s^3+2.5s^2+3.002s+1.541}$$

- %OS (%OS = 17.4%) increased from desired value (<10%) due to addition of infinite zero in closed loop transfer function of a compensator system.



- Add pre-filter to cancel out the effect of zero.

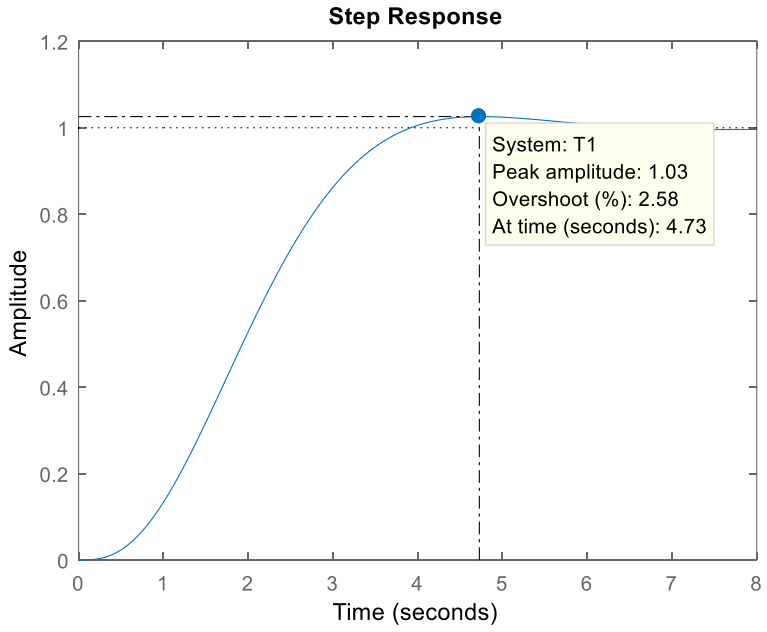
$$G_p(s) = \frac{0.77}{s + 0.77}$$

$$T_1(s) = T(s) \times G_p(s)$$

$$T_1(s) = \frac{1.541s+1.187}{s^4+3.27s^3+4.926s^2+3.852s+1.187}$$

## Model representation and step response in M-file:

```
clear close all;
num=[1];
den=[1 2.5 1];
sys=tf(num,den);
num1=[1 0.77];
den1=[1 0];
Gc=2.0015 * tf(num1,den1)
T=feedback(Gc*sys,1)
num2=[0.77];
den2=[1 0.77];
Gp=tf(num2,den2)
T1=Gp*T
step(T1)
```




- Hence, by adding pre-filter %OS become less than 10%.

# Lab # 6

## Objectives:

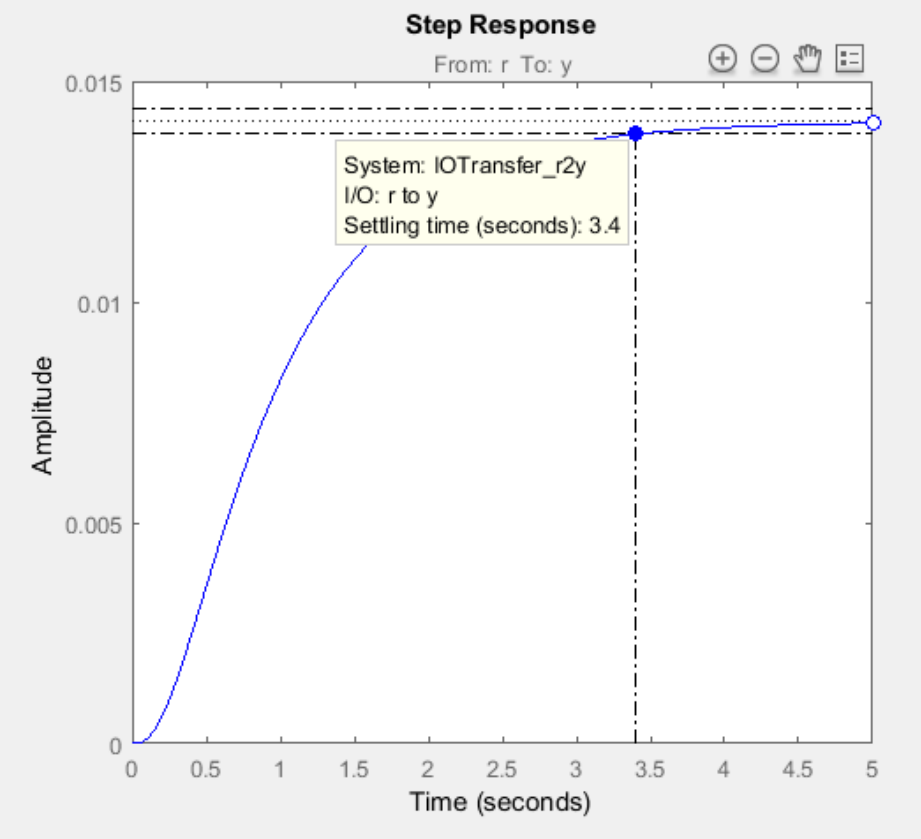
- System design in sisotool using Lead-lag compensator

$$\text{Plant } G(s) = \frac{1}{s^3+17s^2+76s+70}$$

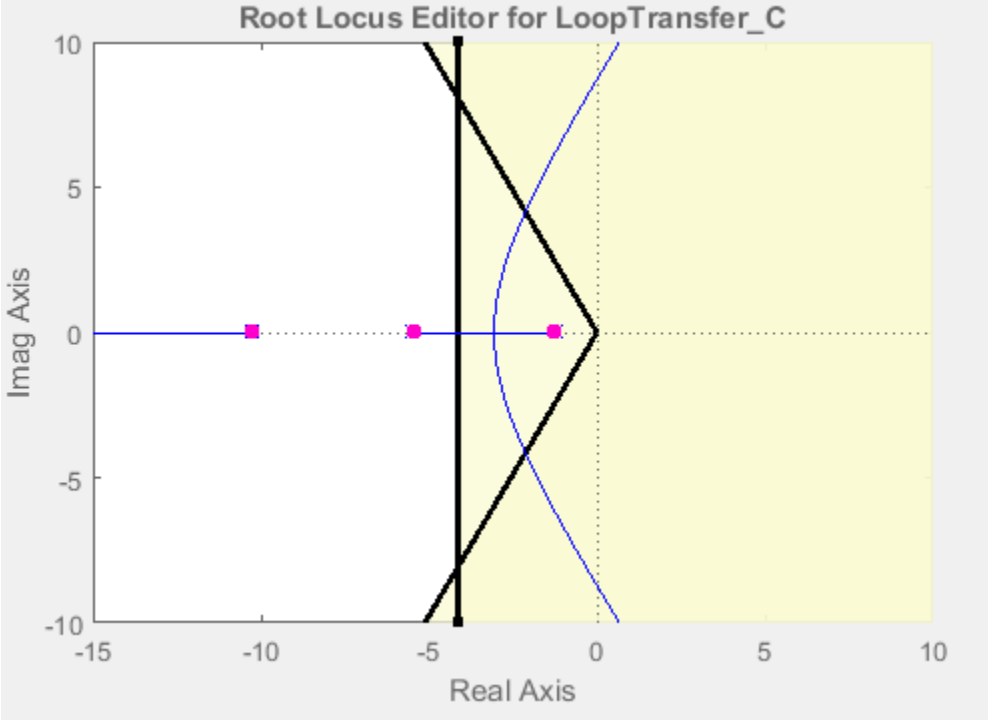
- Design specifications are
  i. % OS < 20%
  ii. Settling time = $t_s$ = 0.95s.

## Model representation and step response in M-file:

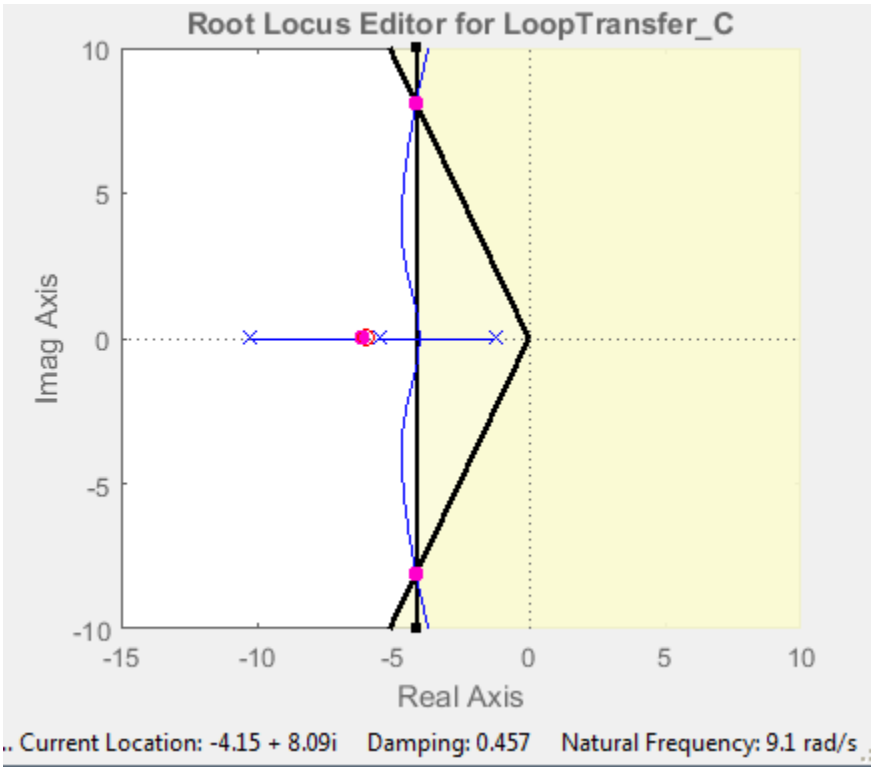
```
clear all;
close all;
num=[1];
den=[1 17 76 70];
g=tf(num,den);
sisotool('rlocus',g)
```

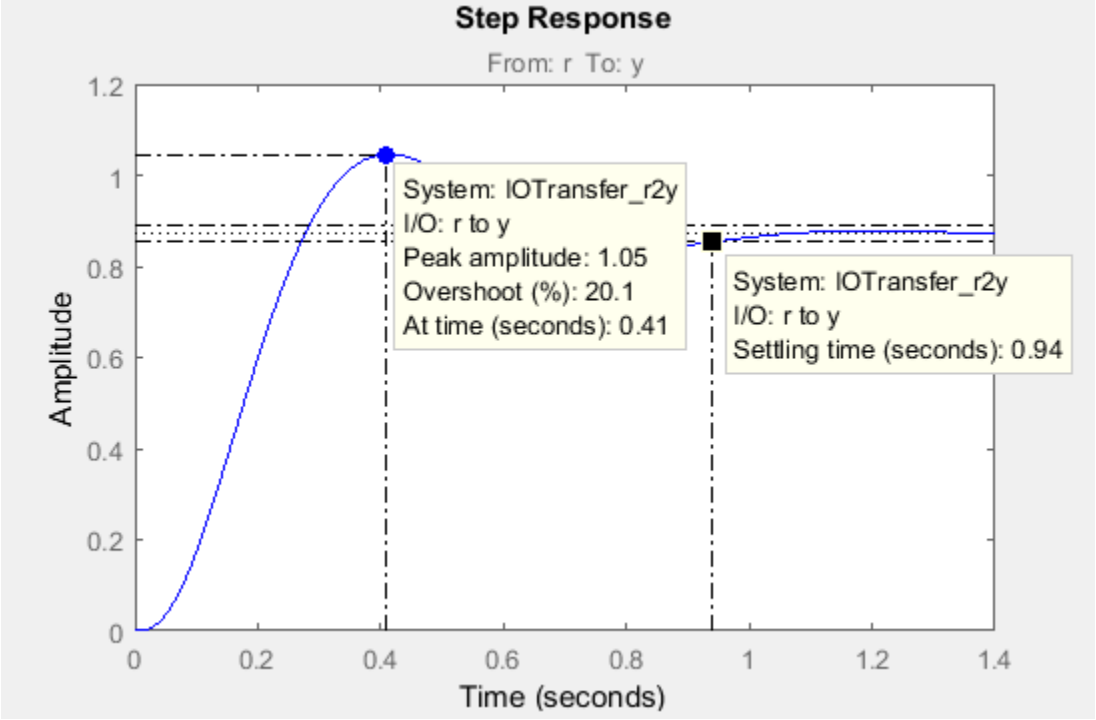



- Add Settling time = 0.95s and % OS < 20% in design Requirements in sisotool window.

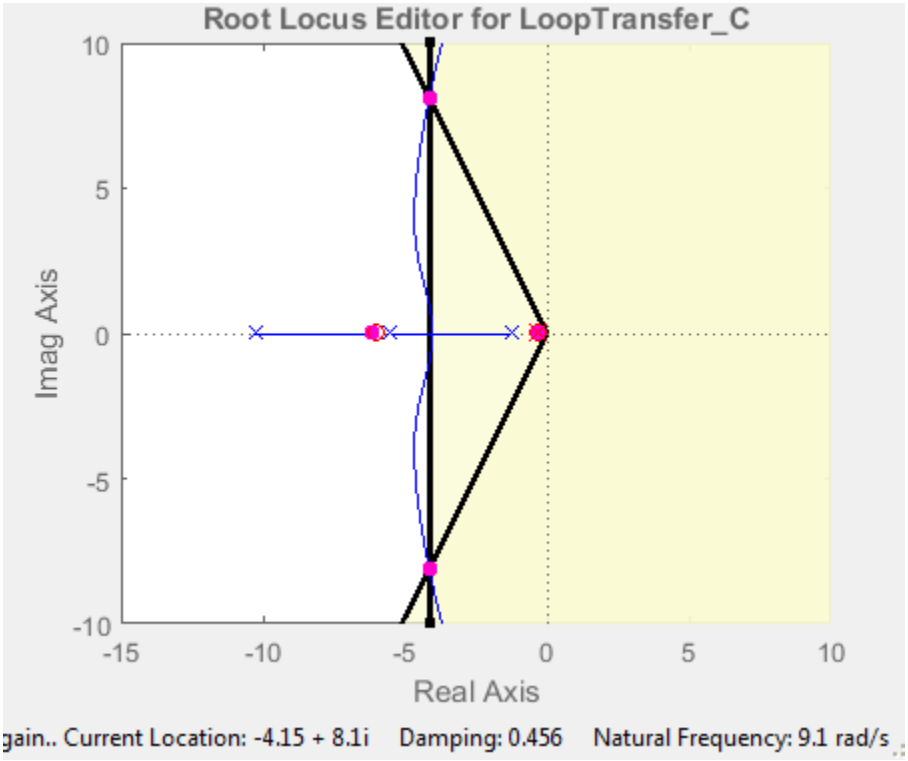



- Add Lead Compensator (suppose Zero $Z_C$ = -1 and $P_C$ = -37, **abs($P_C$)>abs($Z_C$)**) in Compensator Editor in sisotool window.






- Add Lag Compensator (suppose Zero $Z_C$ = -0.3 and $P_C$ = -0.295 **abs($P_C$)<abs($Z_C$))** in Compensator Editor in sisotool window.



Compensator

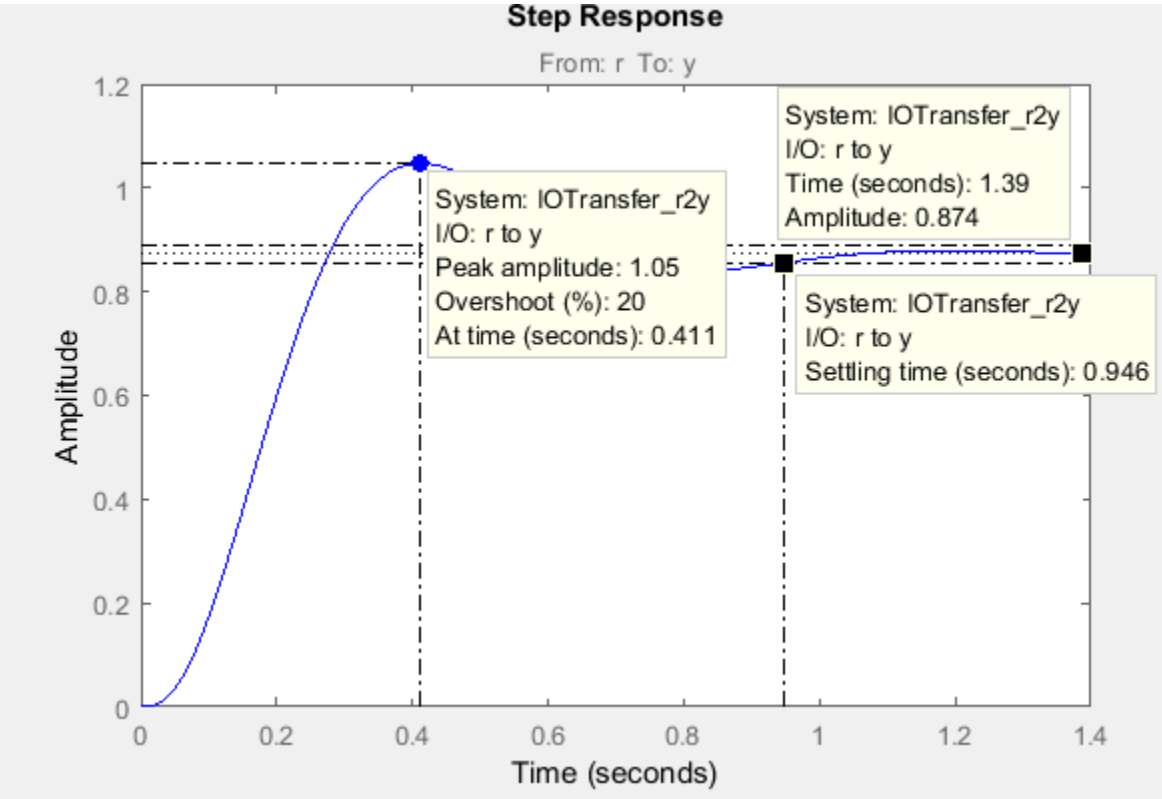
C = 481.31 x $\frac{(1 + 0.17s)(1 + 3.3s)}{(1 + 0.027s)(1 + 3.4s)}$

Pole/Zero | Parameter

Dynamics

| Type | Location | Damping | Frequency |
|---|---|---|---|
| Lead | -6, -37 | 1 | 6, 37 |
| Lag | -0.3, -0.295 | 1 | 0.3, 0.295 |

Edit Selected Dynamics




## Conclusion:

- Lead compensator can fast up the system response.
- Lag compensator reduce the steady state error

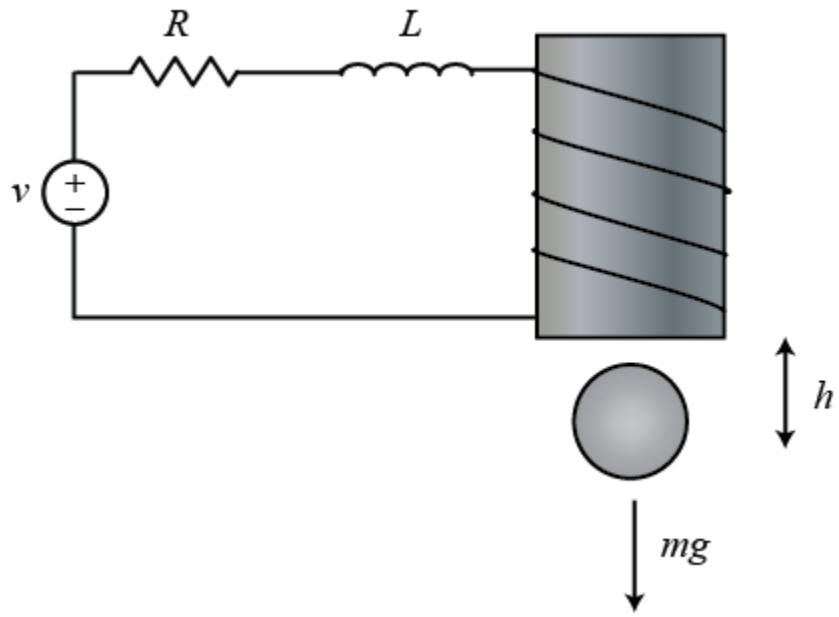
## Lab # 7

### Full state Feedback control design in MATLAB

### Objectives:

- Consider a magnetically suspended system, which input reference (r) = 0 (Regulator problem).
- Design specifications are
  iii. % OS < 5%
  iv. Settling time = $t_s$ < 0.5s.

### Modeling:




The equations for the system are given by:

$$m\frac{d^2h}{dt^2} = mg - \frac{Ki^2}{h}$$

$$V = L\frac{di}{dt} + iR$$

where $h$ is the vertical position of the ball, $i$ is the current through the electromagnet, $V$ is the applied voltage, $m$ is the mass of the ball, $g$ is the acceleration due to gravity, $L$ is the inductance, $R$ is the resistance, and $K$ is a coefficient that determines the magnetic force exerted on the ball. For simplicity, we will choose values $m$ = 0.05 kg, $K$ = 0.0001, $L$ = 0.01 H, $R$ = 1 Ohm, $g$ = 9.81 m/s^2. The system is at equilibrium (the ball is suspended in mid-air) whenever $h = Ki^2/mg$ (at which point $dh/dt$ = 0). We

linearize the equations about the point $h$ = 0.01 m (where the nominal current is about 7 Amps) and obtain the linear state-space equations:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + Bu$$

$$y = C\mathbf{x} + Du$$

where $\mathbf{x}$ is an n by 1 vector representing the system's state variables, $u$ is a scalar representing the input, and $y$ is a scalar representing the output. The matrices $A$ (n by n), $B$ (n by 1), and $C$ (1 by n) determine the relationships between the state variables and the input and output.

In our case:

$$x = \begin{bmatrix} \Delta h \\ \Delta \dot{h} \\ \Delta i \end{bmatrix}$$

is the set of state variables for the system (a 3x1 vector), $u$ is the deviation of the input voltage from its equilibrium value ($\Delta V$), and $y$ (the output) is the deviation of the height of the ball from its equilibrium position ($\Delta h$).

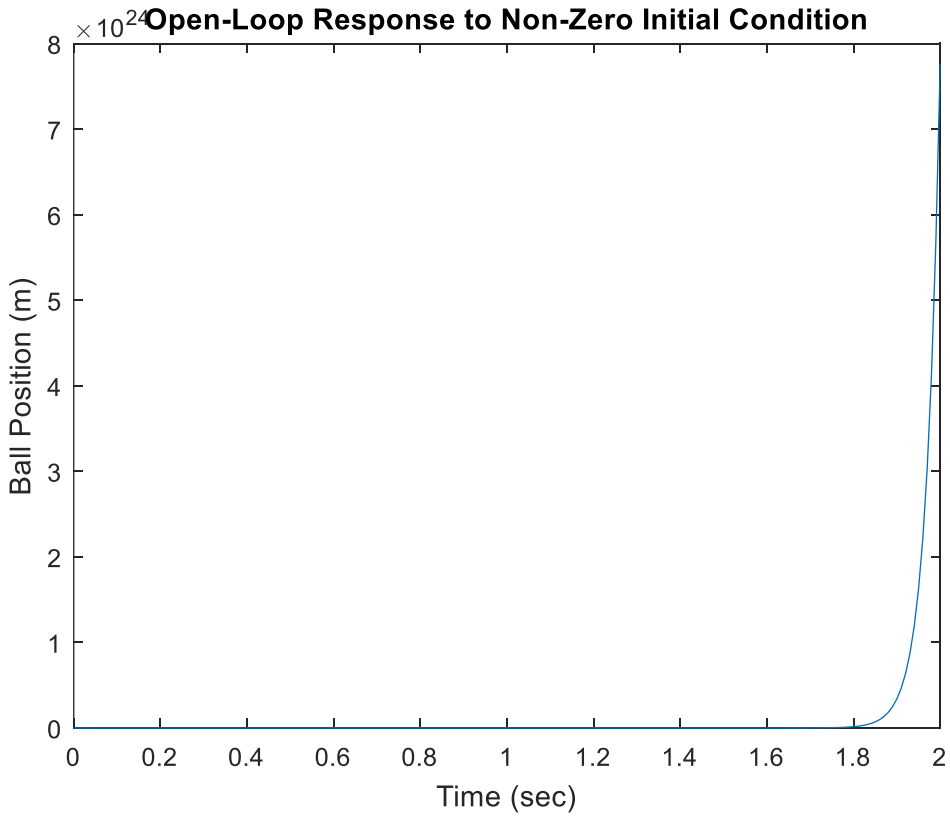
## Model representation in M-file:

```
1 -    clc;
2 -    clear close all
3 -    A=[0 1 0; 980 0 -28;0 0 -100];
4 -    B= [0;0;100];
5 -    C=[ 1 0 0];
6 -    D=0;
7 -    eig(A)
ans =

   31.3050
  -31.3050
 -100.0000
```

It can be seen that one of the poles is in the right-half plane (i.e. has positive real part), which means that the open-loop system is unstable.

To observe what happens to this unstable system when there is a non-zero initial condition, add the following lines to your m-file and run it again:

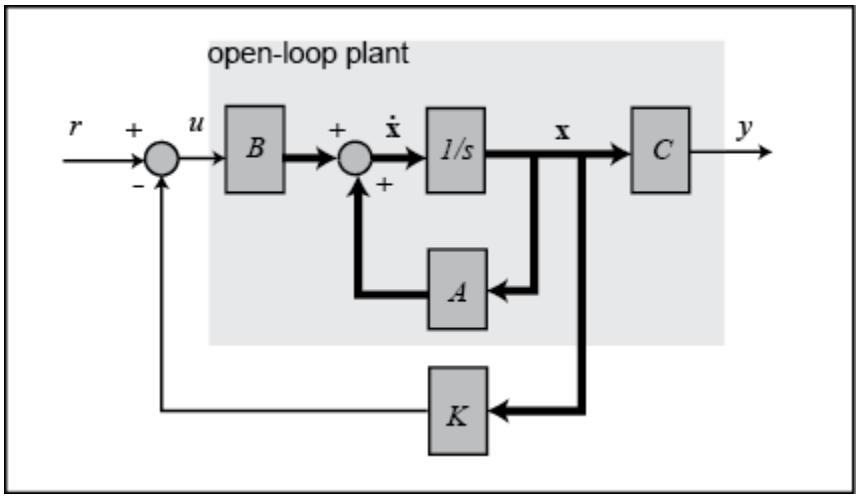
```
t = 0:0.01:2;
u = zeros(size(t));
x0 = [0.01 0 0];

sys = ss(A,B,C,0);

[y,t,x] = lsim(sys,u,t,x0);
plot(t,y)
title('Open-Loop Response to Non-Zero Initial Condition')
xlabel('Time (sec)')
ylabel('Ball Position (m)')
```




It looks like the distance between the ball and the electromagnet will go to infinity, but probably the ball hits the table or the floor first (and also probably goes out of the range where our linearization is valid).

## Control Design Using Pole Placement:

Let's build a controller for this system using a pole placement approach. The schematic of a full-state feedback system is shown below. By full-state, we mean that all state variables are known to the controller at all times. For this system, we would need a sensor measuring the ball's position, another measuring the ball's velocity, and a third measuring the current in the electromagnet.




Regulator Problem: Input reference r = 0;

$$u = -K\mathbf{x}$$

The state-space equations for the closed-loop feedback system are, therefore,

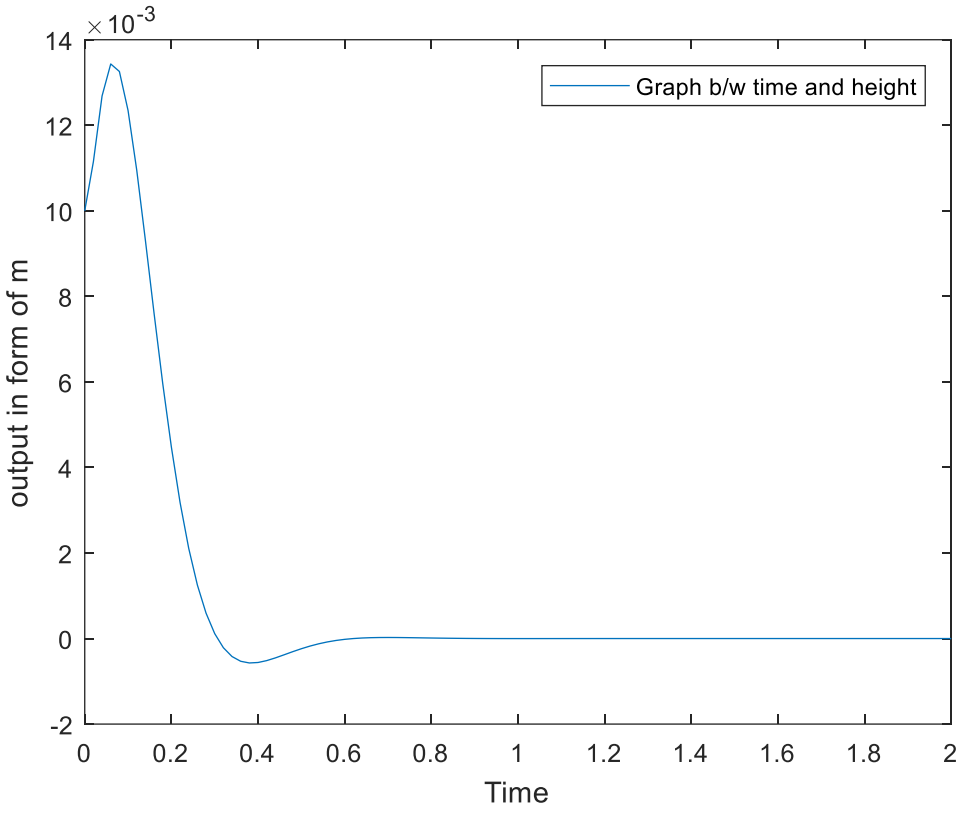
$$\dot{\mathbf{x}} = A\mathbf{x} + B(-K\mathbf{x}) = (A - BK)\mathbf{x}$$

$$y = C\mathbf{x}$$

The stability and time-domain performance of the closed-loop feedback system are determined primarily by the location of the eigenvalues of the matrix ($A - BK$), which are equal to the closed-loop poles. Since the matrices $A$ and $BK$ are both 3x3, there will be 3 poles for the system. By choosing an appropriate state-feedback gain matrix $K$, we can place these closed-loop poles anywhere we'd like (because the system is controllable). We can use the MATLAB function place to find the state-feedback gain, $K$, which will provide the desired closed-loop poles.

Before attempting this method, we have to decide where we want to place the closed-loop poles. Suppose the criteria for the controller were settling time < 0.5 sec and overshoot < 5%, then we might try to place the two dominant poles at -10 +/- 10i (at $\zeta$ = 0.7 or 45 degrees with $\sigma$ = 10 > 4.6*2). The third pole we might place at -50 to start (so that it is sufficiently fast that it won't have much effect on the response), and we can change it later depending on what closed-loop behavior results. Remove the lsim command from your m-file and everything after it, then add the following lines to your m-file:

```
8 -     t=0:0.02:2;
9 -     u=zeros(size(t));%I/P is 0 so, it is regulator problem
10 -    x0=[0.01 0 0];
11 -    p=[-10+10i -10-10i -50]; %postion where percent OS <0.05 and ts <5s
12 -    k=place(A,B,p);%give vale of k where we get desired poles location
13 -    sys=ss(A,B,C,D);
14 -    sys_cl=ss(A-B*k,B,C,D)
15 -    [y,t,x]=lsim(sys_cl,u,t,x0);
16 -    plot(t,y)
17 -    xlabel('Time');
18 -    ylabel('output in form of m');
19 -    legend('Graph b/w time and height');
```




**Note:** If you want to place two or more poles at the same position, place will not work. You can use a function called acker which achieves the same goal (but can be less numerically well-conditioned):

K = acker(A,B,p)

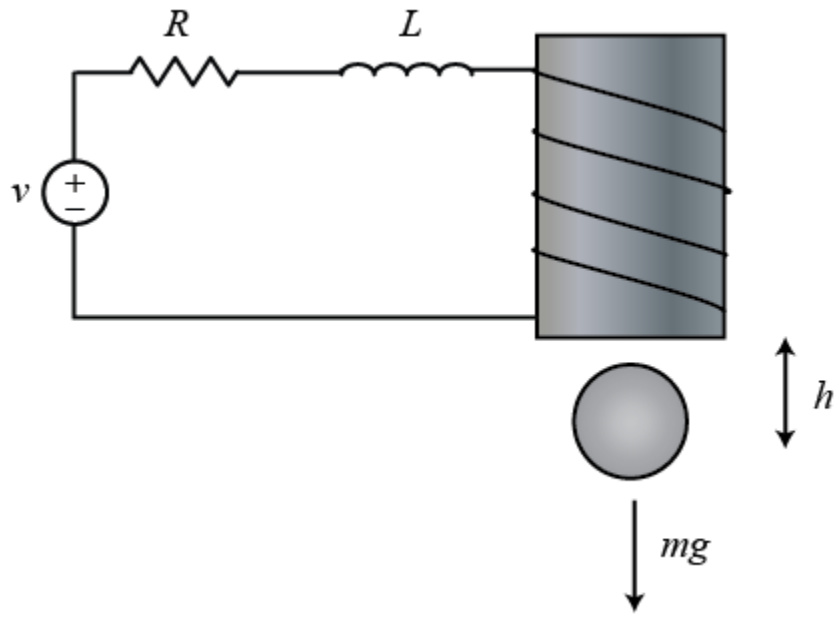
# Lab # 8

## Full state Feedback control design in MATLAB

### Objectives:

- Consider a magnetically suspended system, which input reference (r) ≠ 0 .
- Design specifications are
  - v. % OS < 5%
  - vi. Settling time = $t_s$ < 0.5s.

### Modeling:




The equations for the system are given by:

$$m\frac{d^2h}{dt^2} = mg - \frac{Ki^2}{h}$$

$$V = L\frac{di}{dt} + iR$$

where $h$ is the vertical position of the ball, $i$ is the current through the electromagnet, $V$ is the applied voltage, $m$ is the mass of the ball, $g$ is the acceleration due to gravity, $L$ is the inductance, $R$ is the resistance, and $K$ is a coefficient that determines the magnetic force exerted on the ball. For simplicity, we will choose values $m$ = 0.05 kg, $K$ = 0.0001, $L$ = 0.01 H, $R$ = 1 Ohm, $g$ = 9.81 m/s^2. The system is at equilibrium (the ball is suspended in mid-air) whenever $h = Ki^2/mg$ (at which point $dh/dt$ = 0). We

linearize the equations about the point $h$ = 0.01 m (where the nominal current is about 7 Amps) and obtain the linear state-space equations:

$$\frac{d\mathbf{x}}{dt} = A\mathbf{x} + Bu$$

$$y = C\mathbf{x} + Du$$

where $\mathbf{x}$ is an n by 1 vector representing the system's state variables, $u$ is a scalar representing the input, and $y$ is a scalar representing the output. The matrices $A$ (n by n), $B$ (n by 1), and $C$ (1 by n) determine the relationships between the state variables and the input and output.

In our case:

$$x = \begin{bmatrix} \Delta h \\ \Delta \dot{h} \\ \Delta i \end{bmatrix}$$

is the set of state variables for the system (a 3x1 vector), $u$ is the deviation of the input voltage from its equilibrium value ($\Delta V$), and $y$ (the output) is the deviation of the height of the ball from its equilibrium position ($\Delta h$).

## Model representation in M-file:

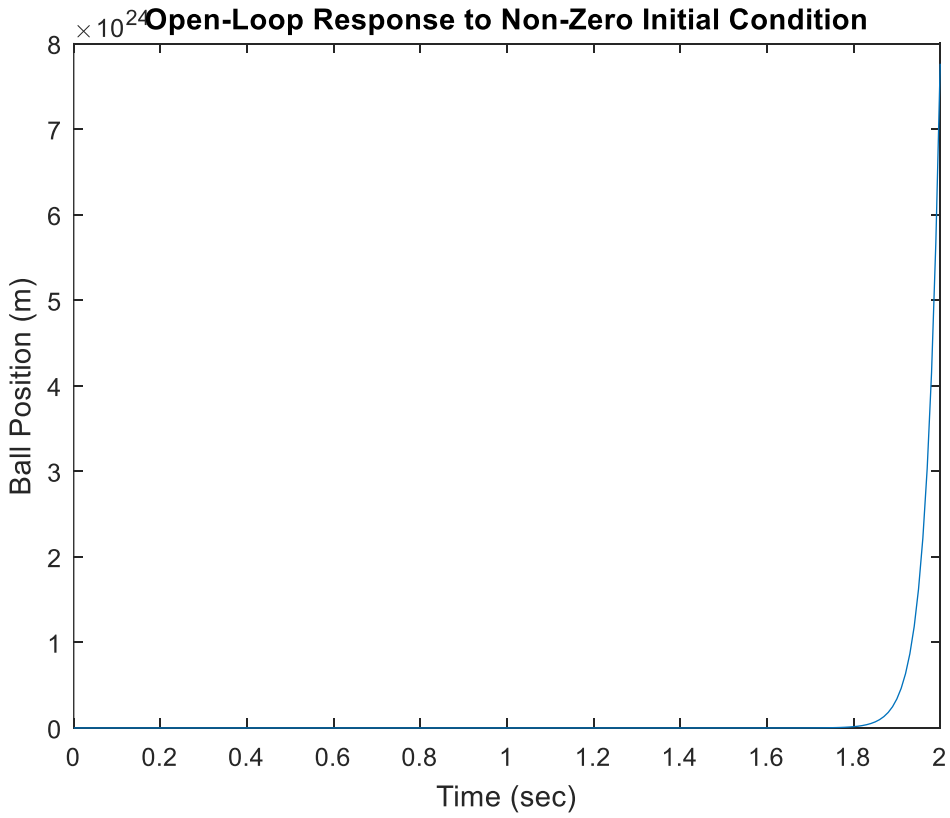
```
clc;
clear close all
A=[0 1 0; 980 0 -28;0 0 -100];
B= [0;0;100];
C=[ 1 0 0];
D=0;
eig(A)
```

```
ans =

   31.3050
  -31.3050
 -100.0000
```

It can be seen that one of the poles is in the right-half plane (i.e. has positive real part), which means that the open-loop system is unstable.

To observe what happens to this unstable system when there is a non-zero initial condition, add the following lines to your m-file and run it again:

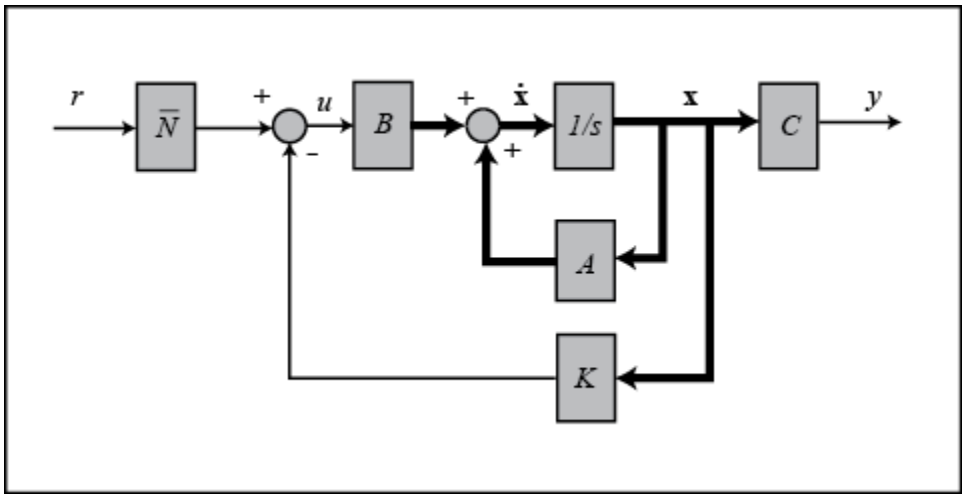
```
8 -     t = 0:0.01:2;
9 -     u = ones(size(t));
10 -    x0 = [0.01 0 0];
11
12 -    sys = ss(A,B,C,0);
13
14 -    [y,t,x] = lsim(sys,u,t,x0);
15 -    plot(t,y)
16 -    title('Open-Loop Response to Non-Zero Initial Condition')
17 -    xlabel('Time (sec)')
18 -    ylabel('Ball Position (m)')
```




It looks like the distance between the ball and the electromagnet will go to infinity, but probably the ball hits the table or the floor first (and also probably goes out of the range where our linearization is valid).

## Control Design Using Pole Placement:

Let's build a controller for this system using a pole placement approach. The schematic of a full-state feedback system is shown below. By full-state, we mean that all state variables are known to the controller at all times. For this system, we would need a sensor measuring the ball's position, another measuring the ball's velocity, and a third measuring the current in the electromagnet.




Regulator Problem: Input reference r ≠ 0;

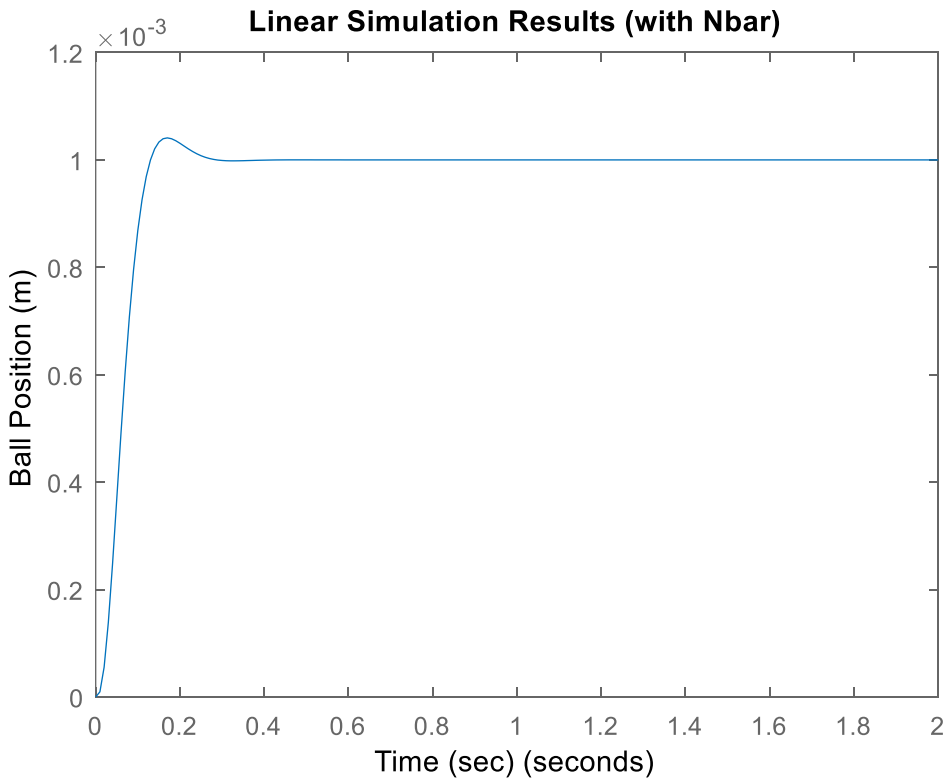
$$u = -K\mathbf{x} \quad + \bar{N}\mathrm{r}$$

The state-space equations for the closed-loop feedback system are, therefore,

The stability and time-domain performance of the closed-loop feedback system are determined primarily by the location of the eigenvalues of the matrix (), which are equal to the closed-loop poles. Since the matrices $A$ and $BK$ are both 3x3, there will be 3 poles for the system. By choosing an appropriate state-feedback gain matrix $K$, we can place these closed-loop poles anywhere we'd like (because the system is controllable). We can use the MATLAB function place to find the state-feedback gain, $K$, which will provide the desired closed-loop poles.

Before attempting this method, we have to decide where we want to place the closed-loop poles. Suppose the criteria for the controller were settling time < 0.5 sec and overshoot < 5%, then we might try to place the two dominant poles at -10 +/- 10i (at $\zeta$ = 0.7 or 45 degrees with $\sigma$ = 10 > 4.6*2). The third pole we might place at -50 to start (so that it is sufficiently fast that it won't have much effect on the response), and we can change it later depending on what closed-loop behavior results. Remove the lsim command from your m-file and everything after it, then add the following lines to your m-file:

```
 8 -    t = 0:0.01:2;
 9 -    u = 0.001*ones(size(t));%I/P is 0 so, it is regulator problem
10 -    x0=[0.01 0 0];
11 -    p= 2*[-10+10i -10-10i -50]; %postion where percent OS <0.05 and ts <5s
12 -    k=place(A,B,p);%give vale of k where we get desired poles location
13 -    sys=ss(A,B,C,D);
14 -    Nbar = rscale(sys,k)
15 -    sys_cl = ss(A-B*k,B,C,0);
16 -    lsim(sys_cl,Nbar*u,t)
17 -    title('Linear Simulation Results (with Nbar)')
18 -    xlabel('Time (sec)')
19 -    ylabel('Ball Position (m)')
20 -    axis([0 2 0 1.2*10^-3])
```




and now a step can be tracked reasonably well. Note, our calculation of the scaling factor requires good knowledge of the system. If our model is in error, then we will scale the input an incorrect amount. An alternative, similar to what was introduced with PID control, is to add a state variable for the integral of the output error. This has the effect of adding an integral term to our controller which is known to reduce steady-state error.

## Lab # 9

### Observer design in MATLAB

### Objectives:

- Consider inverted pendulum on cart.
  1. Find observer gain matrix L to place all the closed-loop system poles at desired locations.

### Modeling:

### Model representation in M-file:

```
clc;
clear close all
A=[0 1 0 0; 0 0 -1 0;0 0 0 1;0 0 100 0]
B= [0;0.1237;0;-1.2621]
C=[ 1 0 0 0]
D=0;
```

```
A =

     0     1     0     0
     0     0    -1     0
     0     0     0     1
     0     0   100     0


B =

         0
    0.1237
         0
   -1.2621


C =

     1     0     0     0
```

Check the observability of this matrix.

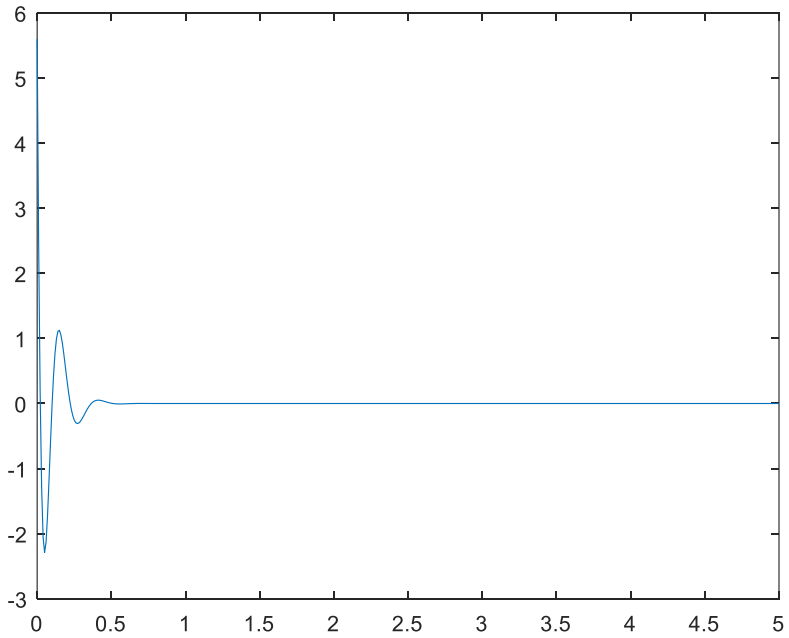
```
>> obsv(A,C)

ans =
```

```
     1     0     0     0
     0     1     0     0
     0     0    -1     0
     0     0     0    -1
```

det $P_0 = 1 \neq 0$; hence, the system is completely observable.

## Observer Design:

```
p=[-16+21.3i -16-21.3i -16+21.3i -16-21.3i];
x0=[5.6 0 0 0];
L=acker(A',C',p)%give vale of L (observer design) where we get desired poles location
L=L';
t=0:0.01:5;
u=zeros(size(t));%I/P is 0 so, it is regulator problem
sys_cl=ss(A-(L*C),B,C,D);
[y,t,x]=lsim(sys_cl,u,t,x0);
%plot(t,x(:,4))
plot(t,y)
```




```
L =

   1.0e+05 *

    0.0006
    0.0254
   -0.5182
   -7.5800
```

## Conclusion:

The observer needs to provide an estimate of the states that cannot be directly observed. The goal is to achieve an accurate estimate as fast as possible without resulting in too large a gain matrix L. How large is too large depends on the problem under consideration. In particular, if there are significant levels of measurement noise (this is sensor dependent), then the magnitude of the observer matrix should be kept correspondingly low to avoid amplifying the measurement noise. The trade-off between the time required to obtain accurate observer performance and the amount of noise amplification is a primary design issue.

# Lab # 10

## Optimal controller (LQR) design in MATLAB

### Objectives:

- Consider an Inverted Pendulum on cart.
- Design specifications are
  1. $e_{ss}$ < 2%
  2. Settling time = $t_s$ < 0.5s.

### Modeling:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{I(M+m)+Ml^2} & \frac{m^2gl^2}{I(M+m)+Ml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{I(M+m)+Ml^2} & \frac{mgl(M+m)}{I(M+m)+Ml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{I(M+m)+Ml^2} \\ 0 \\ \frac{ml}{I(M+m)+Ml^2} \end{bmatrix} u$$

$$\mathbf{y} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

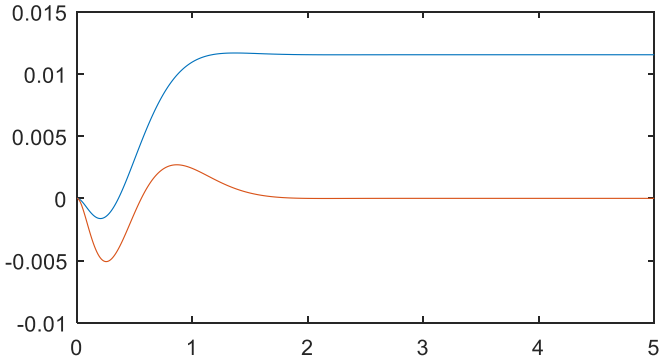
where:

| | | |
|---|---|---|
| (M) | mass of the cart | 0.5 kg |
| (m) | mass of the pendulum | 0.2 kg |
| (b) | coefficient of friction for cart | 0.1 N/m/sec |
| (l) | length to pendulum center of mass | 0.3 m |
| (I) | mass moment of inertia of the pendulum | 0.006 kg.m^2 |
| (F) | force applied to the cart | |
| (x) | cart position coordinate | |
| (theta) | pendulum angle from vertical (down) | |

```
M = 0.5;m = 0.2;b = 0.1;I = 0.006;g = 9.8;l = 0.3;
p = I*(M+m)+M*m*l^2; %denominator for the A and B matrices

A = [0 1 0 0;0 -(I+m*l^2)*b/p (m^2*g*l^2)/p 0;0 0 0 1;0 -(m*l*b)/p m*g*l*(M+m)/p 0]
B = [0;(I+m*l^2)/p;0;m*l/p]
```

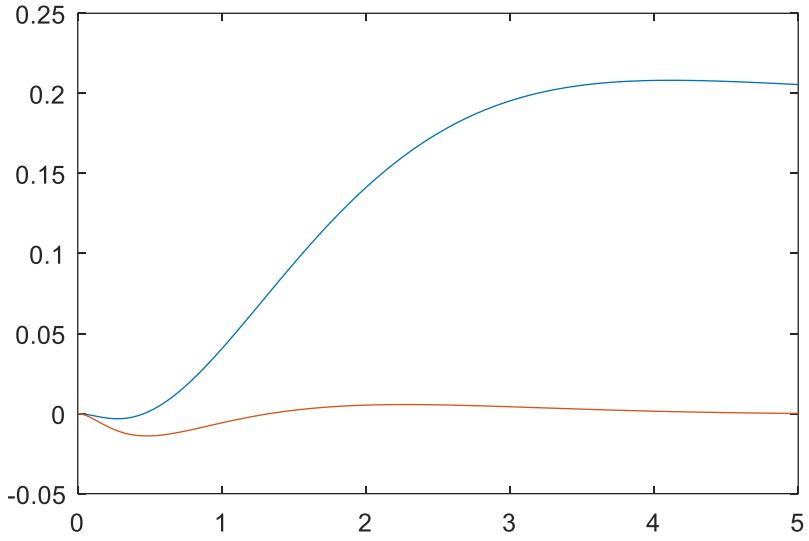
```
A =

         0    1.0000         0         0
         0   -0.1818    2.6727         0
         0         0         0    1.0000
         0   -0.4545   31.1818         0


B =

         0
    1.8182
         0
    4.5455
```

```
 5 -    C=[ 1 0 0 0;0 0 1 0];
 6 -    D=[0;0];
 7 -    sys=ss(A,B,C,D);
 8 -    poles=eig(sys);%check the controlabilty of system
 9 -    Pc=ctrb(sys);
10 -    Controllablity=rank(Pc)
11 -    det(Pc);
12 -    Q=C'*C;%Simplest Selection of Q
13 -    Q(1,1)=300;% Fast the Trasient response
14 -    R=1;
15 -    K=lqr(A,B,Q,R);
16 -    sys_cl=ss(A-(B*K),B,C,D);
17 -    t=0:0.01:5;
18 -    r=-1*0.2*ones(size(t));%I/P isn't 0 so, it isn't regulator problem u=-Nr
19 -    [y,t,x]=lsim(sys_cl,r,t);
20 -    plot(t,y)
```




As we increases the value of Q(1,1) or Q(3,3), we get fast transient response. But peak value reduced.

Q(1,1)=1



# Lab # 11

## Objectives:

- Design the system for optimal (DeadBeat) response.
  1. $G(s) = \frac{K}{s(s+1)} = \frac{K}{s^2+s}$ Required $T_s$ = 2s. (Lab work)
  2. $G(s) = \frac{K}{s(s+1)} = \frac{K}{s^2+s}$ Required $T_s$ = 0.25s. (Task)

## Lab Work:

$$\boldsymbol{G(s) = \frac{K}{s^2+s}\ \text{Required}\ T_s = 2s}$$

## Model representation and step response in M-file:

```
clear close all;
clc;
num=[1];
den=[1 1 0];
sys=tf(num,den)
stepinfo(sys)
```

## Command Window:

```
sys =

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


ans =

  struct with fields:

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

## Compensator Design:

$\omega_n T_s = 4.04$ rad

$\omega_n = 2.02$ rad/s

- Erase 6 line's command in M-file

## Model representation and step response in M-file:

```
a=1.9;b=2.2;wn=2.02;%because of 3rd order equation
p=(a*wn)-1
K=(b*wn*wn)-p
z=(wn^3)/K
T_num=[K K*z];
T_den=[1 1+p p+K K*z];
sys_cl=tf(T_num,T_den)
stepinfo(sys_cl)
```

## Command Window:

```
p =

    2.8380


K =

    6.1389


z =

    1.3427
```

```
sys_cl =

           6.139 s + 8.242
  ---------------------------------
  s^3 + 3.838 s^2 + 8.977 s + 8.242

Continuous-time transfer function.
```

```
ans =

  struct with fields:

        RiseTime: 0.6451
    SettlingTime: 3.4869
     SettlingMin: 0.9284
     SettlingMax: 1.2081
       Overshoot: 20.8078
      Undershoot: 0
            Peak: 1.2081
        PeakTime: 1.5027
```

- Still settling time is greater than 2s because of addition zero add in compensator.
- Remove the addition zero effect by adding **Pre-filter**.

## Model representation and step response in M-file:

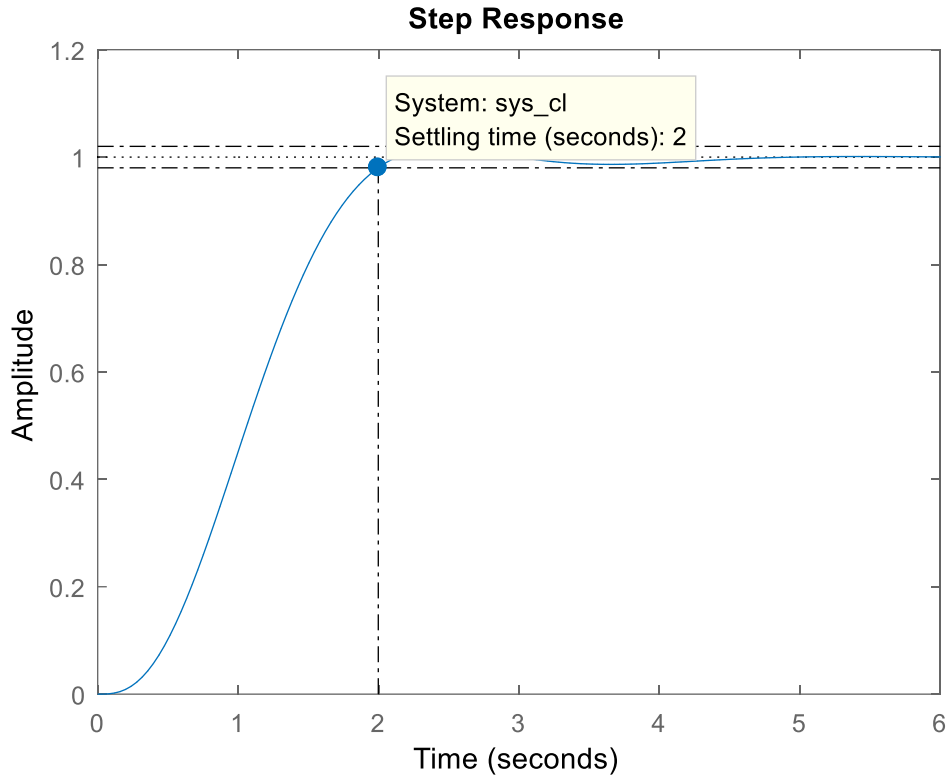
```
13 -    G_num=[z];
14 -    G_den=[1 z];
15 -    Gp=tf(G_num,G_den)
16 -    sys_cl=Gp*sys_cl;
17 -    stepinfo(sys_cl)
18 -    step(sys_cl)
```

## Command Window:

```
Gp =

    1.343
  ---------
  s + 1.343

Continuous-time transfer function.

ans =

  struct with fields:

        RiseTime: 1.2185
    SettlingTime: 1.9984
     SettlingMin: 0.9098
     SettlingMax: 1.0165
       Overshoot: 1.6510
      Undershoot: 0
            Peak: 1.0165
        PeakTime: 2.4518
```

- Now desired settling time ($T_s$ = 2s) achieved by adding pre-filter.

## Result:



**Task:**

$$G(s) = \frac{K}{s(s+1)}$$ **Required $T_s$ = 0.25s**

Model representation and step response in M-file:

```
clear close all;
clc;
num=[1];
den=[1 1 0];
sys=tf(num,den)
a=1.9;b=2.2;wn=16.16;%because of 3rd order equation
p=(a*wn)-1
K=(b*wn*wn)-p
z=(wn^3)/K
T_num=[K K*z];
T_den=[1 1+p p+K K*z];
sys_cl=tf(T_num,T_den)
G_num=[z];
G_den=[1 z];
Gp=tf(G_num,G_den)
sys_cl=Gp*sys_cl;
stepinfo(sys_cl)
step(sys_cl)
```

Command Window:

```
sys =

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

Continuous-time transfer function.


p =

   29.7040


K =

  544.8163
```

```
z =

    7.7459

sys_cl =

          544.8 s + 4220
  -------------------------------
  s^3 + 30.7 s^2 + 574.5 s + 4220

Continuous-time transfer function.


Gp =

    7.746
  ---------
  s + 7.746

Continuous-time transfer function.

ans =

  struct with fields:

        RiseTime: 0.1523
    SettlingTime: 0.2498
     SettlingMin: 0.9098
     SettlingMax: 1.0165
       Overshoot: 1.6510
      Undershoot: 0
            Peak: 1.0165
        PeakTime: 0.3065
```

## Result:

By adding pre-filter we can achieve desired settling time because pre-filter is used to eliminate the effect of addition zero as shown in graph.