A308 Control Theory
Course Project

Soham S. Phanse, 19D170030[1]

Under guidance of

Prof. Arnab Maity[1]

[1]Department of Aerospace Engineering,
Indian Institute of Technology Bombay

Prelude

In the PDF which I received from the TA the settling time was not mentioned hence I asked the TA on MS Teams and he told me settling time was 1.5 seconds. Here is the proof:

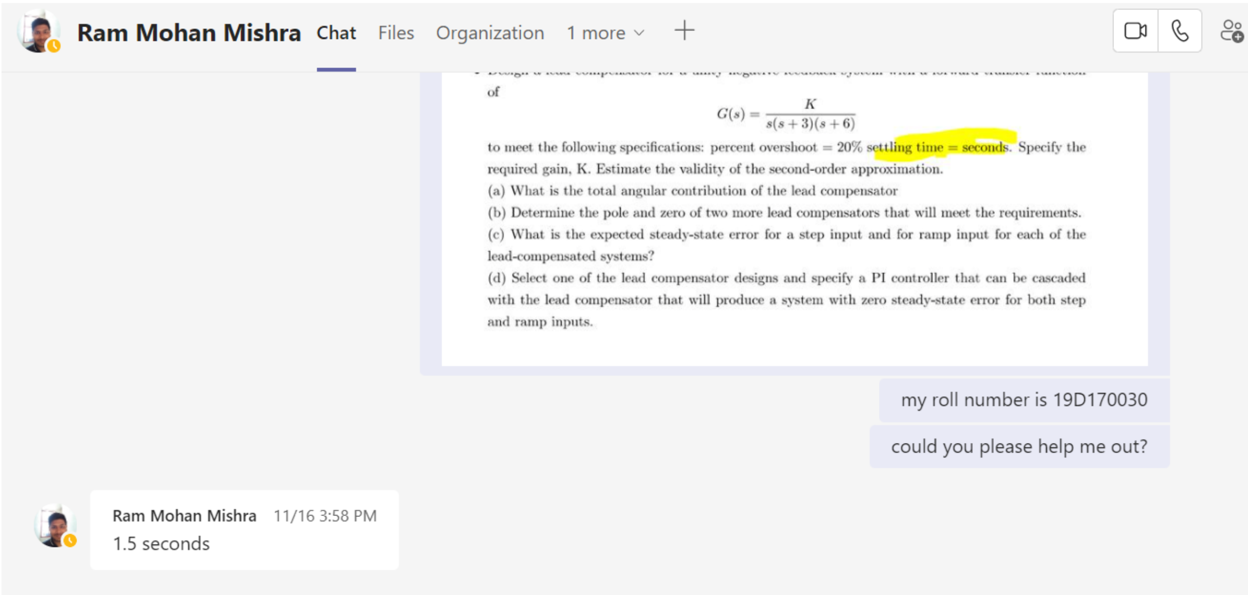


Table of Contents

The Question

Open Loop Transfer Function (negative unity feedback)

$$G(s) = \frac{K}{s(s+3)(s+6)}$$

Main Question

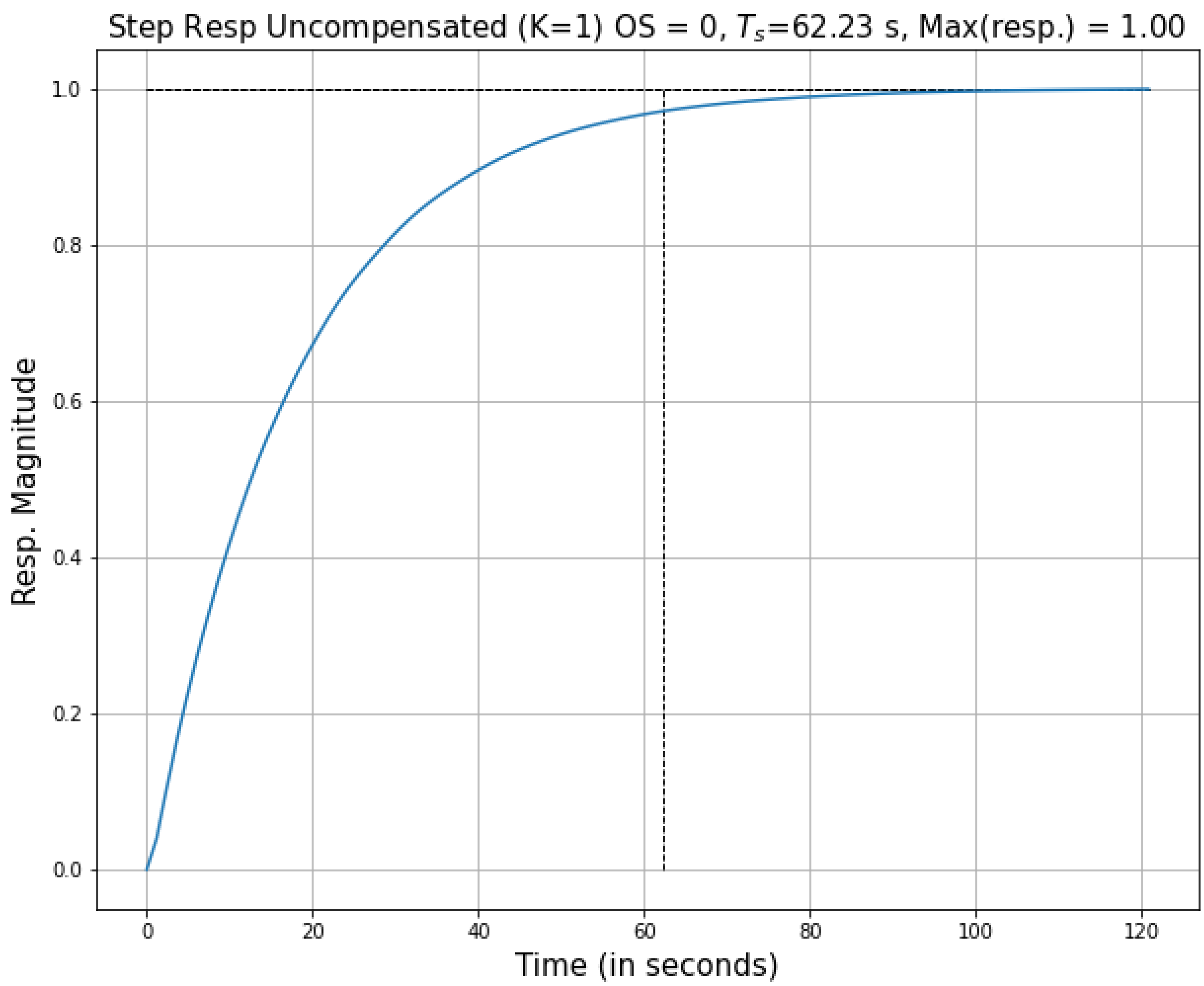
- Design a lead compensator

Requirements

- %OS = 20%
- Settling time = 1.5 seconds

Solution

Step Response of Uncompensated System (K = 1)



We can see that the overshoot is very low and according to the requirements but the settling time is large.

Open Loop Poles

s = 0, -3, -6

Converting Requirements into Closed Loop Poles

$$M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} = 0.2 \longrightarrow \zeta = 0.456$$

$$T_s \approx \frac{4}{\zeta\omega_n} = 1.5 \longrightarrow \omega_n = 5.8479$$

$$w_d = w_n\sqrt{1-\zeta^2} = 5.8479 \times 0.8899 = 5.2045$$

$$p_{1,2} = -\zeta\omega_n \pm i w_d = -2.67 \pm 5.2045i$$

$$\zeta = cos\phi \longrightarrow \phi = cos^{-1}\zeta$$

$$\phi = \pm 62.8707^\circ$$

We also know,

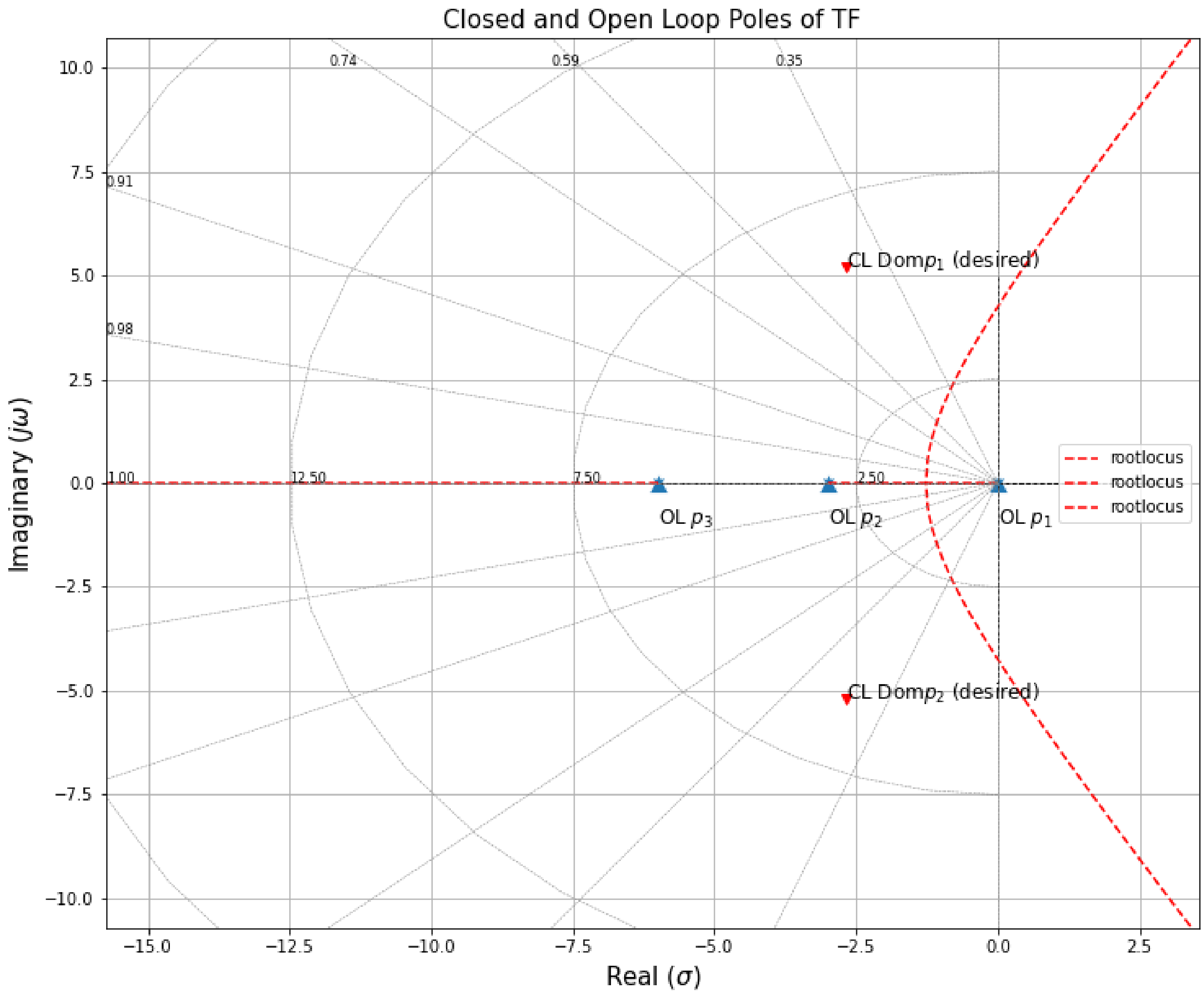
$$\text{Phase margin} = \tan^{-1}\frac{2\zeta}{\sqrt{-2\zeta^2 + \sqrt{1+4\zeta^4}}}$$

$$\zeta = 0.456 \implies \text{PM} = 48.1^\circ$$

Let the poles be $p_{1,2}$. The corresponding characteristic equation is as follows,

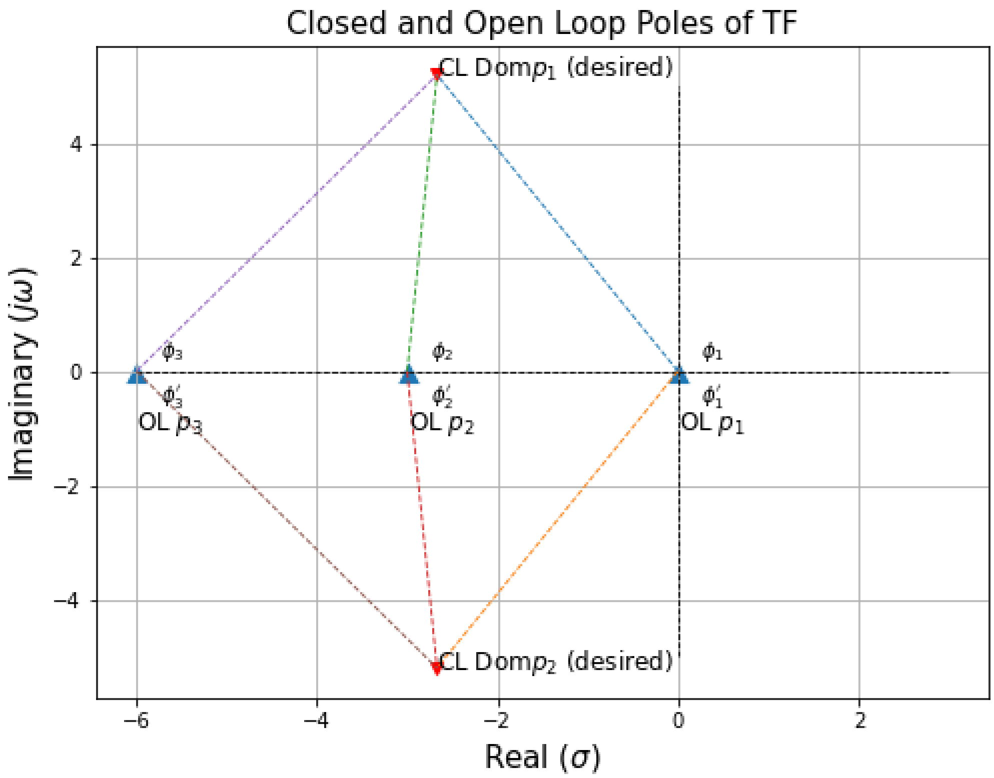
$$(s-p_1)(s-p_2) = s^2 + (-p_1 - p_2)s + p_1 p_2 =\!\!\longrightarrow s^2 + 2\zeta\omega_n s + \omega_n^2$$

OL Root Locus and Desired Pole Locations



Desired Root Locus

Angle Criterion Calculations



From preliminary trigonometric calculations, we conclude that,

$$\phi_1 = tan^{-1}\left(\frac{5.2045}{2.677}\right) = 180 - 62.7940 = 117.1586^{\circ}$$

$$\phi_2 = tan^{-1}\left(\frac{5.2045}{\frac{1}{3}}\right) = 86.5^{\circ}$$

$$\phi_3 = tan^{-1}\left(\frac{5.2045}{\frac{10}{3}}\right) = 57.3616^{\circ}$$

$$\phi_1 + \phi_2 + \phi_3 = 117.1586 + 86.5 + 57.3616 = 261.0202^{\circ}$$

$$261.0202 + \theta_p - \theta_z = 180^{\circ} \longrightarrow \theta_p - \theta_z = -81.0202^{\circ}$$

When a point lies on the root locus the sum of the angles made by each of the poles and zeros is 180^0. Hence with a lead compensator we add a pole and zero to compensate for the summation. Hence we have to add a pole of zero such that the above condition (last line) is satisfied.

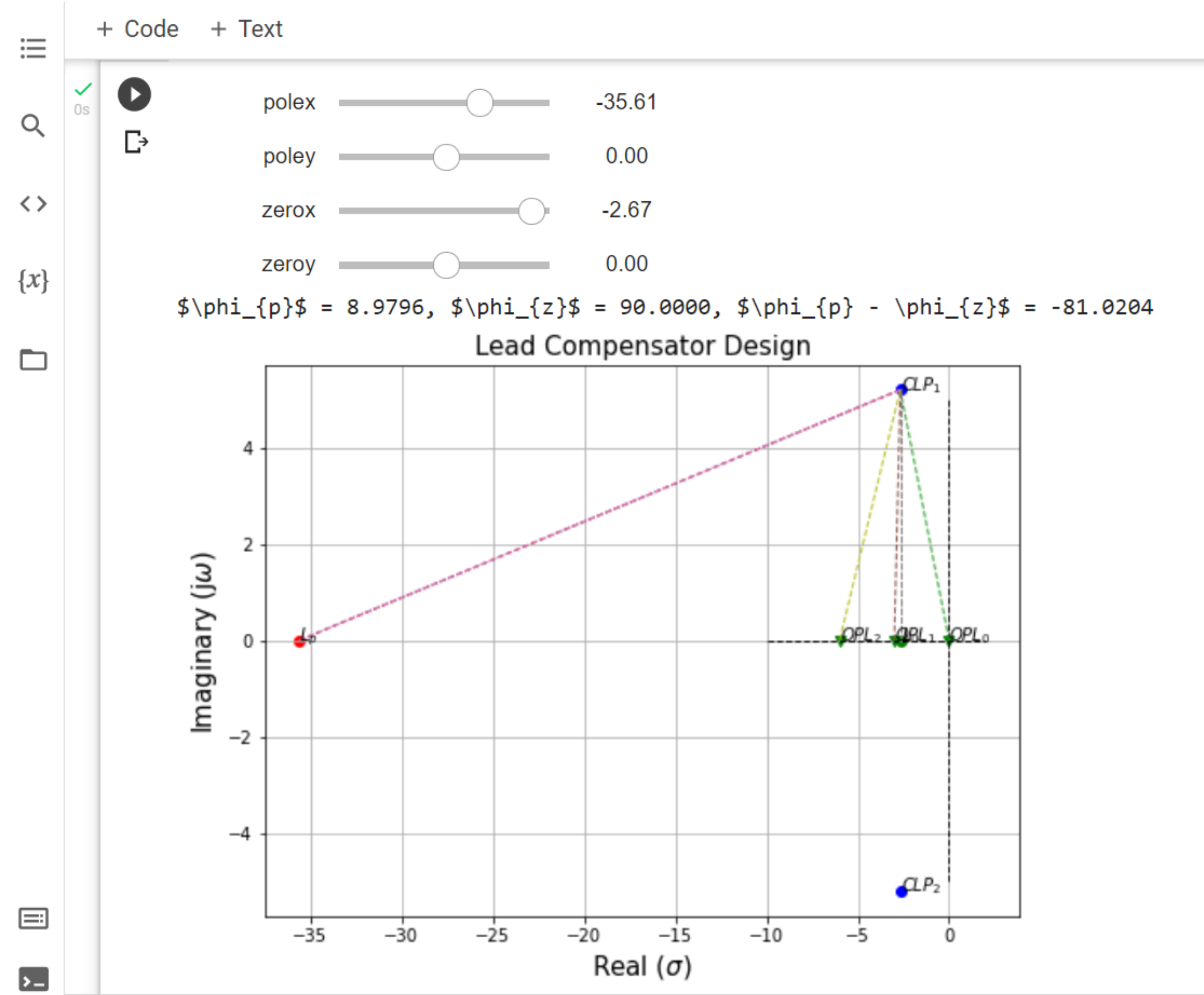
Now, the procedure to choose the pole and zero which will satisfy this criterion is iterative and not straightforward. Hence we automate this with some code.

Automation Codes (Python)

```
def euclidean_distance(p1, p2):
  return np.linalg.norm(np.array(p1)-np.array(p2))

def automate(polex, poley, zerox, zeroy, clpx=-2.67, clpy=5.2045):

  # plotting
  plt.figure(figsize=(7, 5))
  plt.title('Lead Compensator Design')
  plt.xlabel('Real ($\sigma$)')
  plt.ylabel('Imaginary (j$\omega$)')

  plt.scatter(zerox, zeroy, marker='o',c='g')
  plt.annotate(xy=(zerox, zeroy), s='$L_0$')

  plt.scatter(polex, poley, marker='o',c='r')
  plt.annotate(xy=(polex, poley), s='$L_p$')

  plt.scatter([clpx, clpx], [clpy, -clpy], marker='o', c='b')
  plt.annotate(xy=(clpx, clpy), s='$CLP_1$')
  plt.annotate(xy=(clpx, -1*clpy), s='$CLP_2$')

  plt.scatter([0, -3, -6], [0, 0, 0], marker='v', c='g')

  plt.hlines(0, -10.0, 2, linestyle='--', linewidth=1.0)
  plt.vlines(0, -5.0, 5.0, linestyle='--', linewidth=1.0)

  counter = 0
  for each in [0, -3, -6]:
    plt.plot([polex, clpx], [poley, clpy], linestyle='--', linewidth=1.0)
    plt.plot([zerox, clpx], [zeroy, clpy], linestyle='--', linewidth=1.0)
    plt.plot([each, clpx], [0, clpy], linestyle='--', linewidth=1.0)
    plt.annotate(xy=(each, 0), s='$OPL_{%.0f}$'%counter)
    counter += 1
```

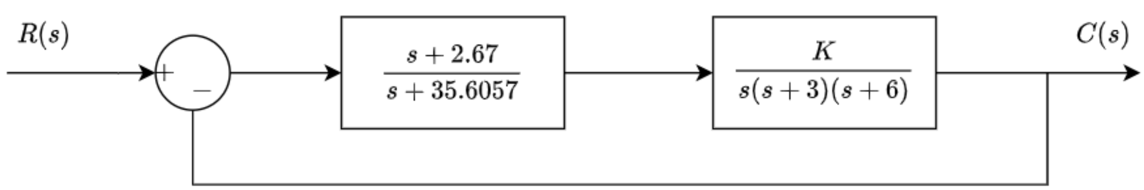
```
  plt.grid(1)

  # calculations
  pole, zero, clp = np.array([polex, poley]), np.array([zerox, zeroy]),
np.array([clpx, clpy])
  lp, lz = euclidean_distance(pole, clp), euclidean_distance(zero, clp)
  xp, xz = abs((pole-clp)[0]), abs((zero-clp)[0])
  tp, tz = np.arccos(xp/lp), np.arccos(xz/lz)
  tpr, tzr = tp, tz
  if pole[0]>clp[0]:
    tpr = 180 - tp
  if zero[0]>clp[0]:
    tzr = 180 - tz
  print('$\phi_p$ = %.4f, $\phi_z$ = %.4f, $\phi_p - \phi_z$ =
%.4f'%(tpr*180/np.pi, tzr*180/np.pi, (tpr-tzr)*180/np.pi))
  return (tpr*180/np.pi, tzr*180/np.pi)

from ipywidgets import interactive, interact, interact_manual
interact(automate, polex = (-50.0, -3.0), poley = (-5.0, 5.0), zerox =
(-10.0, -2.0, 0.0001), zeroy = (-5.0, 5.0));
```

Automation Results



If we place a pole at s = -35.6057, then we have Φ_p = 8.9798^0, hence from the above equation we have, Φ_z = 90^0. Hence the zero will be placed at s = -2.67

Hence the Resulting lead lag compensation and the overall system is as follows:

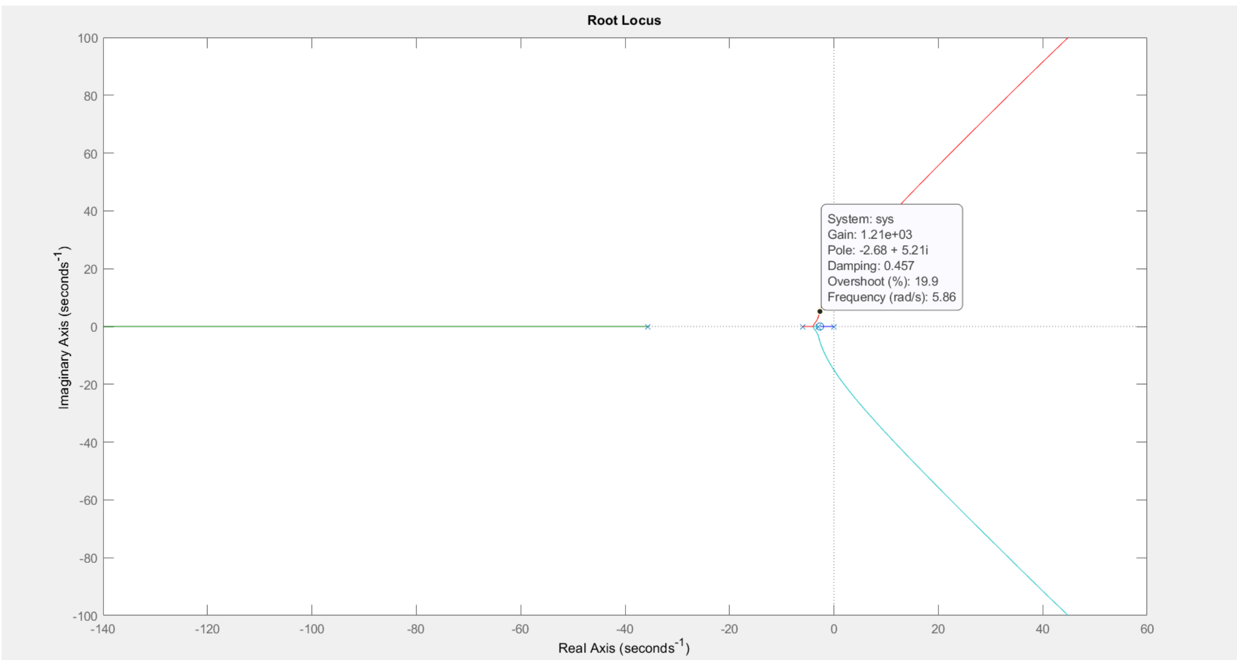


Root Locus of New System

Codes (MATLAB)

```
>> p = 2.67;
>> z = 35.6057;
>> sys = tf([1, z], [1, 9+p, 9*(p+2), 18*p, 0]);
>> rlocus(sys)
```

Plots



From the above figure of new root locus, we conclude that the gain of the system will be 1.21e+03 when the pole and zero are at -35.6057 and -2.67 respectively. And we get %OS ~ 19% and settling time $T_s = 4/\zeta\omega_n$ ~ 1.3 seconds, which is very close to the requirements.

$$G_{overall}(s) = \frac{1210(s+2.67)}{s^4 + 45.6057s^3 + 338.45s^2 + 1850.90s + 3226.67}$$
$$p, z, K = -35.6057, -2.67, 1210$$

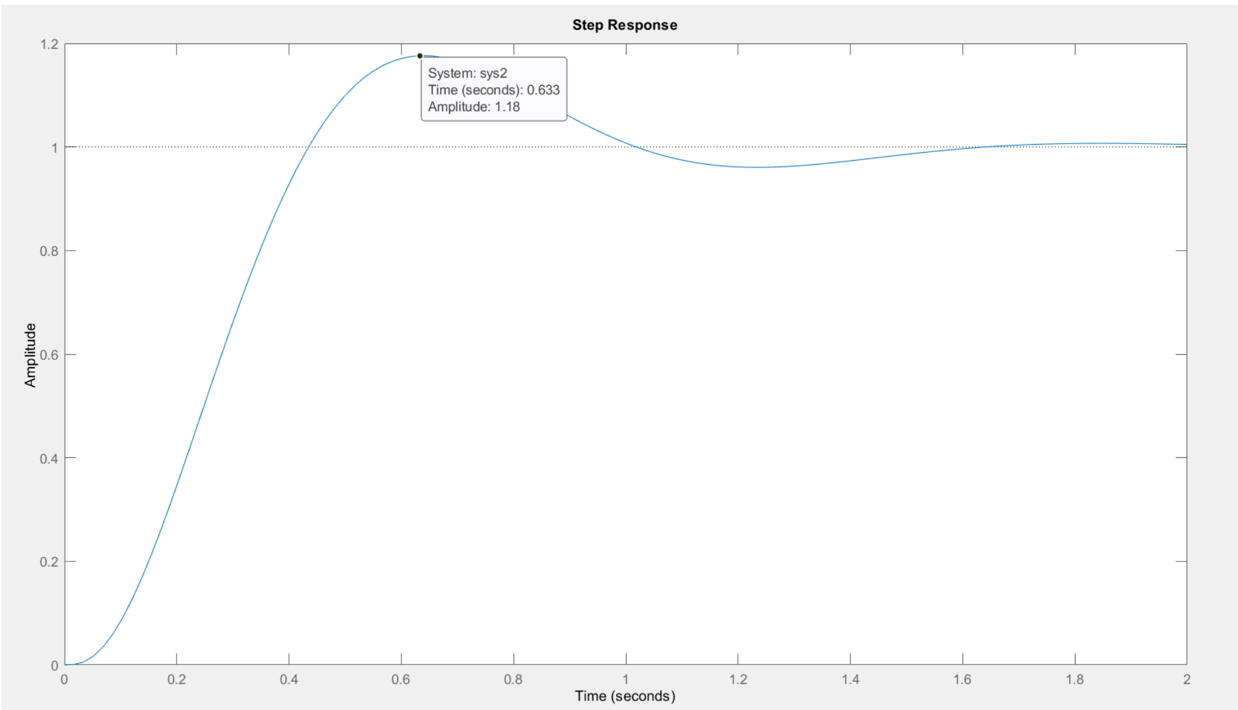
Note that the error is due to the second order approximation we did for calculating the %OS and T_s.

Step Response of Compensated System

Codes

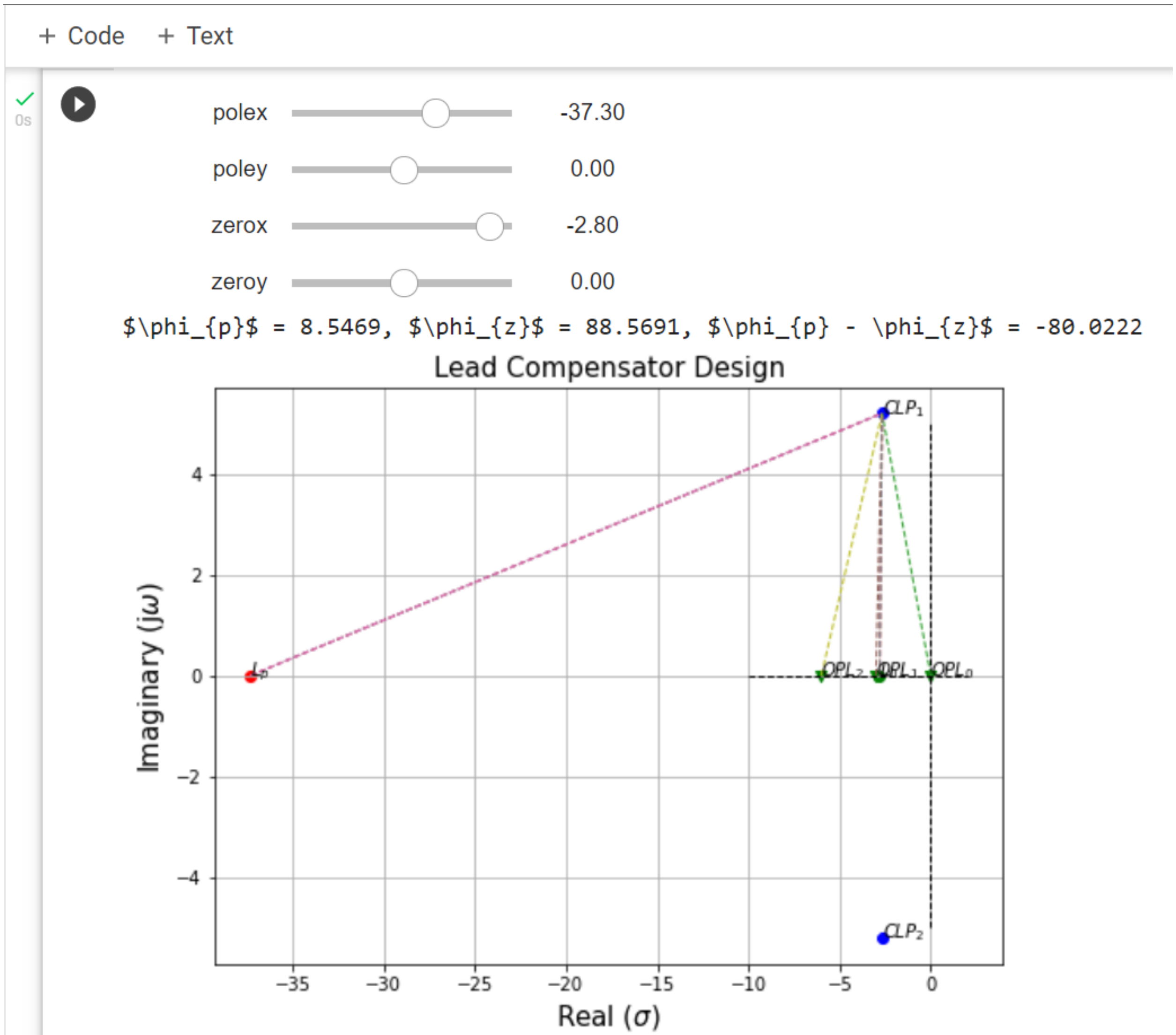
```
>>> step(sys)
```

Plots

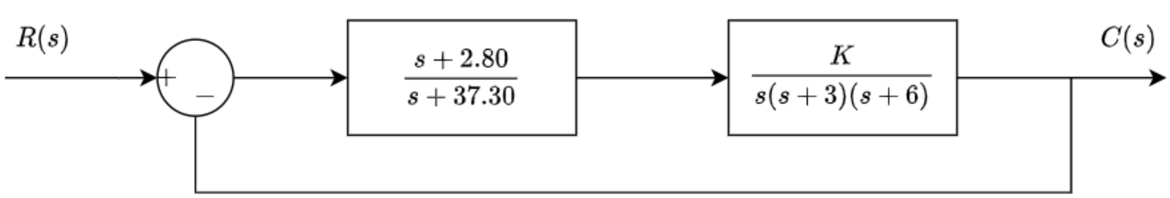


3 More Lead Compensators

First Compensator



Here, zero is at s = -2.80 and pole is at s = -37.30. Hence the overall system looks like

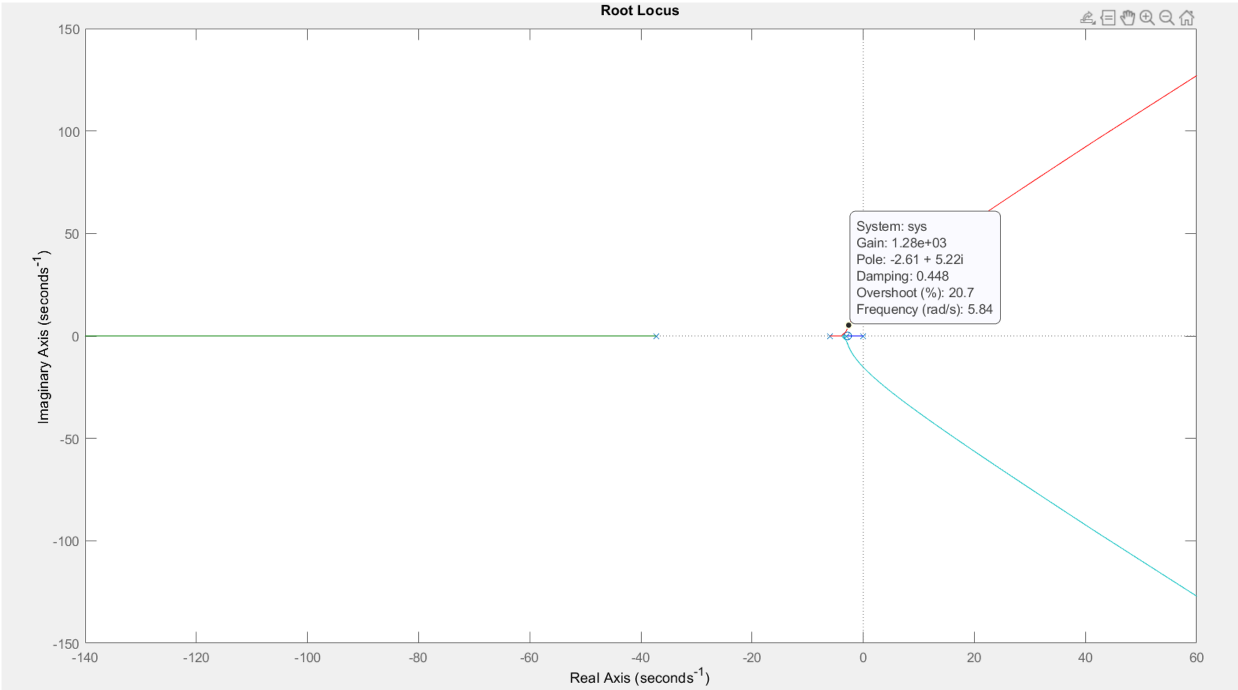


Root Locus

Codes

```
>> z = 2.80

z =

    2.8000

>> p = 37.30

p =

   37.3000

>> num = [1, z]

num =

    1.0000    2.8000

>> den = [1, 9+p, 9*(p+2), 18*p, 0]

den =

    1.0000   46.3000  353.7000  671.4000         0

>> sys = tf(num, den)

sys =

                 s + 2.8
  ------------------------------------
  s^4 + 46.3 s^3 + 353.7 s^2 + 671.4 s

Continuous-time transfer function.

>> rlocus(sys)
```

Plots



From the above figure of new root locus, we conclude that the gain of the system will be 1.28e+03 when the pole and zero are at -37.30 and -2.80 respectively. And we get %OS ~ 20.7% and settling time $T_s = 4/\zeta\omega_n = 1.528$ seconds, which is very close to the requirements. The overall transfer function is as follows

$$G_{overall}(s) = \frac{1280s + 3584}{s^4 + 46.3s^3 + 353.7s^2 + 1951s + 3584}$$

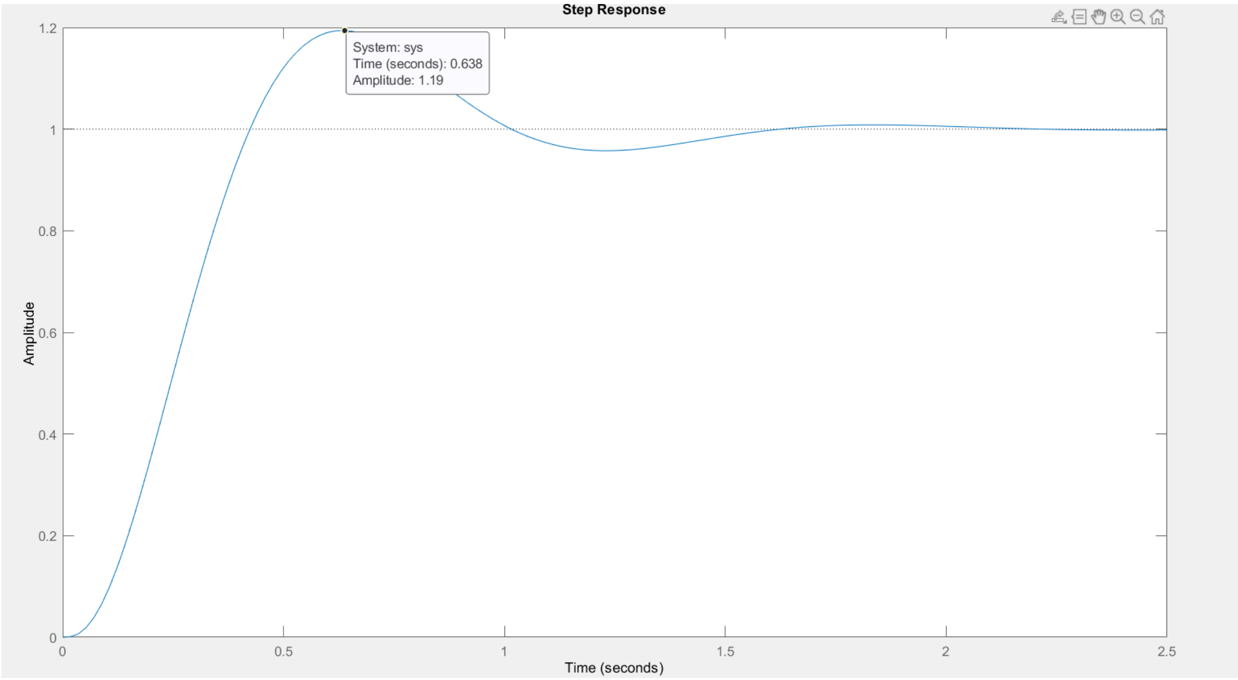
$$p, z, K = -37.30, -2.80, 1280$$

Step Response

Codes

```
>> p = 37.30;
>> z = 2.80;
>> K = 1280;
>> num = [K, K*z];
>> den = [1, p+9, 9*(p+2), 18*p + K, K*z];
>> sys = tf(num, den);
>> step(sys)
```

Plots



Step Response
System: sys
Time (seconds): 0.638
Amplitude: 1.19
Amplitude
Time (seconds)
1.2
1
0.8
0.6
0.4
0.2
0
0.5
1
1.5
2
2.5

Second Compensator

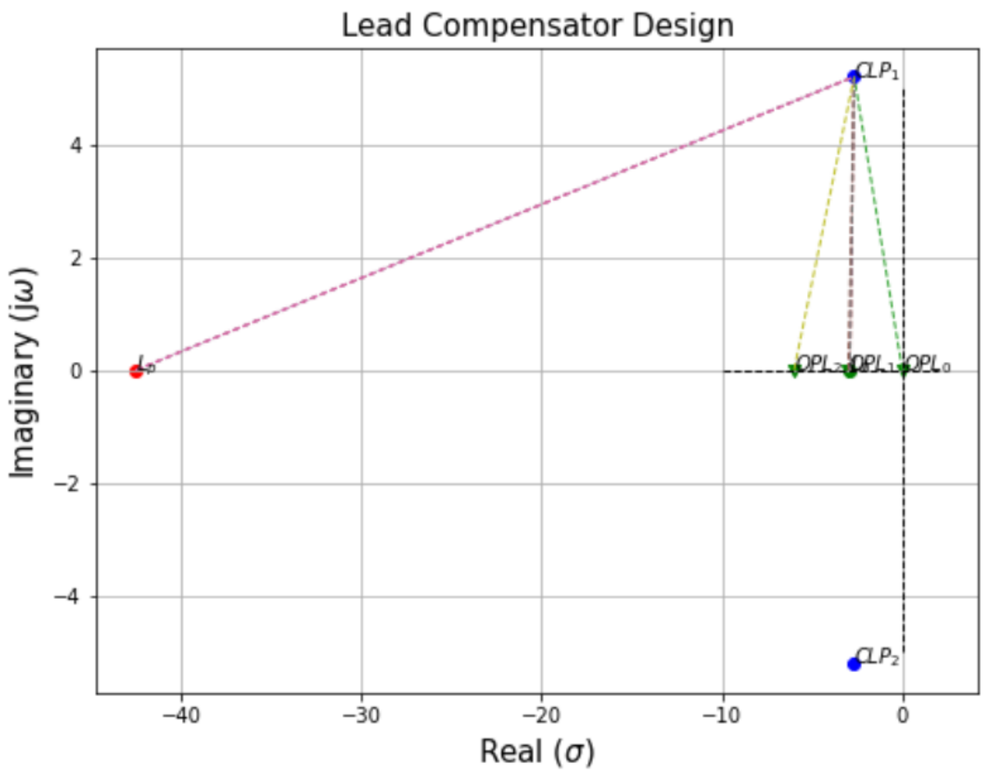
polex -42.48

poley 0.00

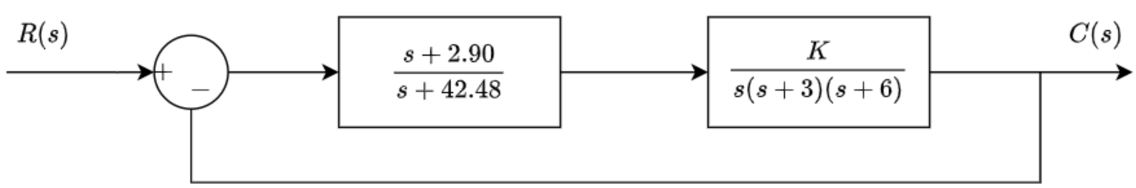
zerox -2.90

zeroy 0.00

```
$\phi_{p}$ = 7.4482, $\phi_{z}$ = 87.4696, $\phi_{p} - \phi_{z}$ = -80.0214
```



Here, zero is at s = -2.90 and pole is at s = -42.48. Hence the overall system looks like

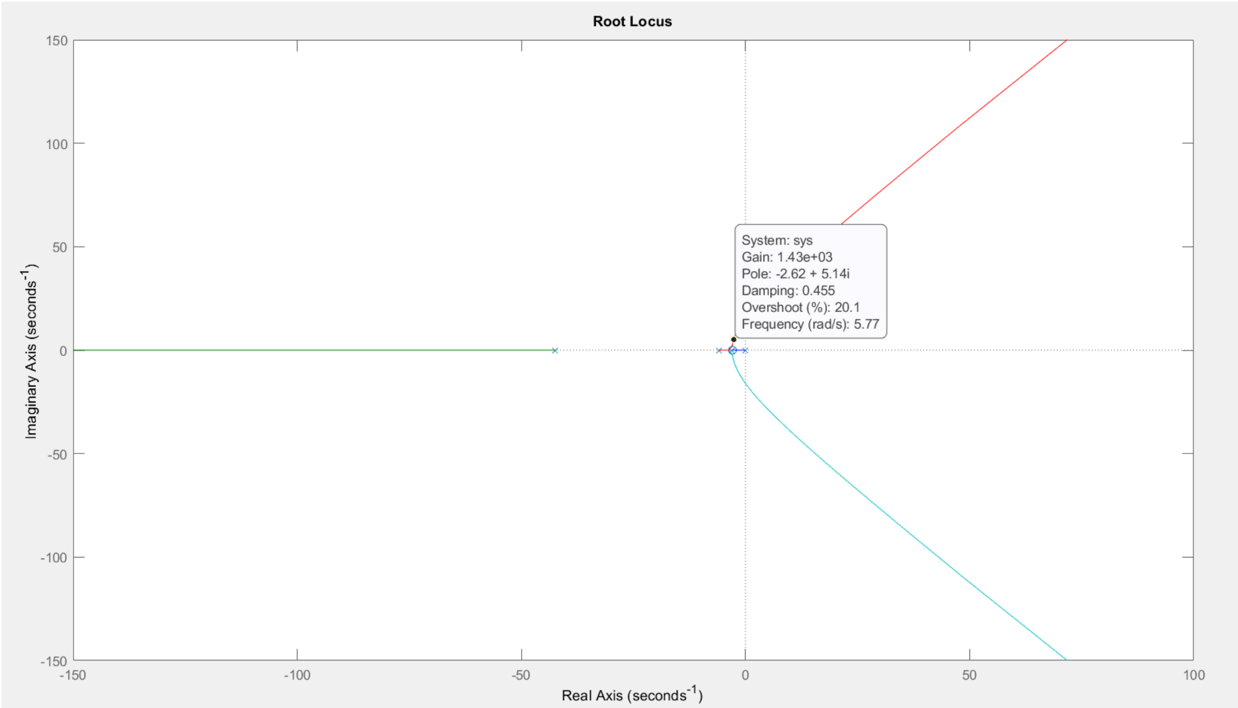


Root Locus

Codes

```
>> p = 42.48

p =

   42.4800

>> z = 2.90

z =

    2.9000

>> num = [1, z]

num =

    1.0000    2.9000

>> den = [1, p+9, 9*(p+2), 18*p, 0]

den =

    1.0000   51.4800  400.3200  764.6400         0

>> sys = tf(num, den)

sys =

                 s + 2.9
  -------------------------------------
  s^4 + 51.48 s^3 + 400.3 s^2 + 764.6 s

Continuous-time transfer function.

>> rlocus(sys)
```

Plots



From the above figure of new root locus, we conclude that the gain of the system will be 1.43e+03 when the pole and zero are at -42.48 and -2.90 respectively. And we get %OS ~ 20.1% and settling time $T_s = 4/\zeta\omega_n$ = 1.536 seconds, which is very close to the requirements. The overall transfer function is as follows:

$$G_{overall}(s) = \frac{1430s + 4147}{s^4 + 51.48s^3 + 400.3s^2 + 2195s + 4147}$$

$$p, z, K = -42.48, -2.90, 1430$$

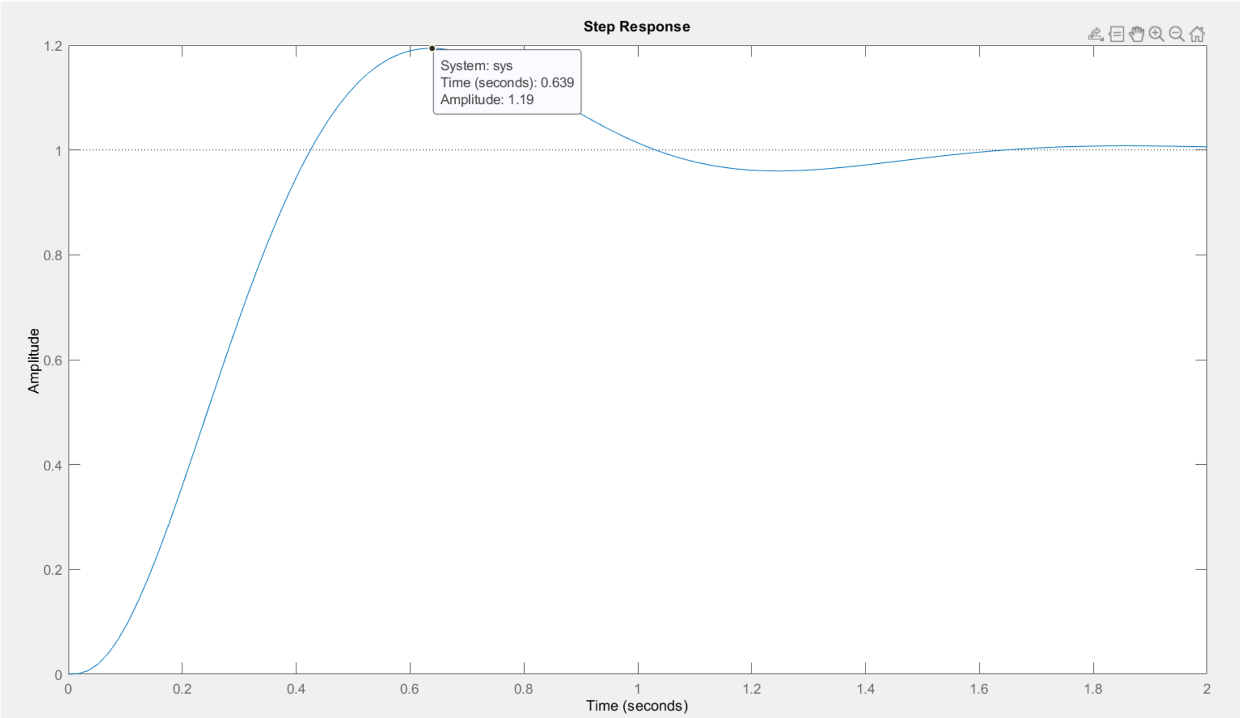
Step Response

Codes

```
>> z = 2.90;
>> p = 42.48;
>> K = 1430;
>> num = [K, K*z]

num =
```

```
        1430        4147

>> den = [1, p+9, 9*(p+2), 18*p + K, K*z]

den =

   1.0e+03 *

    0.0010    0.0515    0.4003    2.1946    4.1470

>> sys = tf(num, den)

sys =

                 1430 s + 4147
  -------------------------------------------
  s^4 + 51.48 s^3 + 400.3 s^2 + 2195 s + 4147

Continuous-time transfer function.

>> step(sys)
```

Plots


Step Response
System: sys
Time (seconds): 0.639
Amplitude: 1.19
Amplitude
Time (seconds)
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

Third Compensator

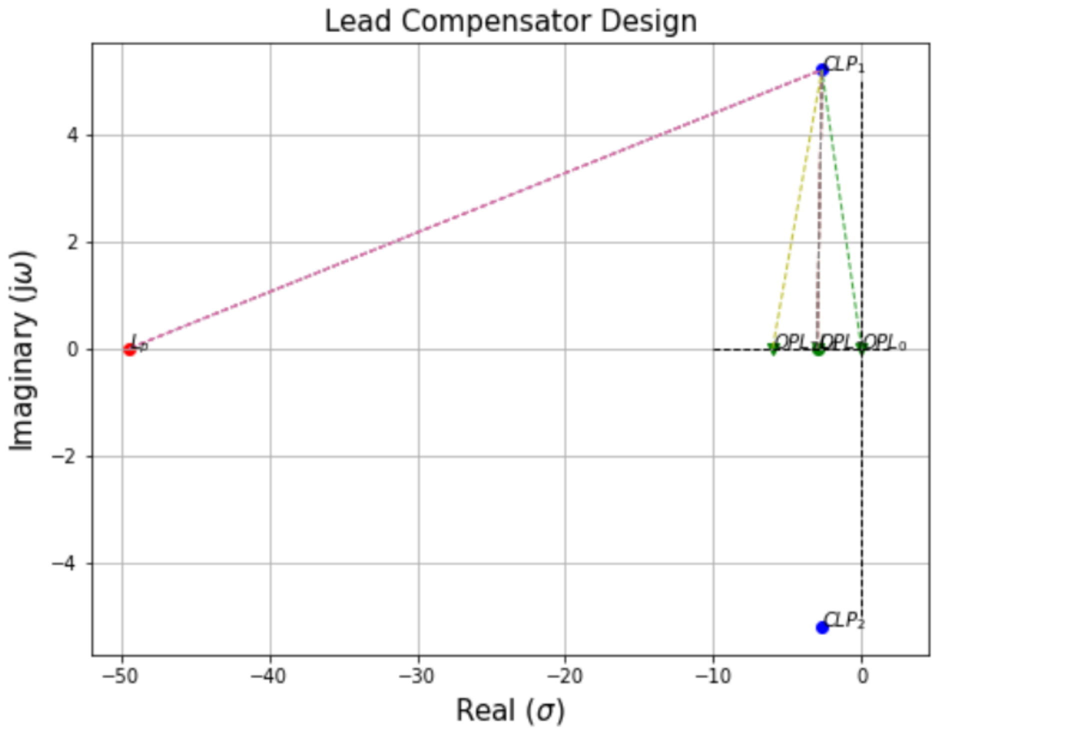
polex -49.43

poley 0.00

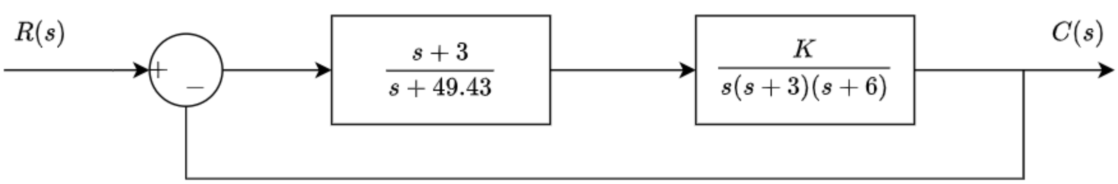
zerox -3.00

zeroy 0.00

```
$\phi_{p}$ = 6.3510, $\phi_{z}$ = 86.3719, $\phi_{p} - \phi_{z}$ = -80.0209
```



Here, zero is at s = -3.00 and pole is at s = -49.43. Hence the overall system looks like

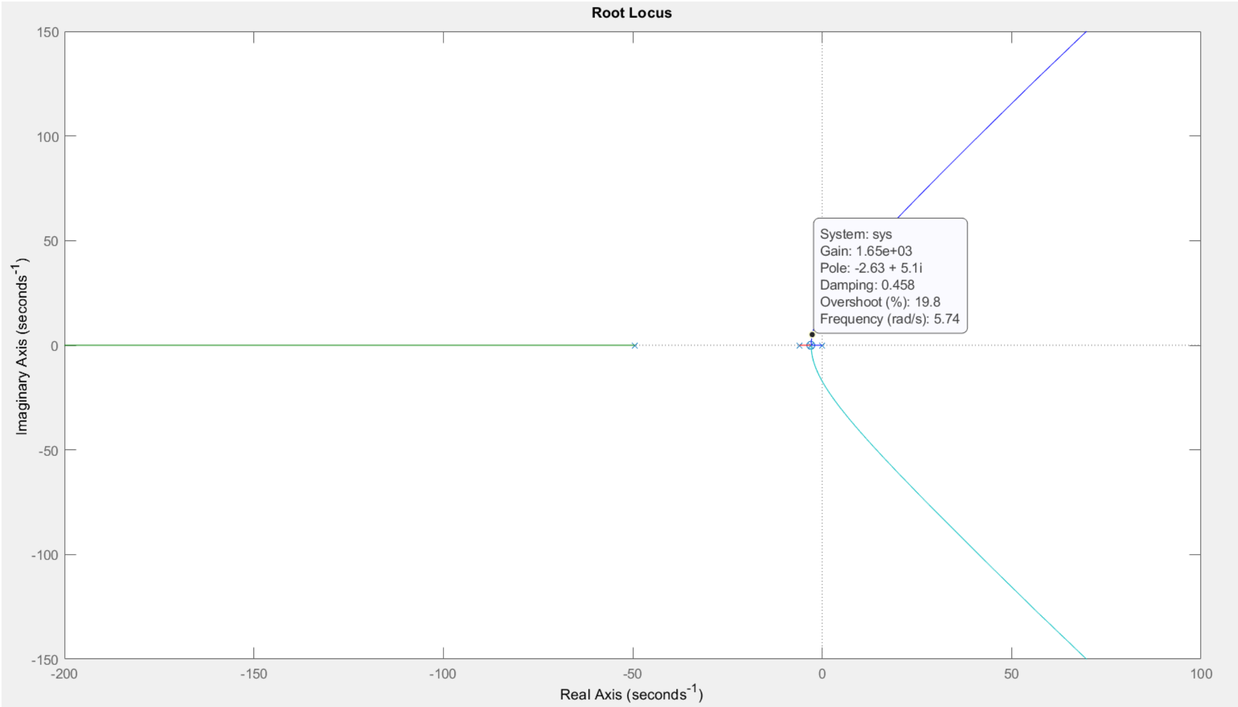


Root Locus

Codes

```
>> p = 49.43

p =

   49.4300

>> z = 3.0

z =

     3

>> num = [1, z]

num =

     1     3

>> den = [1, 9+p, 9*(p+2), 18*p, 0]

den =

    1.0000   58.4300  462.8700  889.7400         0

>> sys = tf(num, den)

sys =

                   s + 3
  -------------------------------------
  s^4 + 58.43 s^3 + 462.9 s^2 + 889.7 s

Continuous-time transfer function.

>> rlocus(sys)
```

Plots



From the above figure of new root locus, we conclude that the gain of the system will be 1.65e+03 when the pole and zero are at -49.43 and -3.0 respectively. And we get %OS ~ 19.8% and settling time $T_s = 4/\zeta\omega_n$ = 1.5215 seconds, which is very close to the requirements. The overall transfer function is as follows:

$$G_{overall}(s) = \frac{1650}{s^3 + 55.43s^2 + 296.58s + 1650}$$

$$p, z, K = -49.43, -3.0, 1650$$

Step Response

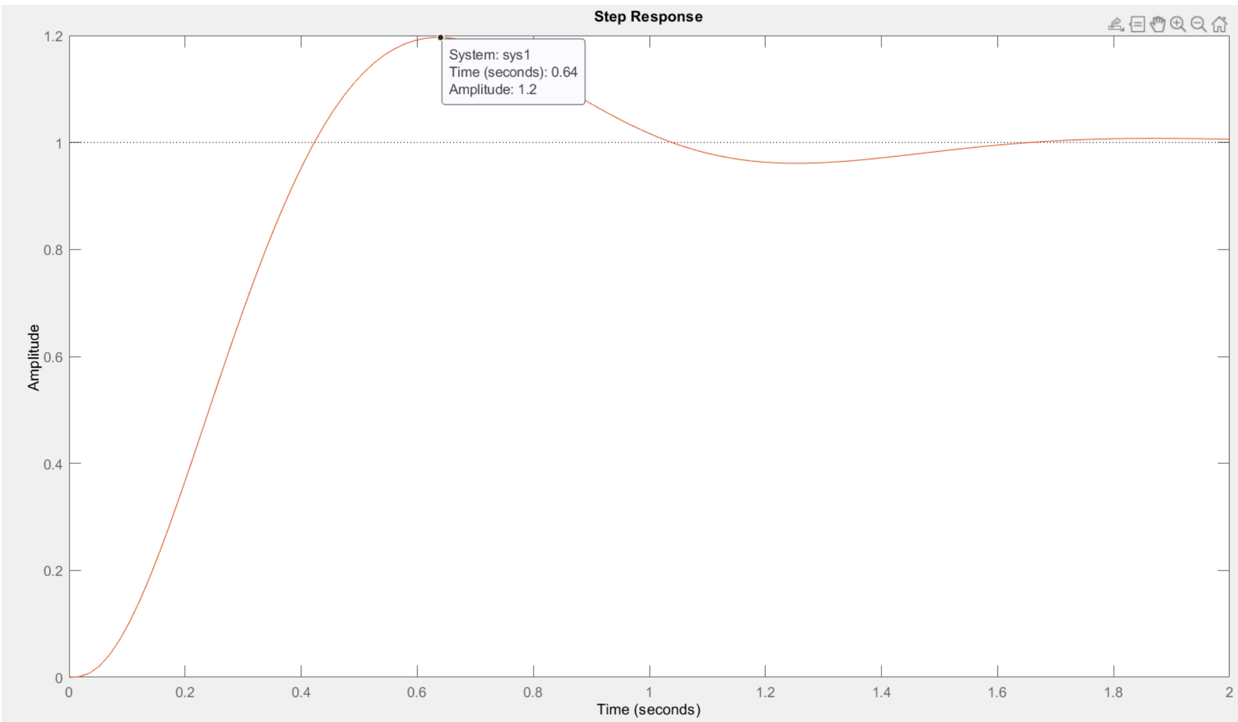
Codes

```
>> p = 49.43

p =

   49.4300
```

```
>> z = 3.0

z =

     3

>> K = 1650

K =

        1650

>> num = [K, K*z]

num =

        1650        4950

>> den = [1, p+9, 9*(p+2), 18*p + K, K*z]

den =

   1.0e+03 *

    0.0010    0.0584    0.4629    2.5397    4.9500

>> sys = tf(num, den)

sys =

                 1650 s + 4950
  -------------------------------------------
  s^4 + 58.43 s^3 + 462.9 s^2 + 2540 s + 4950

Continuous-time transfer function.

>> sys1 = tf([1650], [1, 6+p, 6*p, K])

sys1 =

                1650
```

```
  --------------------------------
  s^3 + 55.43 s^2 + 296.6 s + 1650

Continuous-time transfer function.
>> step(sys)
>> hold on
>> step(sys1)
```

Plots



(a) Total Angular Contribution of the Lead Compensator

All of the lead compensators yield the same angular contribution which is,

$$G_C(s) = \frac{s + 2.67}{s + 35.6057}$$

$$\angle G_C(s) = \angle(s + 2.67) - \angle(s + 35.6057)$$

$$\angle(s + 2.67) = \angle(\sigma + j\omega + 2.67)$$

$$But, \ \ \sigma = -2.67 \ \ and \ \ \omega = 5.2045$$

$$\angle(s + 2.67) = \angle(5.2045j) = 90^\circ$$

$$\angle(s + 35.6057) = \angle(-2.67 + 5.2045j + 35.6057)$$

$$\angle(s + 35.6057) = \angle(32.9357 + 5.2045j) = tan^{-1}\frac{5.2045}{32.9357} = 8.9796^\circ$$

$$\angle G_C(s) = 90 - 8.9796 = 81.0203^\circ$$

(b) Poles and Zeros of 2 more compensators

From First Compensator we have the zeros and poles as follows,
Pole @ s = -37.30 and Zero @ s = -2.80

From Second Compensator we have the zeros and poles as follows,
Pole @ s = -42.48 and Zero @ s = -2.90

From Third Compensator we have the zeros and poles as follows,
Pole @ s = -49.43 and Zero @ s = -3.00

(c) Expected Steady state error for step and ramp inputs, for each compensator

First Compensator (Original)

$$G_{closed}(s) = \frac{1210(s+2.67)}{s^4 + 45.6057s^3 + 338.45s^2 + 1850.90s + 3226.67}$$

$$G_{open}(s) = \frac{1210(s+2.67)}{s^4 + 45.6057s^3 + 338.45s^2 + 640.9s}$$

$$p, z, K = -35.6057, -2.67, 1210$$

$$K_p = lim_{s\to 0} G_{open}(s) = \frac{1210 \times 2.67}{0} \to K_p \to \infty$$

$$e_{ss,step} = \frac{1}{1+K_p} \to 0$$

$$K_v = lim_{s\to 0} sG_{open}(s) = \frac{1210 * 2.67}{640.9} = 5.040$$

$$e_{ss,ramp} = \frac{1}{K_v} = 0.198$$

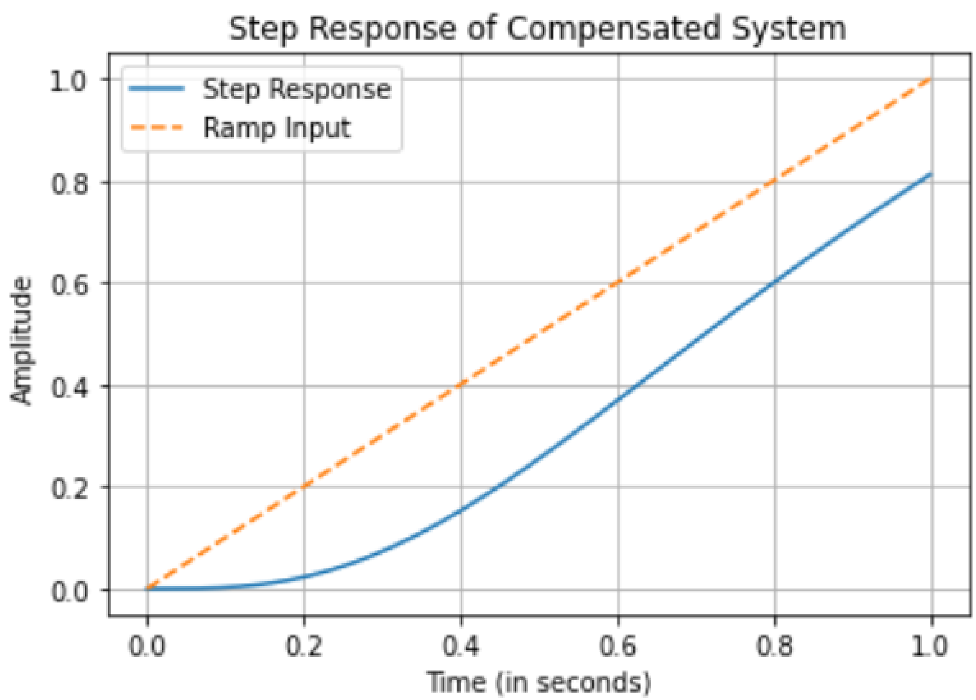
To provide evidence for please find the ramp response below of the system below

Ramp Response

```
1  interact(response, K=1200, p=35.6057, z=8/3)
```

K	1200
p	35.61
z	2.67
t_start	0
t_end	1

```
Steady State Error =  -0.18791005739152855
```




```
<function __main__.response>
```

Second Compensator (First Additional)

$$G_{closed}(s) = \frac{1280s + 3584}{s^4 + 46.3s^3 + 353.7s^2 + 1951s + 3584}$$

$$G_{open}(s) = \frac{1280s + 3584}{s^4 + 46.3s^3 + 353.7s^2 + 671.4s}$$

$$p, z, K = -37.30, -2.80, 1280$$

$$K_p = lim_{s\to 0} G_{open}(s) = \frac{1280 \times 2.80}{0} \to K_p \to \infty$$

$$e_{ss,step} = \frac{1}{1 + K_p} \to 0$$

$$K_v = lim_{s\to 0} sG_{open}(s) = \frac{1280 \times 2.80}{671.4} = 5.338$$

$$e_{ss,ramp} = \frac{1}{K_v} = 0.187$$

Ramp Response

K 1280

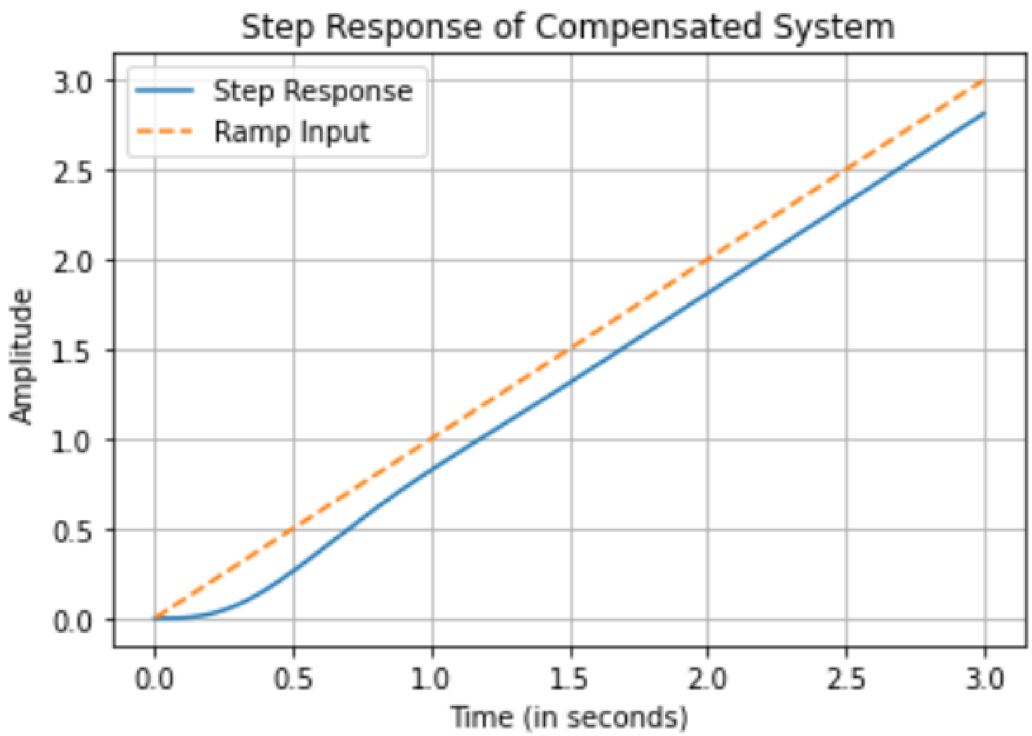
p 37.30

z 2.80

t_start 0

t_end 3

```
Steady State Error =  -0.18739960123625243
```



Third Compensator (Second Additional)

$$G_{overall}(s) = \frac{1430s + 4147}{s^4 + 51.48s^3 + 400.3s^2 + 2195s + 4147}$$

$$p, z, K = -42.48, -2.90, 1430$$

$$G_{open}(s) = \frac{1430s + 4147}{s^4 + 51.48s^3 + 400.3s^2 + 764.6s}$$

$$K_p = lim_{s \to 0} G_{open}(s) = \frac{1430 \times 2.90}{0} \to K_p \to \infty$$

$$e_{ss,step} = \frac{1}{1 + K_p} \to 0$$

$$K_v = lim_{s \to 0} sG_{open}(s) = \frac{1430 \times 2.90}{764.6} = 5.423$$

$$e_{ss,ramp} = \frac{1}{K_v} = 0.184$$

Ramp Response

K 1430

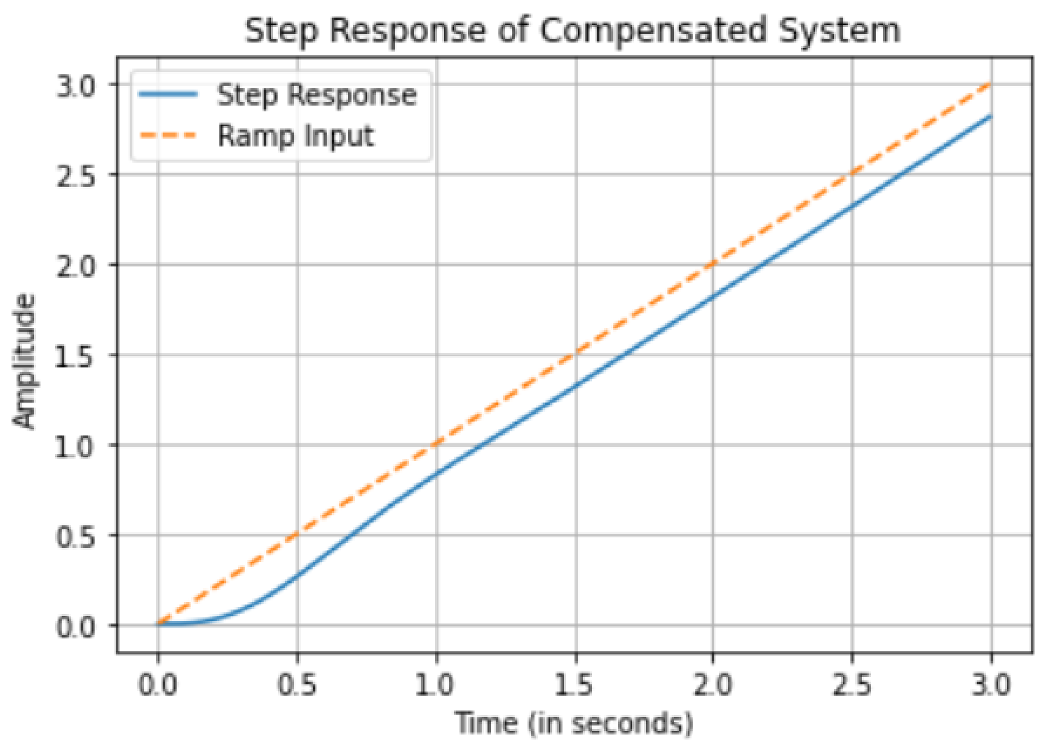
p 42.48

z 2.90

t_start 0

t_end 3

```
Steady State Error =  -0.1844583434173086
```



(d) PI Controller design for 0 steady state error for 'both' step and ramp inputs

Compensator Selection

We select the [third additional compensator](). Here is the steady state step and ramp input calculations

$$G_{overall}(s) = \frac{1650}{s^3 + 55.43s^2 + 296.58s + 1650}$$

$$G_{open}(s) = \frac{1650}{s^3 + 55.43s^2 + 296.58s}$$

$$p, z, K = -49.43, -3.0, 1650$$

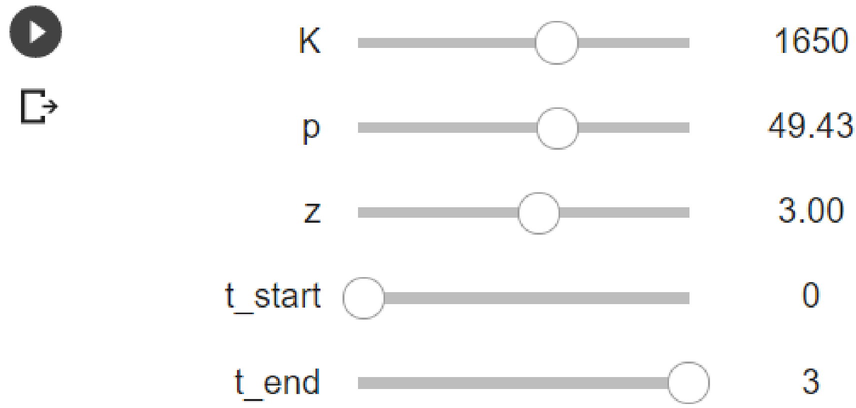
$$K_p = lim_{s \to 0} G_{open}(s) = \frac{1650}{0} \to K_p \to \infty$$

$$e_{ss,step} = \frac{1}{1 + K_p} \to 0$$

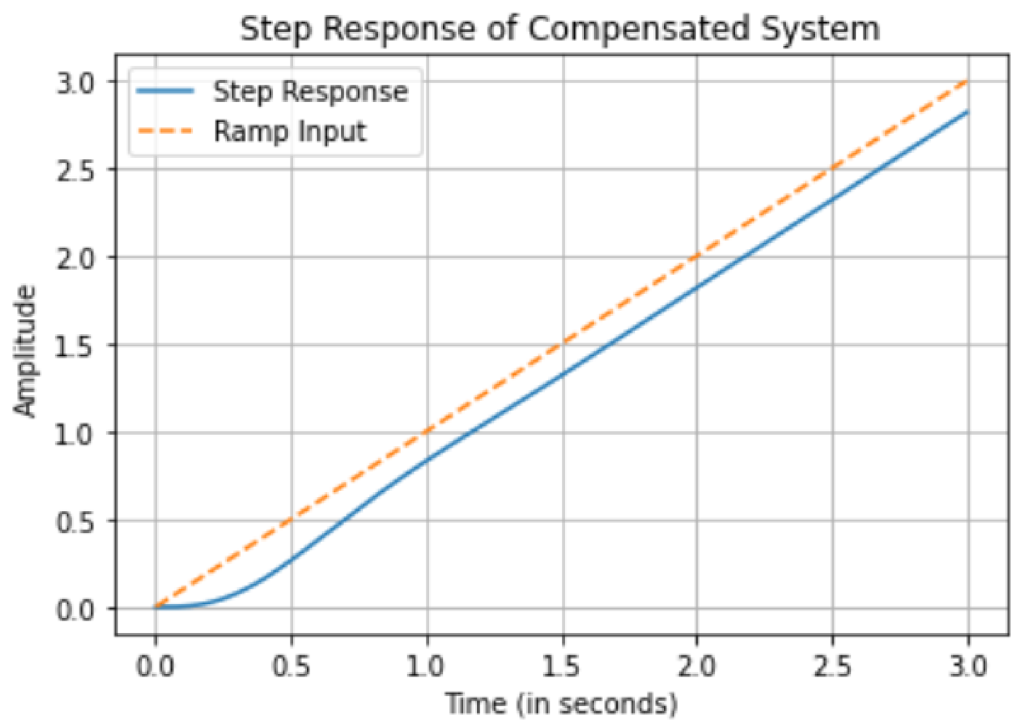
$$K_v = lim_{s \to 0} sG_{open}(s) = \frac{1650}{296.58} = 5.563$$

$$e_{ss,ramp} = \frac{1}{K_v} = 0.1797$$

Ramp Response

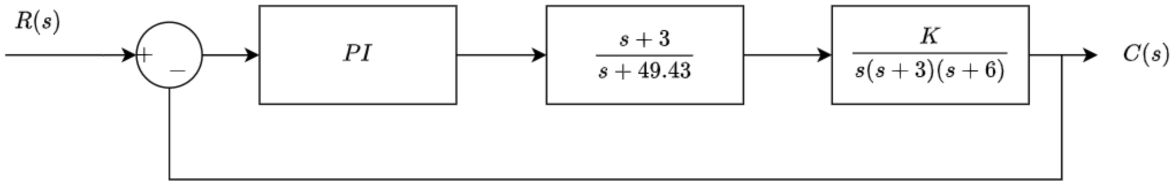



```
Steady State Error =  -0.17982270473748363
```



PI Controller Design

After cascading the PI controller the overall control system will look like,



Requirements

1. Step Steady State Error = 0
2. Ramp Steady State Error = 0

Now, step steady state error tending to zero is already satisfied since the $K_p \rightarrow \infty$
But the ramp steady state error is present. If it tends to zero, it means $K_v \rightarrow \infty$
Which means,

$$G_{open}(s) = \frac{1650}{s^3 + 55.43s^2 + 296.58s}$$

$$p, z, K = -49.43, -3.0, 1650$$

$$G_{PI,open}(s) = \frac{1650G_{PI}(s)}{s^3 + 55.43s^2 + 296.58s}$$

$$K_v = lim_{s\rightarrow 0}\, sG_{PI,open}(s) = 5.563 lim_{s\rightarrow 0} G_{PI}(s) \rightarrow \infty$$

$$Hence,\;\; lim_{s\rightarrow 0} G_{PI}(s) \rightarrow \infty$$

$$then, e_{ss,ramp} = \frac{1}{K_v} \rightarrow 0$$

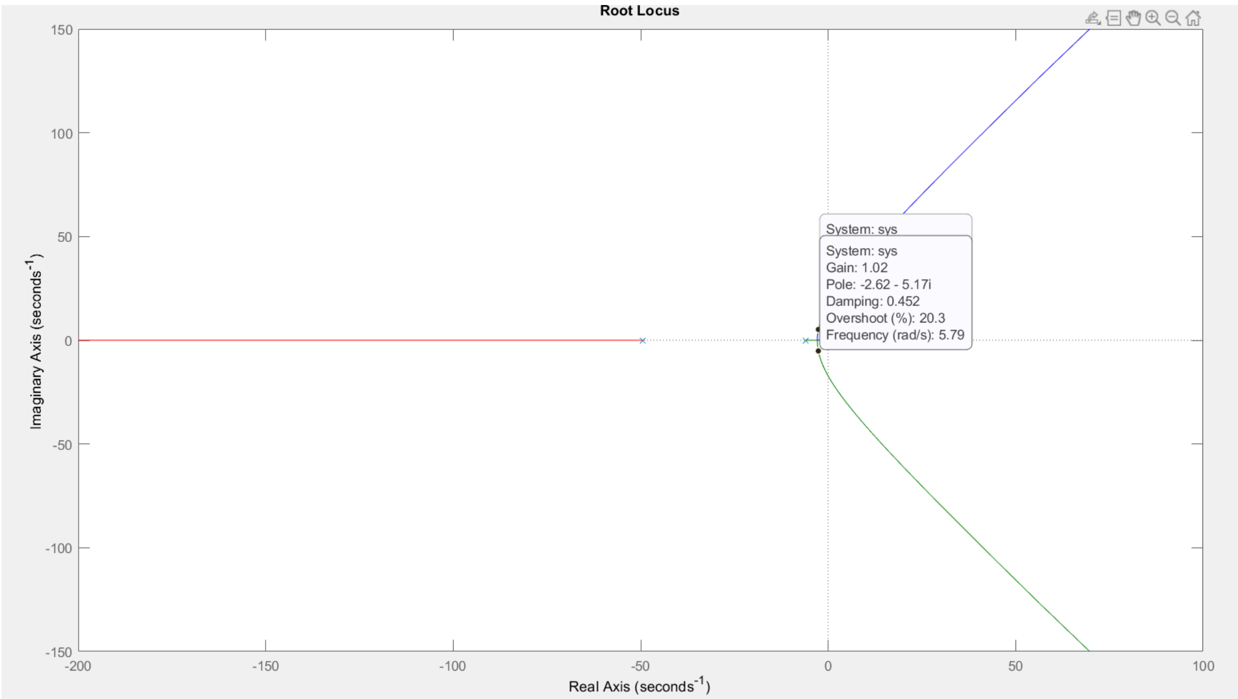
Root Locus of Uncompensated System

Codes (MATLAB)

```
>> sys = tf([1650], [1, 55.43, 296.58, 0])

sys =
```

```
             1650
  -------------------------
  s^3 + 55.43 s^2 + 296.6 s

Continuous-time transfer function.

>> rlocus(sys)
```

Plots



From the root locus we can conclude that dominant existing poles of the system are -2.67±5.2045j.

Solution

We consider the PI controller as $G_{PI}(s) = (K/T_is)(T_is + 1)$, we have,

From the K_v requirements, we get,

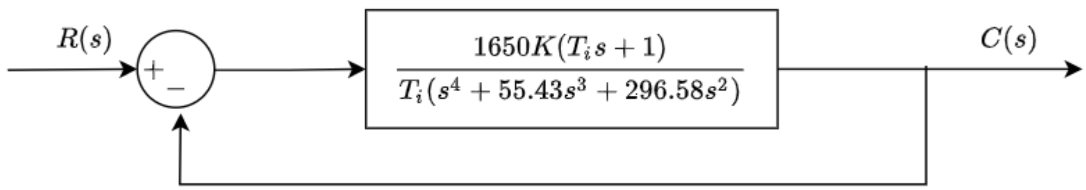
$$K_v = \lim_{s\to 0} sG_{PI}(s)G(s) = \lim_{s\to 0} s\frac{K}{T_i s}(T_i s + 1)\frac{1650}{s(s^2 + 55.43s + 296.58)} \to \infty$$

Hence we can conclude thatb for any gain K and any position of zero the $K_v \to \infty$.

From the K_p (step steady state error) requirements, we get,

$$K_p = \lim_{s\to 0} G_{PI}(s)G(s) = \lim_{s\to 0} \frac{K}{T_i s}(T_i s + 1)\frac{1650}{s(s^2 + 55.43s + 296.58)} \to \infty$$

Here, also the $K_p \to \infty$ irrespective of the values of K and T_i. The equivalent closed loop transfer function we get is as follows:



We follow the strategy of putting the zero very close to the origin and then tuning the gain to get the required parameters.

We arbitrarily place the zero at s = -0.05. Now, we automate the tuning procedure with the code below:

Tuning Gain Automation

```
def PI_both_response(K, z, start, end):
  time_range = np.linspace(start, end, 1000)
  sys = tf([1650*K, 1650*K*z], [1, 55.43, 296.58, 1650*K, 1650*K*z])
  time, resp = forced_response(sys, time_range, time_range)
  time2, resp2 = step_response(sys)
  ts = time2[np.where(abs(resp2 - 0.98)<1e-3)]

  fig, ax = plt.subplots(1, 2, figsize=(15, 5))

  ax[0].plot(time2, resp2, label='Step Response', linewidth=3)
  ax[0].plot(time2, np.ones_like(time2), label='Step Input',
linestyle='--')
  ax[0].plot(time2, np.ones_like(time2)*max(resp2), label='Overshoot',
linestyle='--')
  ax[0].legend(fontsize=15)
  ax[0].grid(1)
  ax[0].set_title('Step Response - PI Compensated', fontsize=15)
  ax[0].set_xlabel('Time (in seconds)', fontsize=15)
  ax[0].set_ylabel('Amplitude', fontsize=15)
  print('Steady State Error (Step) = ', abs(1-resp2[-1]))
  if len(ts)>=2:
```

```
    print('Settling time = ', ts[2])
  else:
    print('Settling time = ', ts[-1])
  print('Overshoot = ', max(resp2) - 1)

  ax[1].plot(time, resp, label='Ramp Response', linewidth=3)
  ax[1].plot(time_range, time_range, label='Ramp Input', linestyle='--')
  ax[1].legend(fontsize=15)
  ax[1].set_title('Step Response - PI Compensated', fontsize=15)
  ax[1].set_xlabel('Time (in seconds)', fontsize=15)
  ax[0].set_ylabel('Amplitude', fontsize=15)
  ax[1].grid(1)
  print('Steady State Error (Ramp) = ', time[-1] - resp[-1])

interact(PI_both_response, K=(0, 5, 0.01), z = (0, 0.1, 0.001), start=0,
end=20);
```

Tuning Gain Automation Results

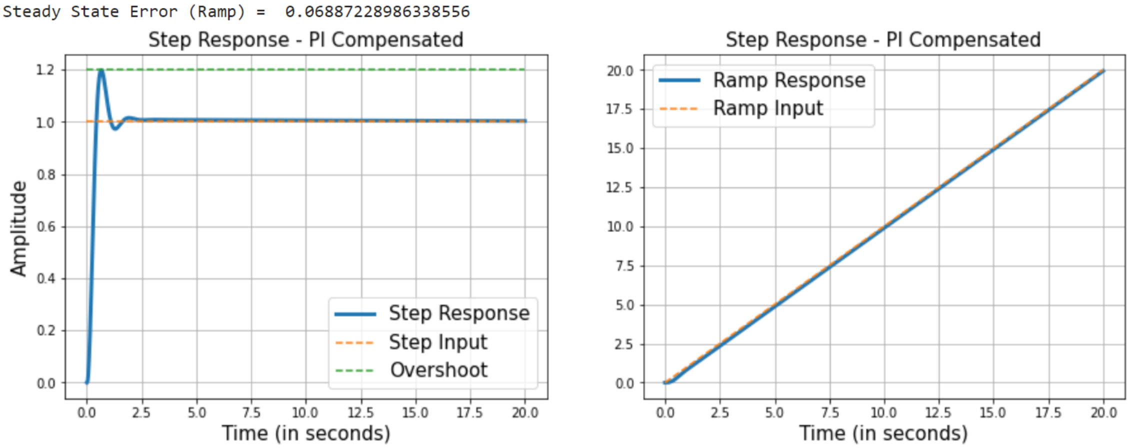
K 0.96

z 0.05

start 0

end 20

```
Steady State Error (Step) =  0.0034761552795754547
Settling time =  1.4214214214214214
Overshoot =  0.19924676984289547
Steady State Error (Ramp) =  0.06887228986338556
```



As seen from the figure, for the zero at s = -0.05 and the Gain K = 0.96 we are able to meet the requirements of %OS, settling time and steady state step and ramp inputs. Hence the design procedure is complete. The overall transfer function (PI Compensated) is as follows:

Final PI Compensated System

$$G_{closed} = \frac{1650 \times 0.96 \times (s + 0.05)}{s^4 + 55.43s^3 + 296.58s^2 + 1650 \times 0.96s + 1650 \times 0.96 \times 0.05} = \frac{1584(s + 0.05)}{s^4 + 55.43s^3 + 296.58s^2 + 1584s + 79.2}$$

Code Files and Latex Outputs [only LDAP]

Access the Codes here (also attached in moodle in .ipynb format)
https://colab.research.google.com/drive/1TPMR4Q3_HZALGuuXNjlgLImNidvYg9VD?usp=sharing

MATLAB Codes given in the report itself wherever used.