Faculty of Electrical Engineering

Design of Lead and Lag Compensators
In Electrical Engineering - Control Specialization

Student Name:
Amirrreza Zarrabinia

Professor Name:
Dr. Soheil Ganjefar

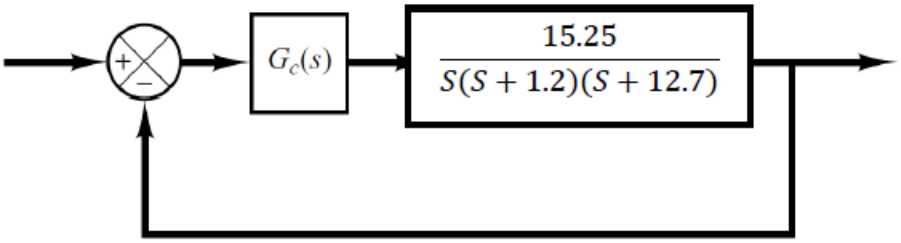
January 2023

Table of Contant

Chapter 1:
Introduction

1-1: Introduction

A closed-loop system is defined as follows:



We will now proceed to design a controller for this system to meet the requirements of the problem.

We will first start with two conventional methods: graphical and pole cancellation. Then, we will attempt to improve the design by adjusting the zeros and poles of the controller. Finally, we will evaluate the advantages and disadvantages of each method.

The problem requirements specify a 15% overshoot and a 4-second settling time. Additionally, as designers, we must reduce the steady-state error in response to a ramp input by designing a Lag controller.

1-2: Analysis of Desired Points

We know that:

$$\xi = \frac{-\ln Mp}{\sqrt{\pi^2 + ln^2 Mp}}$$

Additionally, for the settling time, we have:

$$t_s = \frac{4}{\xi \omega_n}$$

Now, we will define the desired points in MATLAB once and for all, so that we can use them in the design of other controllers as well.
We know that $\cos\beta = \xi$, so we have the angle of the desired points. Furthermore, we know that the radius of this circle is ω_n, and these two parameters give us the exact points through which the root locus must pass.

```
Mp = 0.15;
zeta = -log(Mp)/(sqrt(pi^2 + log(Mp)^2))
zeta = 0.5169
```

```
ts = 4;
Wn = 4/(ts*zeta)
Wn = 1.9345
```

```
beta = acos(zeta);
beta = rad2deg(beta);
xpoint = -zeta*Wn
xpoint = -1
tan_beta = tan((beta*pi)/180);
ypoint = tan_beta*zeta*Wn
ypoint = 1.6560
```

(1-1)

1-3: Initial Root Locus Plot

Plotting the root locus in MATLAB is straightforward once we have the transfer function.
Here, along with plotting the root locus, we will also display the desired points on the plot. As a result, if the desired points lie on the root locus, the system will only require a simple gain adjustment.

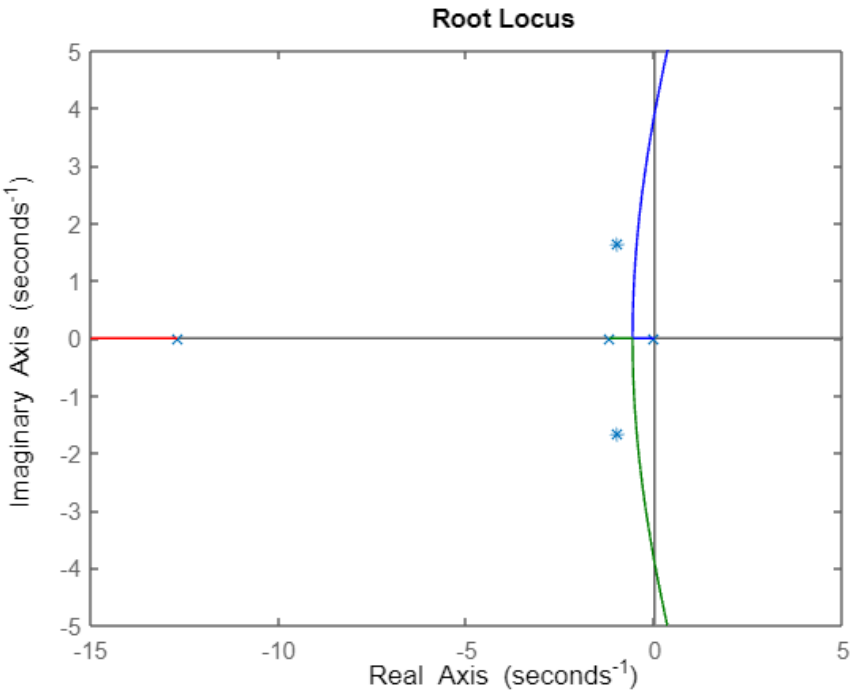
However, it is possible that the desired points are to the right of the root locus. In this case, shifting the root locus to the right will slow down the system's response because as $\xi\omega_n$ decreases, the settling time increases. Therefore, as designers, we may conclude that the previous points are more desirable for the system.

In the third scenario, if the desired points are to the left of the root locus, we will need to use a Lead controller (which will be explained later) to move the system to the desired points. Naturally, in this case, the settling time will be shorter, and the controller will optimize the system.

The transfer function of the system is as follows:
We input this transfer function into MATLAB and use the `rlocusplot` command to plot the initial root locus along with the desired points. The `rlocusplot` command also allows us to specify additional options, such as setting the range of the root locus plot.

$$G(s) = \frac{15.25}{s(s+1.2)(s+12.7)}$$

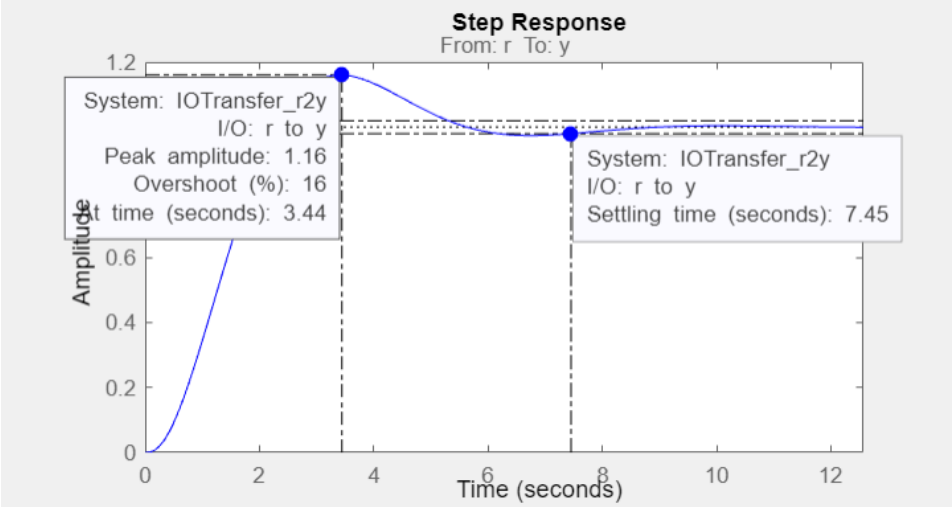
```
num1 = 15.25;
den1 = [1 13.9 15.24 0];
g1 = tf (num1,den1);
rlocusplot (g1);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



1-4: Initial Step Response of the System

We can analyze the step response and characteristics such as overshoot and settling time for this closed-loop system before applying the controller. This will give us a clearer view of how the system's characteristics improve after the controller is designed and implemented.

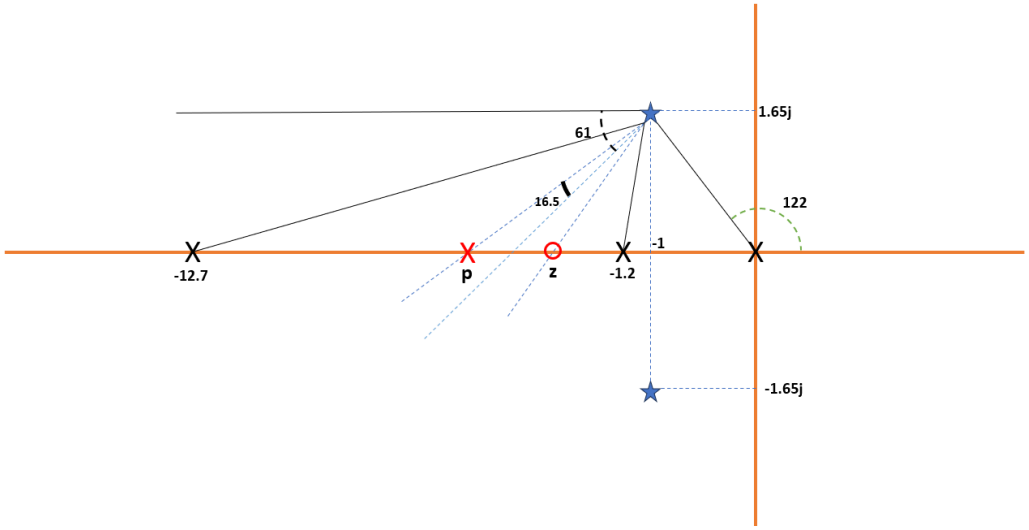
To achieve this, we will use a tool called **sisotool** to evaluate the system's step response and its characteristics in response to a step input. This allows us to examine the system's behavior and performance before controller application.



It is observed that the system has a very large settling time, which needs to be properly corrected through controller design.
One important point regarding overshoot is that as overshoot decreases, the system's speed also decreases. Therefore, if the problem does not have strict requirements on overshoot, it is preferable to allow for a maximum acceptable overshoot to maintain the system's speed and avoid slowing it down.

Chapter 2:
Lead Controller Design via Graphical Method and Lag

2-1: Lead Controller Design and Analysis



(2-1) Location of Lead Controller Zero and Pole in the Graphical Method

In Figure (1-1), we saw that the angle β was calculated to be approximately 58 degrees. Therefore, the bisector drawn in the graphical method should be plotted at an angle of 61 degrees, as shown in the figure. The rest of the calculations related to Figure (1-2) are as follows:

$$\theta_z - \theta_p = \varphi = 180 + 122 + tg^{-1}\frac{1.65}{0.2} + tg^{-1}\frac{1.65}{11.7}$$

$$\rightarrow \varphi \simeq 33 \rightarrow \frac{\varphi}{2} = 16.5$$

From the figure, we can understand that:

$$\theta_p = 61 - 16.5 = 44.5 \rightarrow \theta_z = \varphi + 44.5 = 77.5$$

Next, in MATLAB, we will determine the values of z (zero) and p (pole) using the calculated angles.
After that, we need to calculate the desired gain for the controller using the magnitude condition, which can be seen in the following code.

```
teta_p = 44.5;
teta_z = 77.5;
x = ypoint/(tan((44.5*pi)/180));
p = x + 1
p = 2.6851
y = ypoint/(tan((77.5*pi)/180));
z = y + 1
z = 1.3671
```

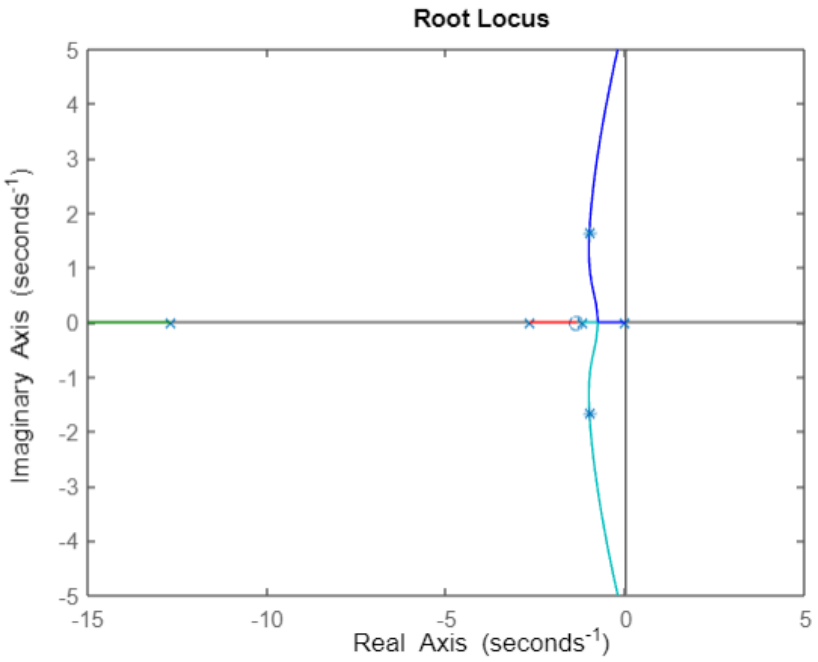
```
s = -1 + 1.65j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2)*(s+12.7);
k = abs(den2/num2)
k = 3.4666
```

As a result, the transfer function of our system changes as follows after applying the Lead controller:

$$G_c(s)G(s) = \frac{3.46(s+1.36)}{s+2.68}\frac{15.25}{s(s+1.2)(s+12.7)}$$

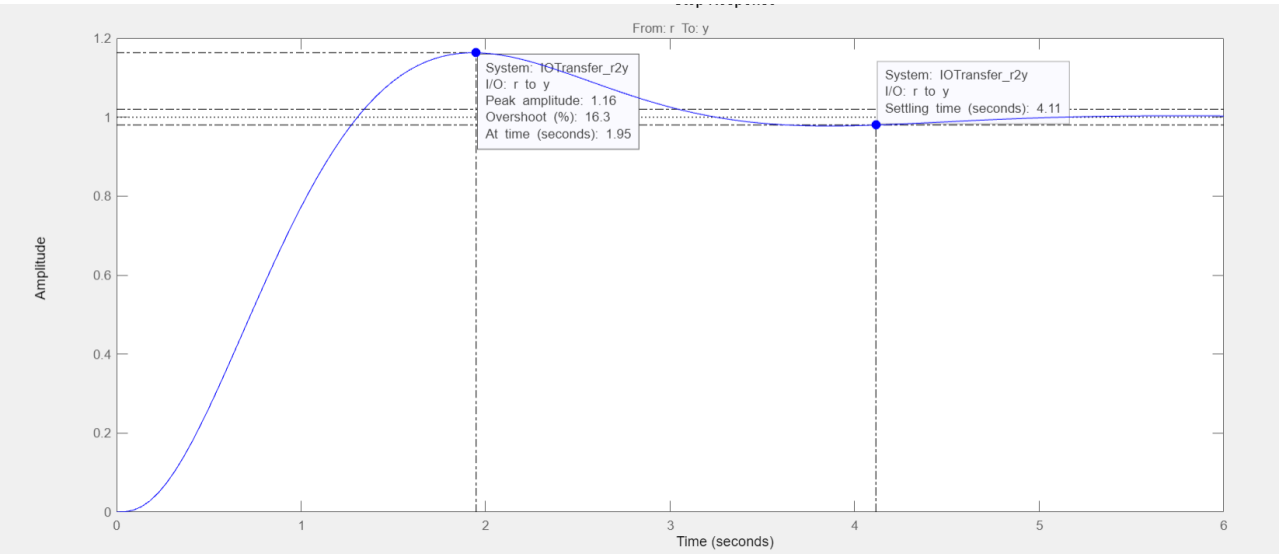
Now, we will plot the root locus of the new system to examine whether the root locus passes through the desired points or not.

```
num3 = [k*15.25 k*15.25*z];
den3 = [1 13.9+p p*13.9+15.24 15.24*p 0];
g2 = tf(num3,den3);
rlocusplot (g2);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



As shown, the root locus passes through the desired points effectively after designing the Lead controller.

Next, we want to check whether the system meets the desired criteria after the Lead controller design. As mentioned earlier, we will use the **sisotool** to analyze the system's performance and verify if the required specifications have been achieved.



It is observed that the settling time is much more desirable compared to the state before applying the controller, but it still falls short of the desired value.
Even though the settling time has decreased with this method, the overshoot has slightly increased compared to the initial state.

2-2: Lag Controller Design and Analysis

As mentioned earlier, systems may have a steady-state error in response to a ramp input, and this error can be partially compensated by designing a Lag controller. To determine the zero and pole values of the Lag controller, we use the following two equations:

$$z = \frac{\alpha}{10}$$

$$\frac{K_v old}{K_v new} = \frac{z}{p}$$

α represents the distance of the desired points from the $j\omega$-axis.

The old velocity constant $K_v old$ is calculated using the equation below, while the new $K_v new$ is provided by the problem.

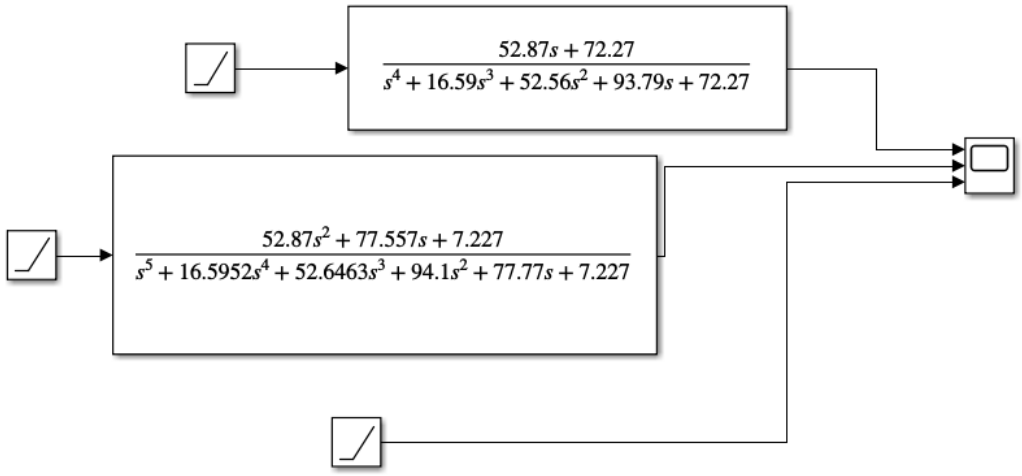
$$K_v old = \lim_{s \to 0} sG_c(s)G(s) = 1.766$$

Thus, the zero and pole values of the Lag controller can be easily calculated using these relationships.

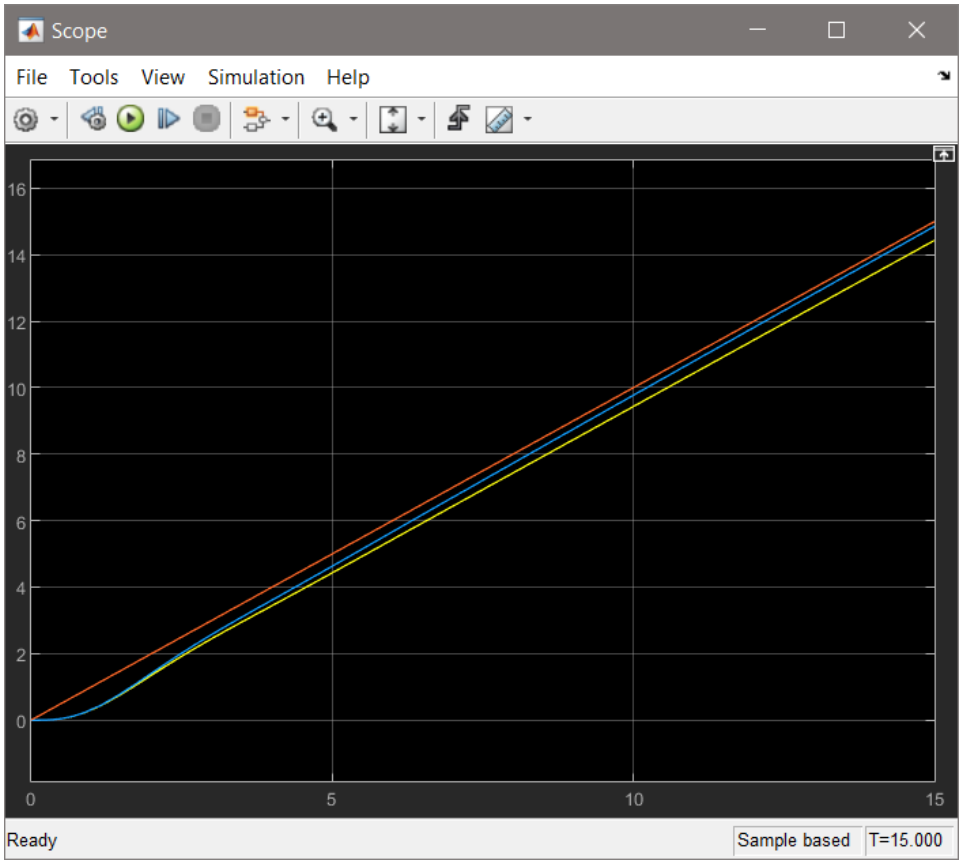
```
K_old = 1.766;
K_new = 34;
z2 = -xpoint/10
z2 = 0.1000
p2 = (z2*K_old)/K_new
p2 = 0.0052
```

Next, to observe the performance of the Lag controller, we need to examine the system's response to a ramp input.
To do this, we must input the ramp signal into the closed-loop transfer function within the Simulink environment.
The closed-loop transfer function can be obtained using the following equation:

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



Now, we want to observe the output of these transfer functions, where the first one represents the system before applying the Lag controller and the second one represents the system after applying the Lag controller:



It is evident that after applying the Lag controller, the steady-state error in response to the ramp input has decreased.

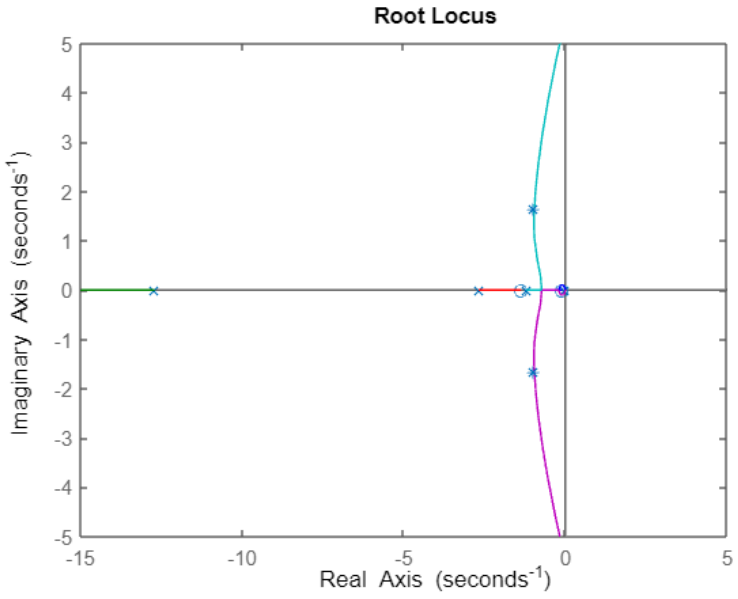
2-3: System After Lead-Lag

As a result, the final transfer function of the system after designing the Lead-Lag controller is as follows:

$$G(s) = \frac{3.46 \times 15.25(s+1.36)(s+0.1)}{s(s+1.2)(s+12.7)(s+2.68)(s+0.0052)}$$

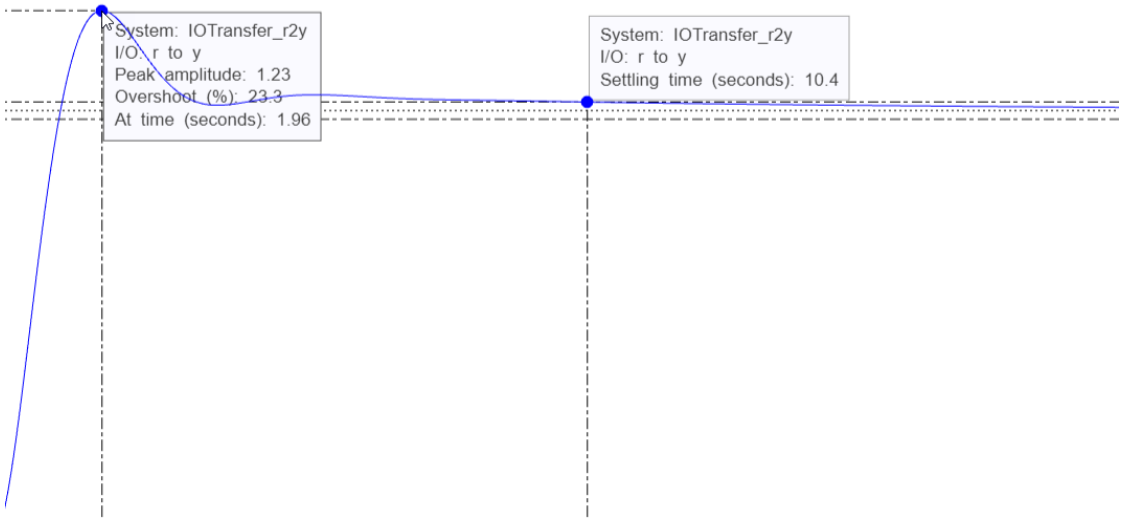
By plotting the root locus for this transfer function, we will obtain:

```
num4 = [52.87 77.557 7.227];
den4 = [1 16.5952 52.6463 41.2233 0.2128 0];
g3 = tf(num4,den4);
h = rlocusplot (g3);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



It is observed that after applying the Lag controller, the root locus deviates slightly from the desired points.

If we plot the step response of the system after designing the Lead-Lag controller, we will notice that the step response has significantly changed and no longer aligns well with the desired characteristics:



At this point, we need to find a method to improve the step response. After testing several approaches, as outlined below, we conclude that the best way to improve the response is by adjusting the zero of the Lag Controller.

First, changes to the gain were examined. Although increasing the gain brought the settling time closer to the desired value, it also caused a significant increase in overshoot, which was not acceptable for our requirements.

Next, I manually adjusted the pole and achieved a relatively good result, but in this approach, the velocity constant K_v deviated significantly from the problem's requirements.

Finally, I adjusted the zero of the Lag Controller, which is shown below. This approach provided a more satisfactory outcome.

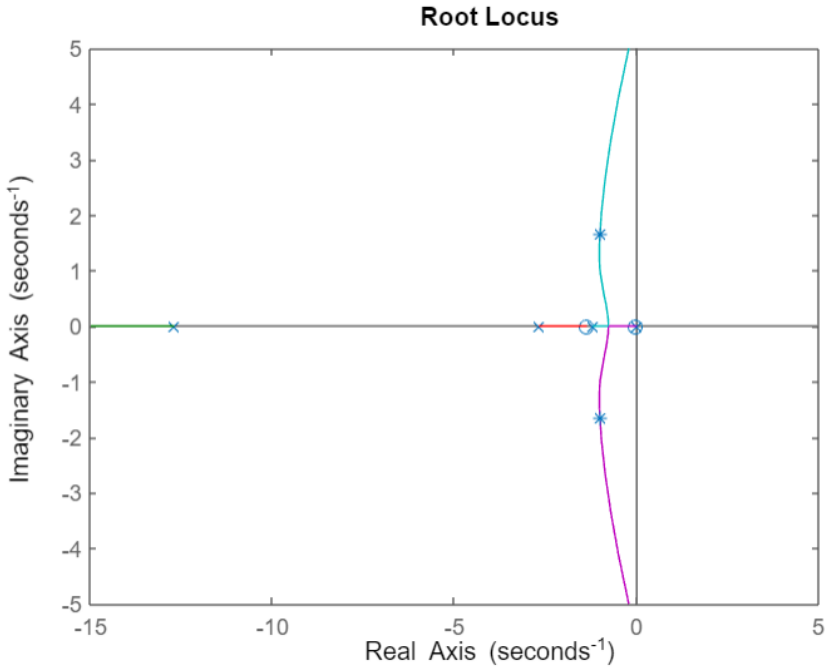
```
K_old = 1.766;
K_new = 34;
z2 = 0.1/10
z2 = 0.0100
p2 = (z2*K_old)/K_new
p2 = 5.1941e-04
```

Another important point to note is that with this adjustment of the zero, the velocity constant K_v reaches its desired value of 34.
The transfer function of the system after designing the Lead-Lag controller is as follows:

$$G(s) = \frac{3.46 \times 15.25(s+1.36)(s+0.01)}{s(s+1.2)(s+12.7)(s+2.68)(s+0.00052)}$$

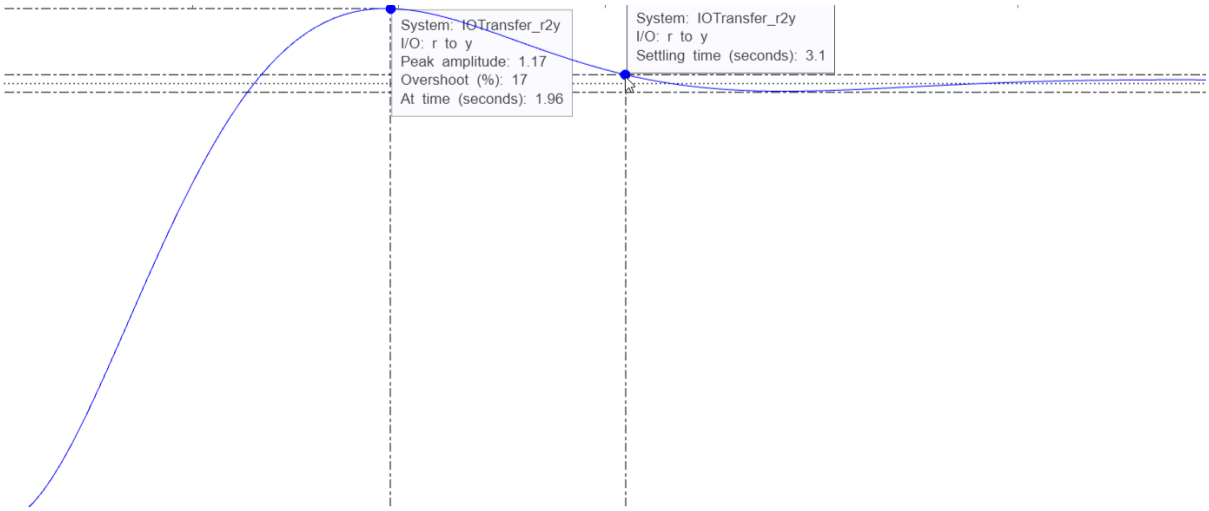
Now, we will plot the root locus of the system.

```
num4 = [52.8661 72.803 0.7227];
den4 = [1 16.5856 52.572 40.949 0.0213 0];
g3 = tf(num4,den4);
h = rlocusplot (g3);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



It is clear that after adjusting the zero of the Lag Controller, the root locus passes through the desired points more effectively.

Now, let's examine the step response of the system.



It is quite clear that with this adjustment, we have come much closer to achieving the desired system characteristics.

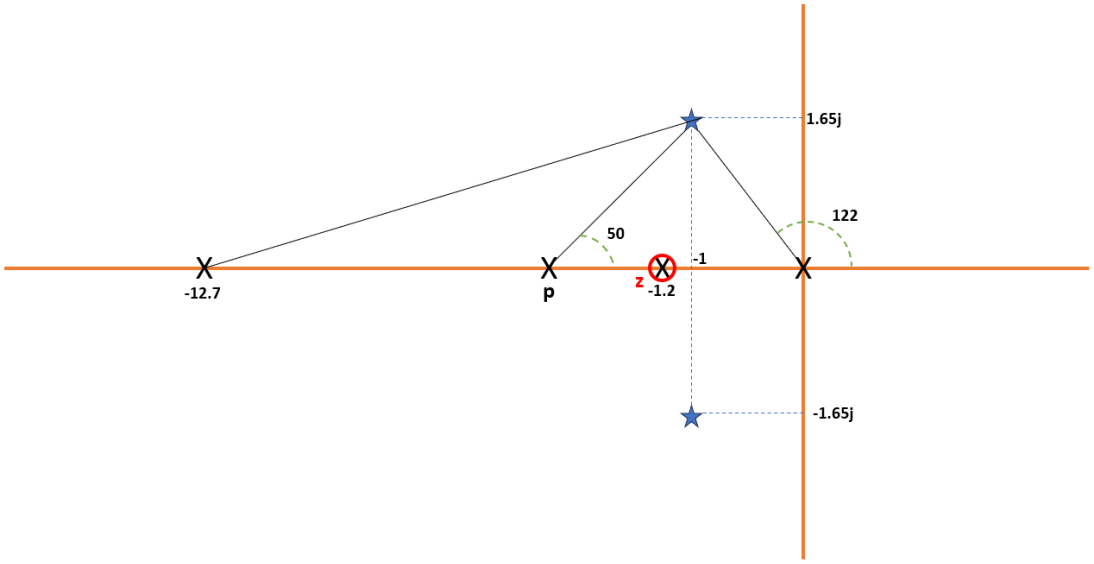
Chapter 3:
Lead Controller Design via Pole Cancellation and Lag

3-1: Lead Controller Design and Analysis

We now proceed with the second method of designing the Lead controller.

Given that a Lead controller has three unknowns -zero, pole, and gain- it is sometimes necessary to assign a value to one of these unknowns. In this case, we use the pole-cancellation method, where we place the zero of the Controller on one of the system poles.

For this design, we place the zero at the pole $p = -1.2$, as shown in the figure below.



Next, to calculate the angle, the value of p (pole), and the gain of the controller, we will proceed as follows:

$$-\theta_p = 180 + 122 + tg^{-1}\frac{1.65}{11.7}$$

$$\rightarrow \theta_p \simeq 50$$

```
z = 1.2;
teta_p = 50;
x = ypoint/(tan((teta_p*pi)/180));
p = x + 1
p = 2.3895
```

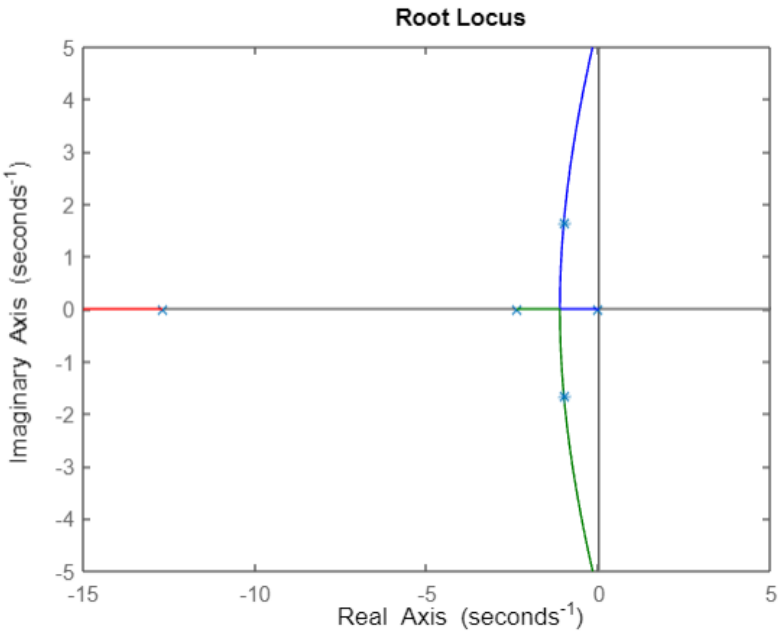
```
s = -1 + 1.65j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2)*(s+12.7);
k = abs(den2/num2)
k = 3.2247
```

As a result, the transfer function of the system after designing the Lead controller using the pole cancellation method becomes as follows:

$$G_c(s)G(s) = \frac{3.22 \times 15.25}{s(s+12.7)(s+2.39)}$$

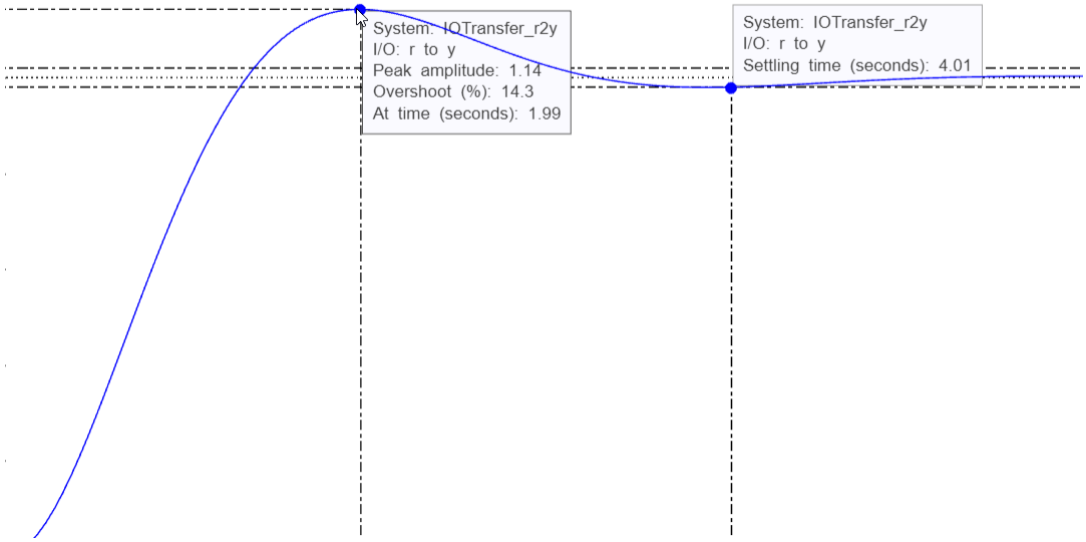
Next, we need to plot the root locus of the system after designing the Lead controller using the pole cancellation method to see whether the root locus passes through the desired points or not.

```
num3 = [k*15.25];
den3 = [1 12.7+p 12.7*p 0];
g2 = tf(num3,den3);
rlocusplot (g2);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold on
```



As it is evident, the root locus effectively passes through the desired points after designing the Lead controller.

Next, we will examine the step response of the system to see how closely we have approached the specified characteristics.



It is observed that the settling time in this method is very close to the desired state. The level of overshoot is also acceptable; however, if this overshoot can be reduced to around 15%, it would indicate a faster system response. Efforts will be made to achieve this improvement moving forward.

3-2: Lag Controller Design and Analysis

As we discussed:

$$z = \frac{\alpha}{10}$$

$$\frac{K_v old}{K_v new} = \frac{z}{p}$$

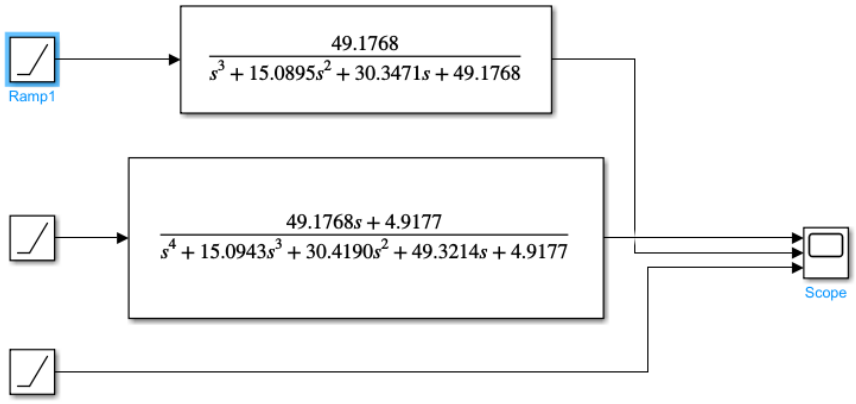
We also know that $K_v old$ is calculated using the equation below.

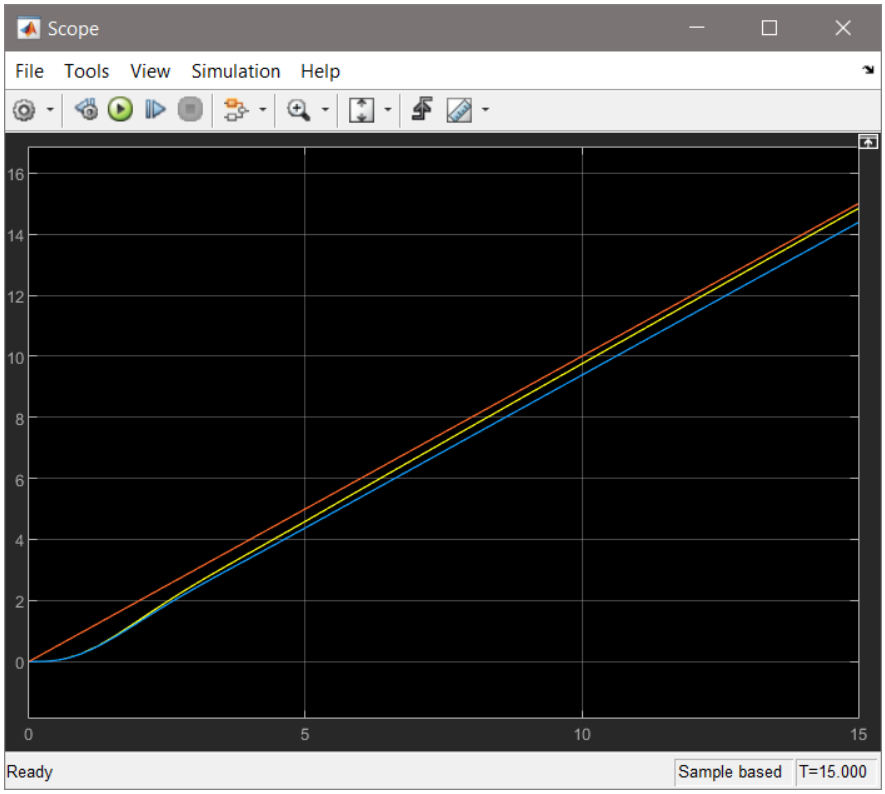
$$K_v old = \lim_{s \to 0} sG_c(s)G(s) = 1.62$$

Therefore, to calculate the zero and pole of the Lag controller, we proceed as follows:

```
K_old = (k*15.25)/(12.7*p)
K_old = 1.6205
K_new = 34;
z2 = -xpoint/10;
p2 = (z2*K_old)/K_new
p2 = 0.0048
```

Next, to observe the performance of the Lag controller, we need to examine the system's response to a ramp input.
We will again use Simulink to visualize the ramp response.





(3-1)

It is evident that after applying the Lag controller, the ramp response has a lower steady-state error.

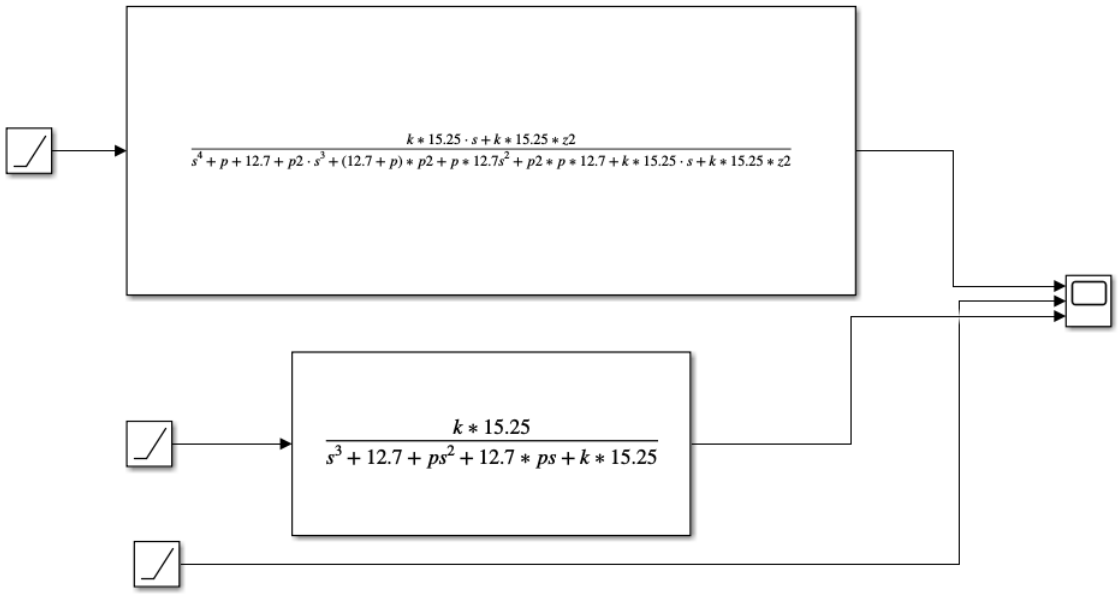
Now, we want to investigate whether this error can be further reduced by adjusting the zero and pole. Therefore, we will consider z as follows:

$$z = \frac{\alpha}{8}$$

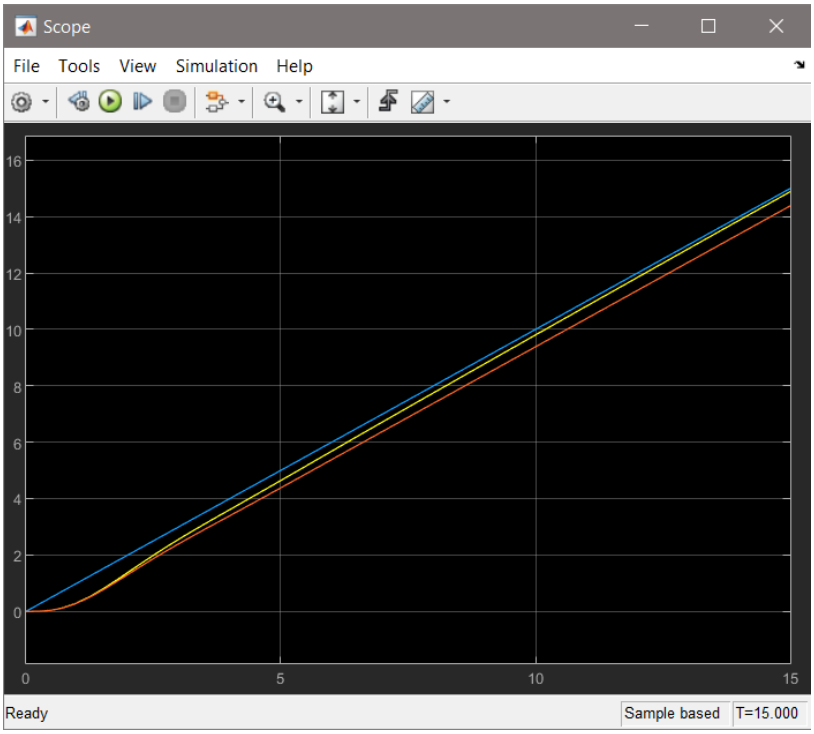
We will perform the calculations in MATLAB.

```
K_old = (k*15.25)/(12.7*p)
K_old = 1.6205
K_new = 34;
z2 = -xpoint/8
z2 = 0.1250
p2 = (z2*K_old)/K_new
p2 = 0.0060
```

Now, with these values, we will analyze the ramp response in the Simulink environment.



By capturing the ramp output from the scope, we have the following:



(3-2)

As it is evident, by adjusting the zero and pole of the Lag controller, we have achieved a much more suitable result. In fact, the steady-state error in response to the ramp input has significantly decreased.
This improvement can be clearly understood by comparing Figure (3-1) with Figure (3-2).

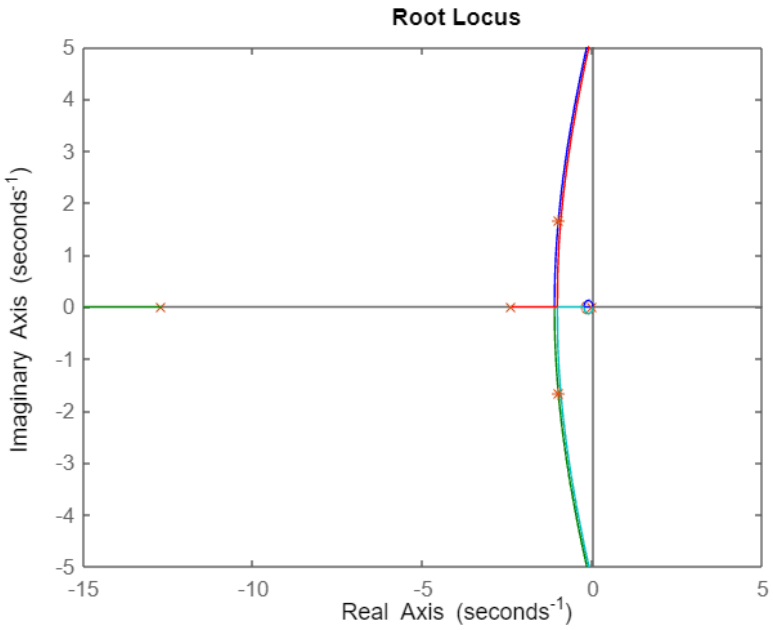
3-3: System After Lead-Lag

Finally, the transfer function of the system after designing the Lead-Lag controller will be as follows:

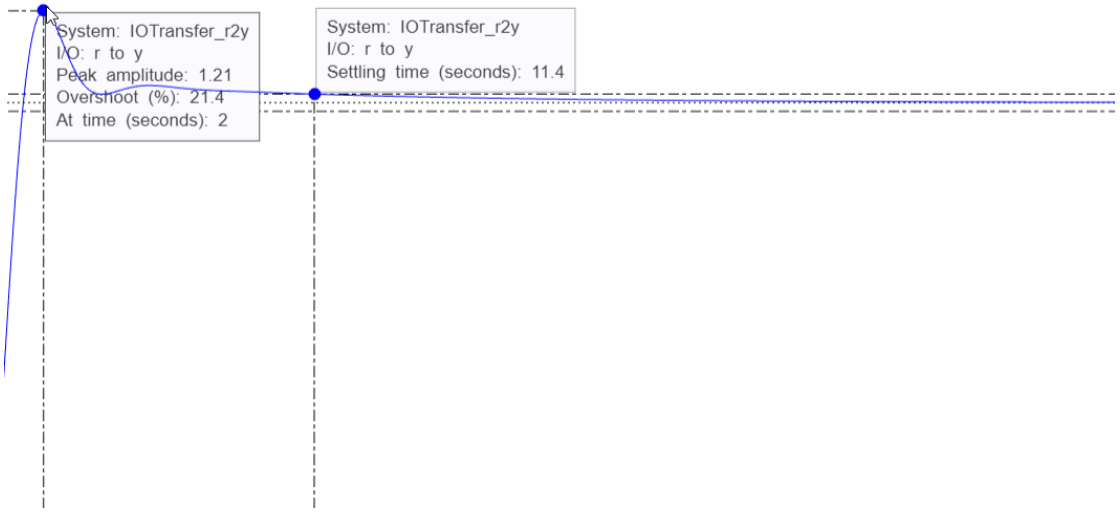
$$G(s) = \frac{3.22 \times 15.25(s + 0.125)}{s(s + 12.7)(s + 2.39)(s + 0.006)}$$

For the root locus of the system, we have:

```
num4 = [k*15.25 k*15.25*z2];
den4 = [1 p+12.7+p2 (12.7+p)*p2+p*12.7 p2*p*12.7 0];
g3 = tf(num4,den4);
h = rlocusplot (g3);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



If we examine the step response, we will have:



It is clear that we have deviated from the desired specifications.

Here, we will also plot the step response and the root locus by reducing the zero of the Controller.

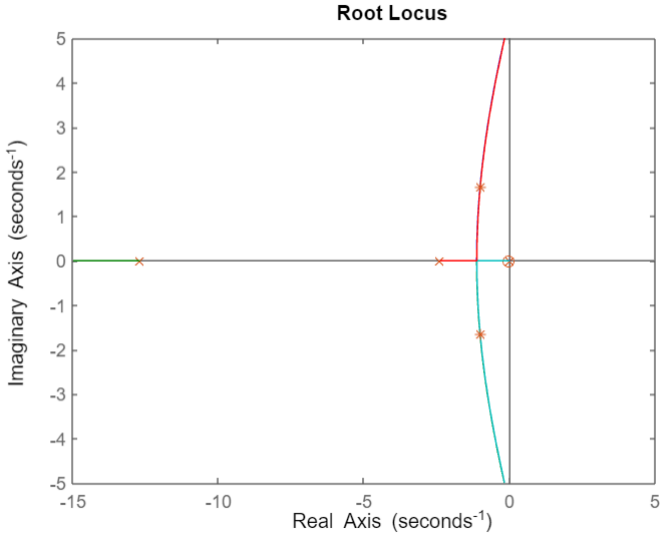
```
K_old = (k*15.25)/(12.7*p);
K_new = 34;
z2 = 0.1/10
z2 = 0.0100
p2 = (z2*K_old)/K_new
p2 = 4.7661e-04
```

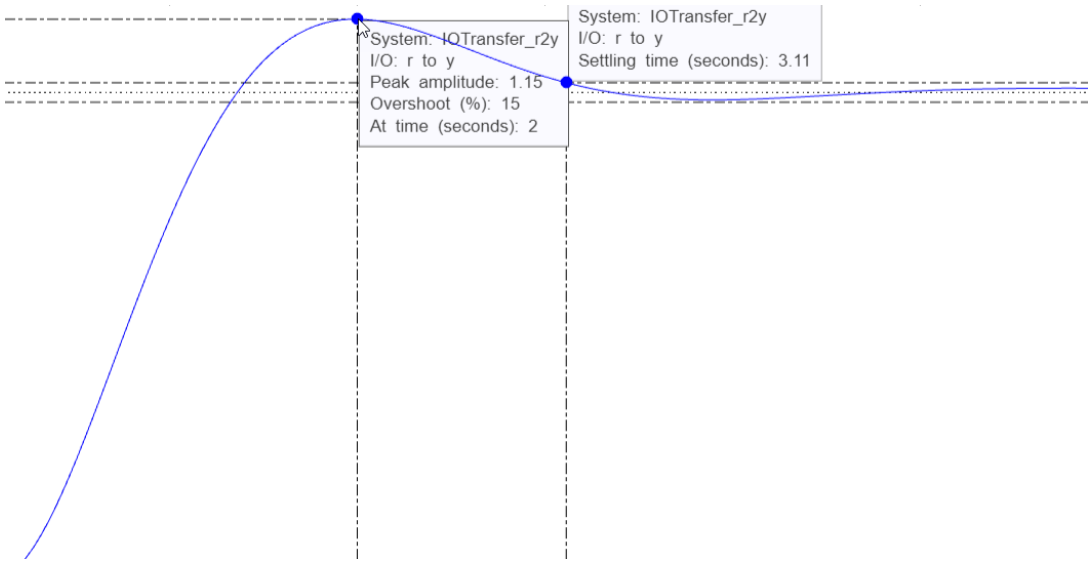
```
num4 = [k*15.25 k*15.25*z2];
den4 = [1 p+12.7+p2 (12.7+p)*p2+p*12.7 p2*p*12.7 0];
g3 = tf(num4,den4);
h = rlocusplot (g3);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```

As a result:

$$G(s) = \frac{3.22 \times 15.25(s + 0.01)}{s(s + 12.7)(s + 2.39)(s + 0.00047)}$$

Next, we will plot the root locus and the step response of the system to evaluate the characteristics.





It is evident that both the root locus has become more precise, and the step response has significantly improved.

Chapter 4:

Lead-Lag Controller Design with Zero and Pole Adjustment

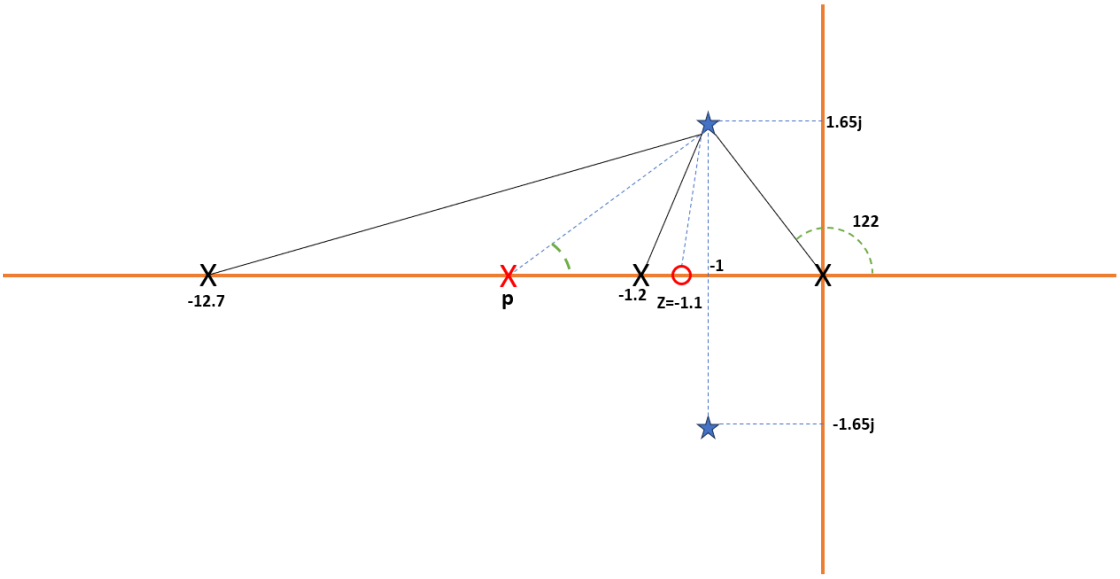
4-1: Introduction

In this section, we aim to achieve the optimal solution by adjusting the zero of the Lead Controller, which will consequently shift its pole.

To this end, we will first place the zero at the point $z = 1.1$ and then at $z = 1.25$ to analyze the step response of the system after applying these controllers.

4-2: Zero Placement at Point 1.1 (Z=1.1)

First, we will specify the desired location of the zero and then attempt to determine the pole and gain of the controller.



Next, to calculate the angle, the value of p (pole), and the gain of the controller, we will proceed as follows:

$$-\theta_p = 180 + 122 + tg^{-1}\frac{1.65}{11.7} + tg^{-1}\frac{1.65}{0.2} - tg^{-1}\frac{1.65}{0.1}$$

$$\rightarrow \theta_p \simeq 53$$

```
z = 1.1;
teta_p = 53;
x = ypoint/(tan((teta_p*pi)/180));
p = x + 1
p = 2.2479
```

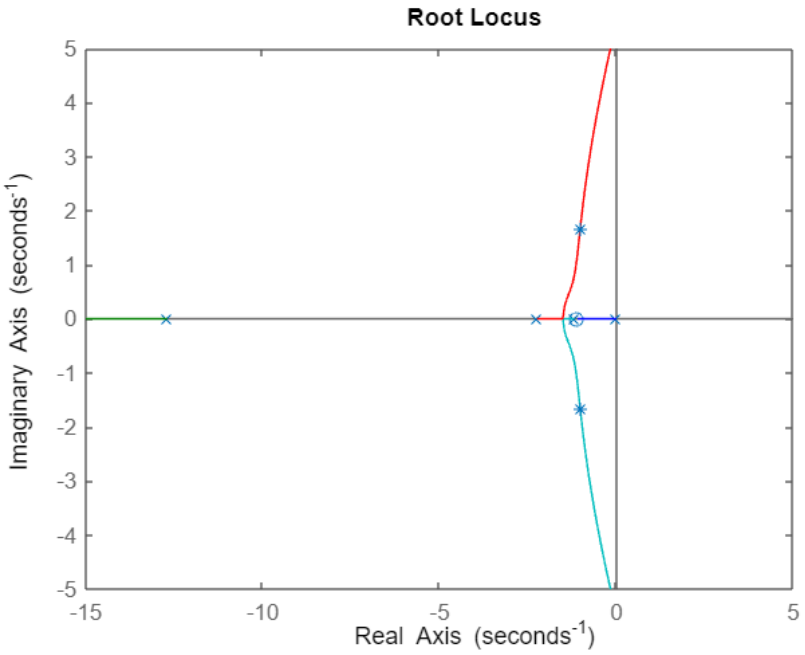
```
s = -1 + 1.65j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2)*(s+12.7);
k = abs(den2/num2)
k = 3.1095
```

As a result, the transfer function of the system after designing the Lead controller with $z = 1.1$ becomes as follows:

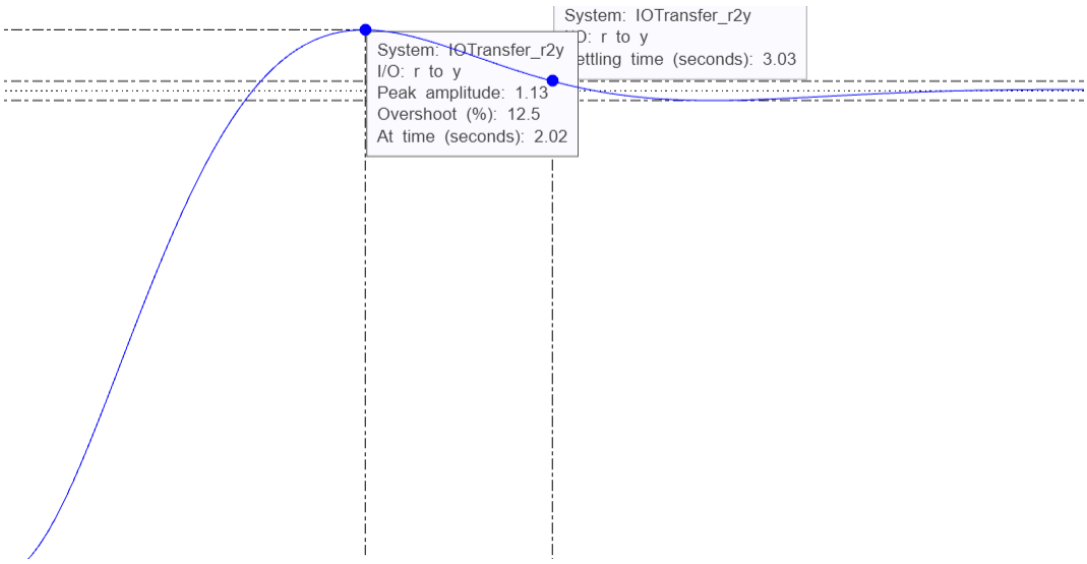
$$G_c(s)G(s) = \frac{3.1 \times 15.25(s+1.1)}{s(s+12.7)(s+1.2)(s+2.24)}$$

Now, if we plot the root locus and the step response for a system with this transfer function, we will have:

```
num3 = [k*15.25 k*15.25*z];
den3 = [1 13.9+p p*13.9+15.24 15.24*p 0];
g2 = tf(num3,den3);
rlocusplot (g2);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



Next, we will plot the step response of the system using **sisotool**.



It can be observed that by moving the zero towards the pole located at $p = -1.2$, the oscillation and overshoot of the step response have decreased, and the settling time has also become less than the desired value.

The advantages and disadvantages of this response will be discussed at the end of the report, along with the other controllers.

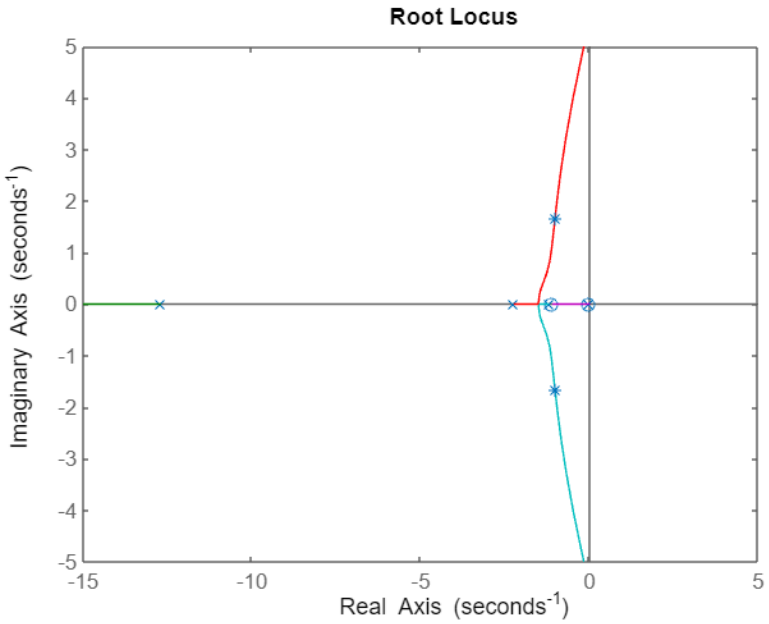
Now, if we design the Lag controller in an optimal manner (by adjusting the zero of the Lag Controller to $Z = \frac{0.1}{10}$), we will have:

```
K_old = 1.523;
K_new = 34;
z2 = 0.1/10
z2 = 0.0100
p2 = (z2*K_old)/K_new
p2 = 4.4794e-04
```

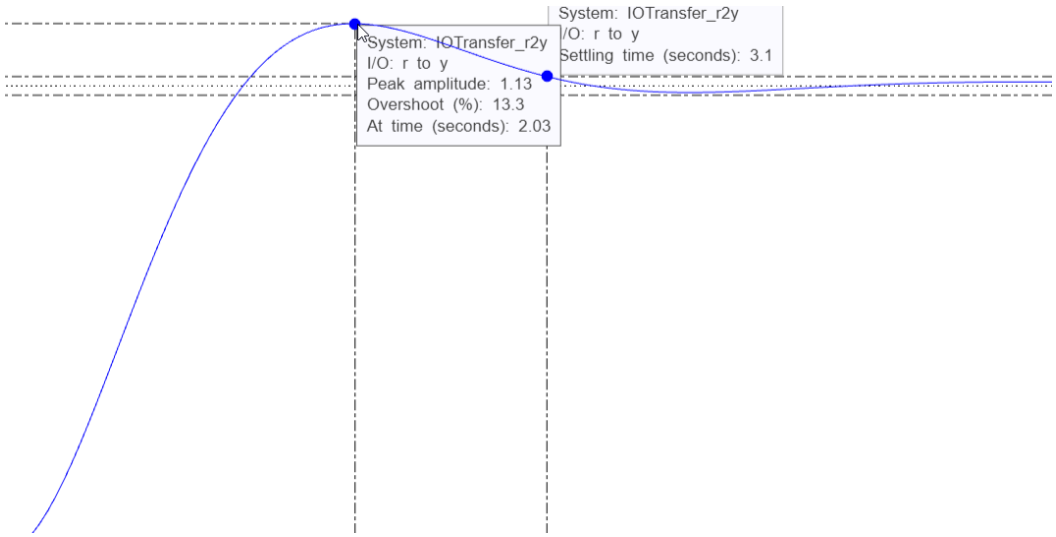
The transfer function after applying the Lead-Lag controller will be as follows:

$$G(s) = \frac{3.1 \times 15.25(s+1.1)(s+0.01)}{s(s+12.7)(s+1.2)(s+2.24)(s+0.00045)}$$

```
num4 = [47.4194 52.6355 0.5216];
den4 = [1 16.1483 46.4925 34.2779 0.015 0];
g3 = tf(num4,den4);
h = rlocusplot (g3);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```

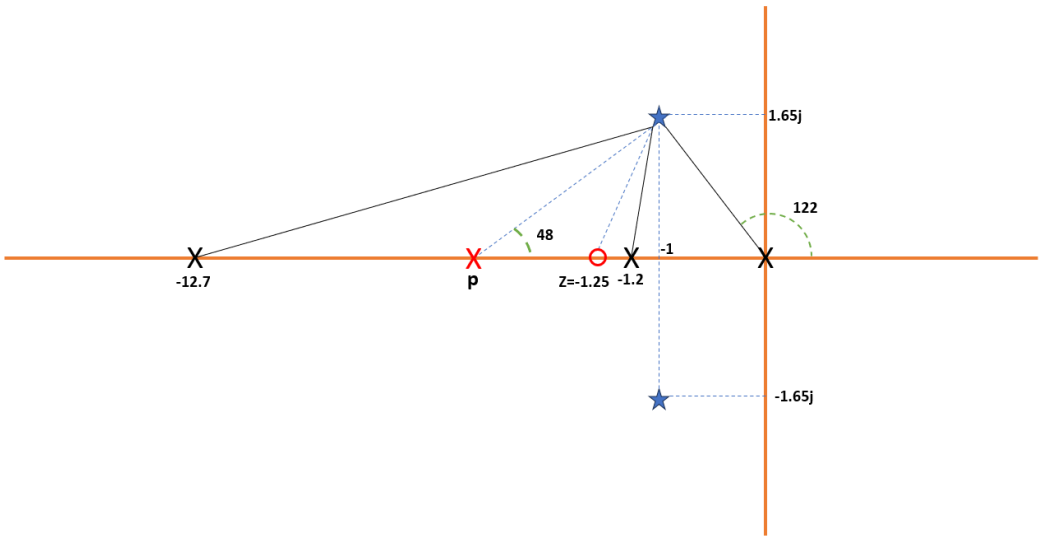


For the step response of the above system, we have:



4-3: Zero Placement at Point 1.25 (Z=1.25)

First, we will specify the desired location of the zero and then attempt to determine the pole and gain of the controller.



Next, to calculate the angle, the value of *p* (the pole), and the gain of the controller, we will proceed as follows:

$$-\theta_p = 180 + 122 + tg^{-1}\frac{1.65}{11.7} + tg^{-1}\frac{1.65}{0.2} - tg^{-1}\frac{1.65}{0.25}$$

$$\rightarrow \theta_p \simeq 48$$

```
z = 1.25;
teta_p = 48;
x = ypoint/(tan((teta_p*pi)/180));
p = x + 1
p = 2.4911
```

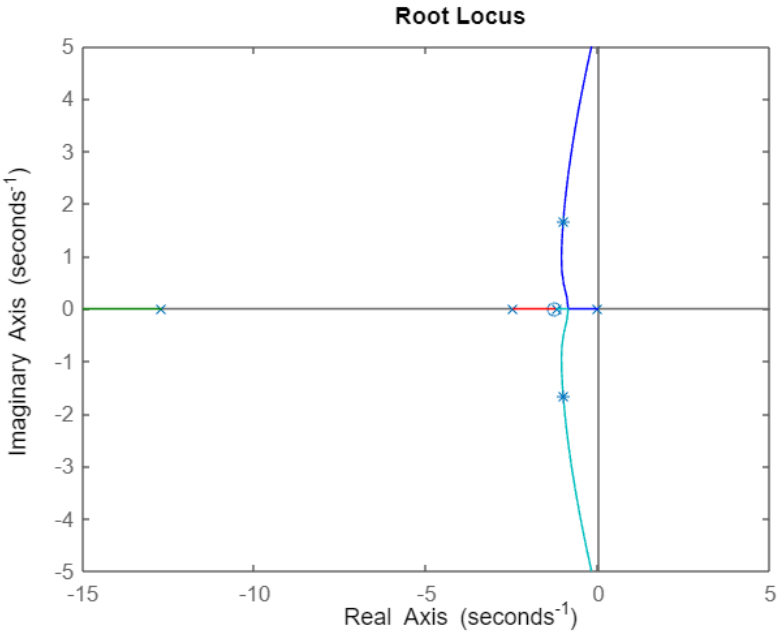
```
s = -1 + 1.65j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2)*(s+12.7);
k = abs(den2/num2)
k = 3.3110
```

As a result, the transfer function of the system after designing the Lead controller with $z = 1.25$ becomes as follows:

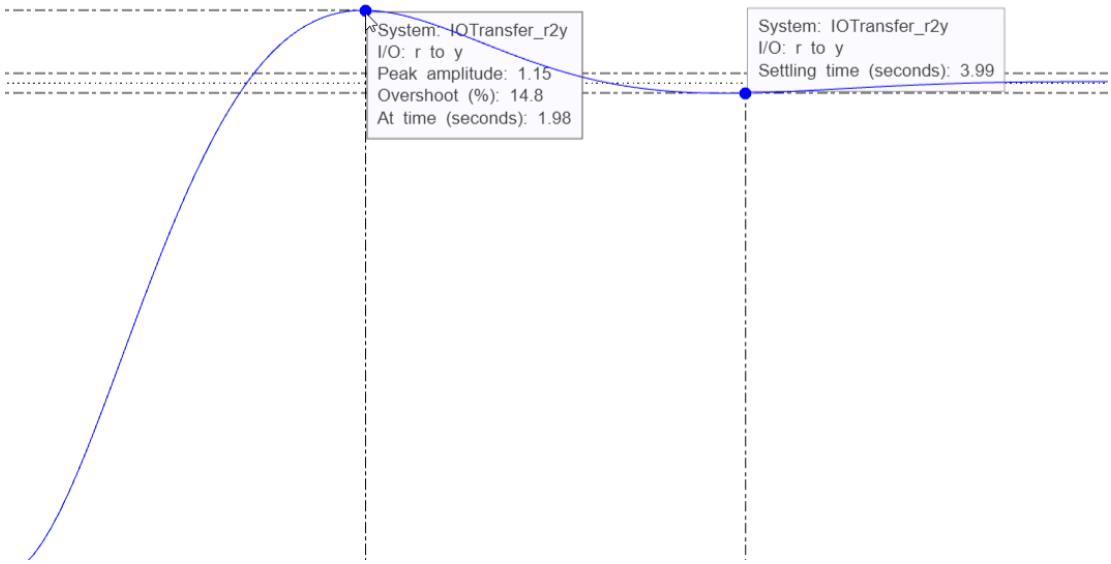
$$G_c(s)G(s) = \frac{3.31 \times 15.25(s+1.25)}{s(s+12.7)(s+1.2)(s+2.49)}$$

Now, if we plot the root locus and the step response for the system with this transfer function, we will have:

```
num3 = [k*15.25 k*15.25*z];
den3 = [1 13.9+p p*13.9+15.24 15.24*p 0];
g2 = tf(num3,den3);
rlocusplot (g2);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```



Next, we will plot the step response of the system using **sisotool**.



What appears to be the case is that, in this situation, we have achieved the closest result to the desired specifications. Both the system's settling time is very close to 4 seconds, and the overshoot has reached an appropriate level.

Now, if we design the Lag controller in an optimal manner (by adjusting the zero of the Lag Controller to $Z = \frac{0.1}{10}$) as we previously analyzed, we will have:

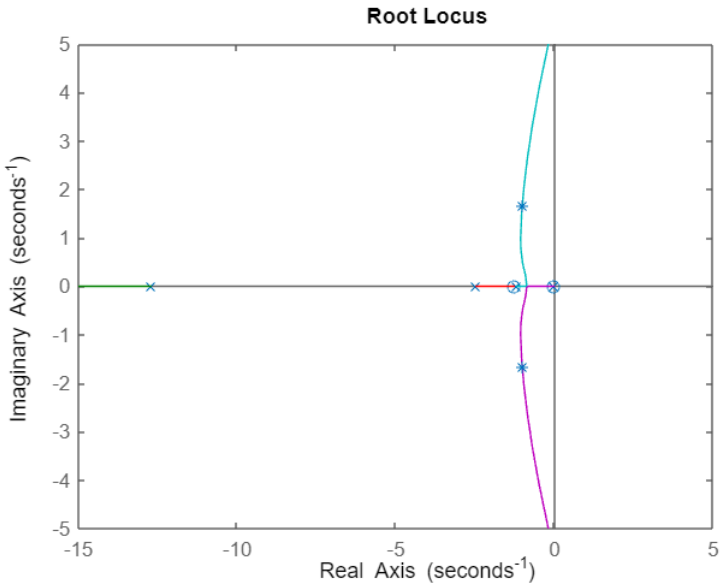
```
K_old = 1.662;
K_new = 34;
z2 = 0.1/10
z2 = 0.0100
p2 = (z2*K_old)/K_new
p2 = 4.8882e-04
```

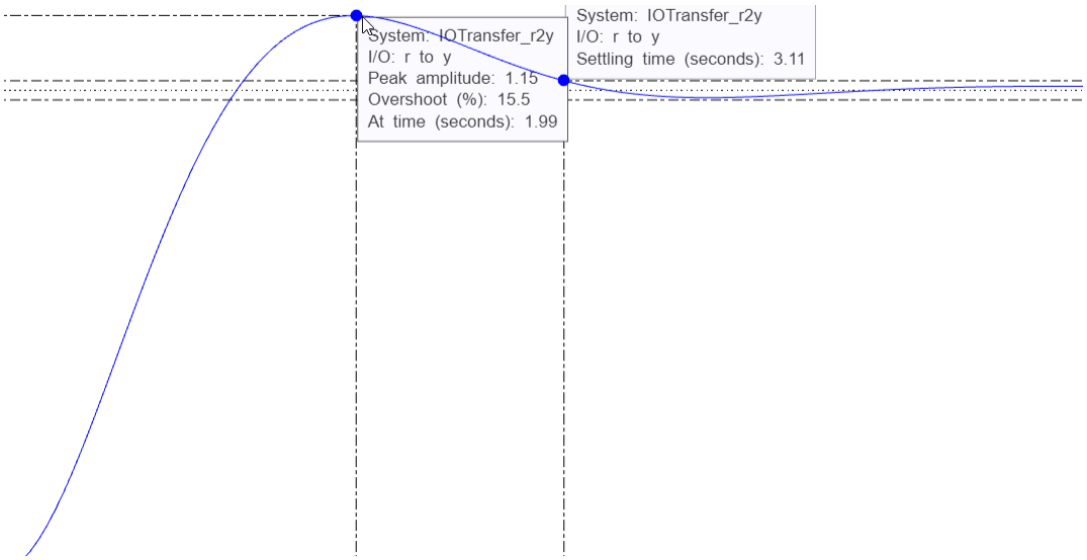
The transfer function after applying the Lead-Lag controller will be as follows:

$$G(s) = \frac{3.31 \times 15.25(s+1.25)(s+0.01)}{s(s+12.7)(s+1.2)(s+2.49)(s+0.00049)}$$

Next, we need to plot the root locus and the step response of the system.

```
num4 = [50.4933 63.6215 0.6311];
den4 = [1 16.3916 49.8736 37.988 0.0186 0];
g3 = tf(num4,den4);
h = rlocusplot (g3);
xlim([-15, 5]);
ylim([-5, 5]);
hold on
points = [xpoint + ypoint*1i, xpoint - ypoint*1i];
plot(points, '*');
hold off
```


Root Locus
Imaginary Axis (seconds-1)
Real Axis (seconds-1)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-15
-10
-5
0
5


System: IOTransfer_r2y
I/O: r to y
Peak amplitude: 1.15
Overshoot (%): 15.5
At time (seconds): 1.99
System: IOTransfer_r2y
I/O: r to y
Settling time (seconds): 3.11

Chapter 5:
Comparison of Designed Controllers

5-1- Comparison of Lead Controllers

The important characteristics after designing four types of Lead controllers are summarized in the table below.

Method	Overshoot (%)	Settling Time (s)
Graphical Method	16.3	4.11
Pole-Zero Cancellation	14.3	4.01
Z=1.25	14.8	3.99
Z=1.1	12.5	3.03

Clearly, the graphical method performs worse here compared to the other methods, as it has both a higher overshoot and a slower settling time.

On the other hand, comparing the pole-zero cancellation method and placing the zero at $Z = 1.25$, we find that there is not much difference in settling time. Therefore, we should examine their overshoot. The overshoot for $Z = 1.25$ is much closer to the desired specification. Additionally, a lower overshoot in the pole-zero cancellation method implies a slower rise time. Thus, we can conclude that the system performs better with $Z = 1.25$ than with the pole-zero cancellation method.

Regarding the last two methods, it is evident that with $Z = 1.25$, our specifications are closest to the desired values. However, with $Z = 1.1$, the system exhibits better performance. It shows both lower overshoot and shorter settling time, resulting in a faster system response.

In summary, with $Z = 1.25$, we achieve the closest configuration to the problem specifications. Conversely, $Z = 1.1$ represents the optimal design of the Lead controller.

5-2: Comparison of Lead-Lag Controllers

The key characteristics after designing four types of Lead-Lag controllers are summarized in the table below. It is important to note that in all four cases, the optimal controller condition $z = \frac{0.1}{10}$ has been considered. In fact, all cases are compared under the same conditions.

Method	Overshoot (%)	Settling Time (s)
Graphical Method	17	3.1
Pole-Zero Cancellation	15	3.11
Z=1.25	15.5	3.11
Z=1.1	13.3	3.1

As is evident, there is not much difference in settling time among the systems after applying the Lead-Lag controllers. Therefore, the characteristic we will use for comparison is overshoot. As mentioned, the key point about overshoot is that reducing it leads to a decrease in the speed of the system. Therefore, when the issue does not have a problem with a certain amount of overshoot, it is better to consider the maximum overshoot so that the system's speed does not decrease.

Thus, it can be said that in the case of $z = 1.1$, by reducing the overshoot, we are actually increasing the rise time of the step response, which is not desirable for us. On the other hand, having overshoot greater than the desired value is also not suitable for us as designers.

Consequently, the graphical method and $z = 1.25$ seem worse than the other methods. Given the aforementioned points, if I, as a designer, had to choose a method for the system, my choice would be the pole placement method.

Chapter 6:
Optimization and Practical Considerations

6-1: Introduction

In the controllers designed so far, particularly the Lag controllers, their practical implementation has not been thoroughly examined. It is known that if a system has very small zeros or very large poles, constructing such a controller becomes extremely challenging from a practical standpoint. In the earlier designs, especially with the Lag controllers, very small zeros were used.

Therefore, we need to find another solution to optimize the controller after applying the Lag component. To address this, we will attempt to design the Lead controller in such a way that the system characteristics fall below the desired values, so that after applying the Lag controller, these characteristics do not deviate too far from the required values.

Additionally, we will explore adjustments to the gain to achieve better system responses.

Another assumption we will consider for improving the system response is:
We know that poles far from the $j\omega$-axis are less influential on the system. Thus, in the following steps, we will disregard the pole located at 12.7.

As a result, the system's transfer function will be as follows:

$$G(s) = \frac{15.25}{s(s+1.2)}$$

6-2: Optimal Graphical Lead-Lag Design

As mentioned earlier, here we will first try to design the Lead controller with characteristics below the desired values so that after applying the Lag controller, the characteristics do not deviate significantly from the desired values.

In one case, we consider the settling time to be 2 seconds and the overshoot to be 12%. In another case, we set the settling time to 3 seconds and the overshoot to 13%, and we will compare the results.

6-2-1: 2-second Settling Time and 12% Overshoot

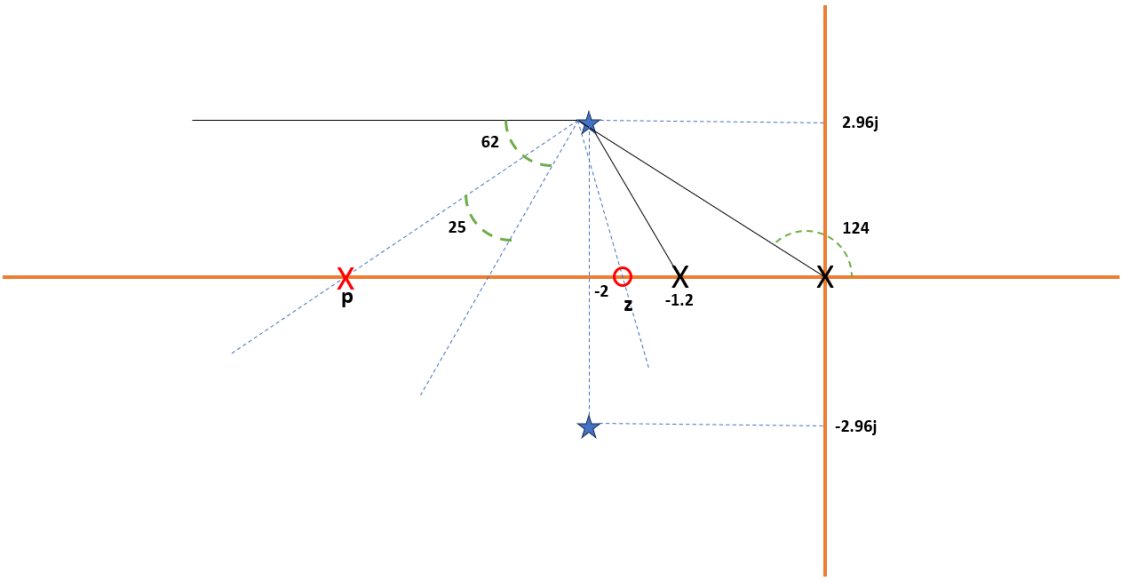
First, we set the desired system characteristics. Let's start by determining the key parameters.

```
Mp = 0.12;
zeta = -log(Mp)/(sqrt(pi^2 + log(Mp)^2))
zeta = 0.5594
```

```
ts = 2;
Wn = 4/(ts*zeta)
Wn = 3.5752
```

```
beta = acos(zeta);
beta = rad2deg(beta);
xpoint = -zeta*Wn
xpoint = -2
tan_beta = tan((beta*pi)/180);
ypoint = tan_beta*zeta*Wn
ypoint = 2.9634
```

Now, to find the zero and pole using the graphical method in the Lead controller design, we plot the following figure.



$$\theta_z - \theta_p = \varphi = 180 + 124 + 180 - tg^{-1}\frac{2.96}{0.8}$$

$$\rightarrow \varphi \simeq 50 \rightarrow \frac{\varphi}{2} = 25$$

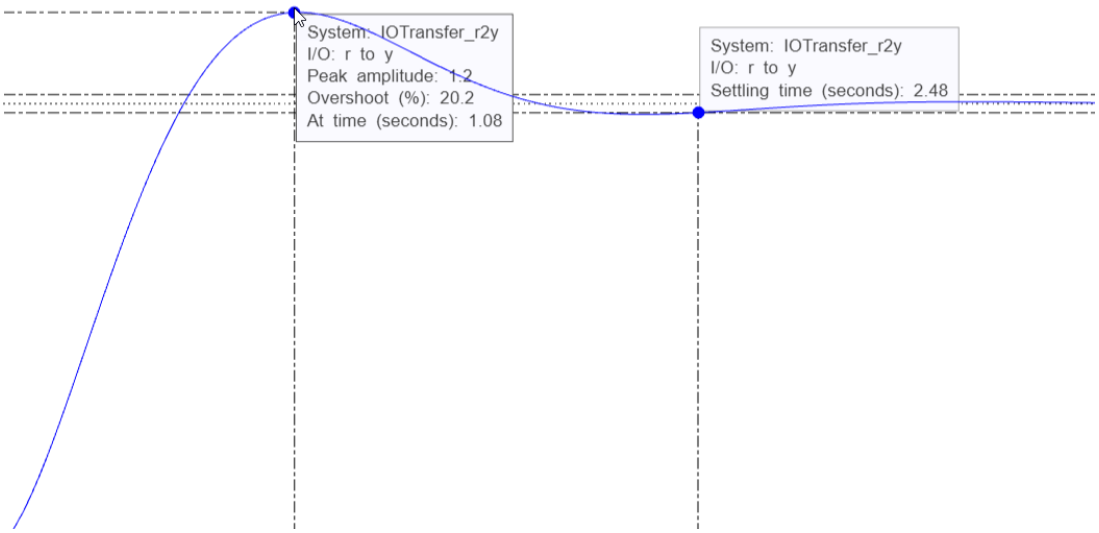
$$\theta_p = 62 - 25 = 37 \rightarrow \theta_z = \varphi + 50 = 87$$

As a result, we calculate the zero and pole of the controller and then plot the system's response.

```
teta_p = 37;
teta_z = 87;
x = ypoint/(tan((44.5*pi)/180));
p = x + 1
p = 4.0156
y = ypoint/(tan((77.5*pi)/180));
z = y + 1
z = 1.6570
```

```
s = -2 + 2.69j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2);
k = abs(den2/num2)
k = 0.7646
```

```
num3 = [k*15.25 k*15.25*z];
den3 = [1 1.2+p 1.2*p 0];
g2 = tf(num3,den3);
sisotool(g2);
```



Now, we apply the Lag controller in its standard definition. Additionally, we place the zero at the smallest permissible value, which is $z = \frac{\alpha}{12}$.

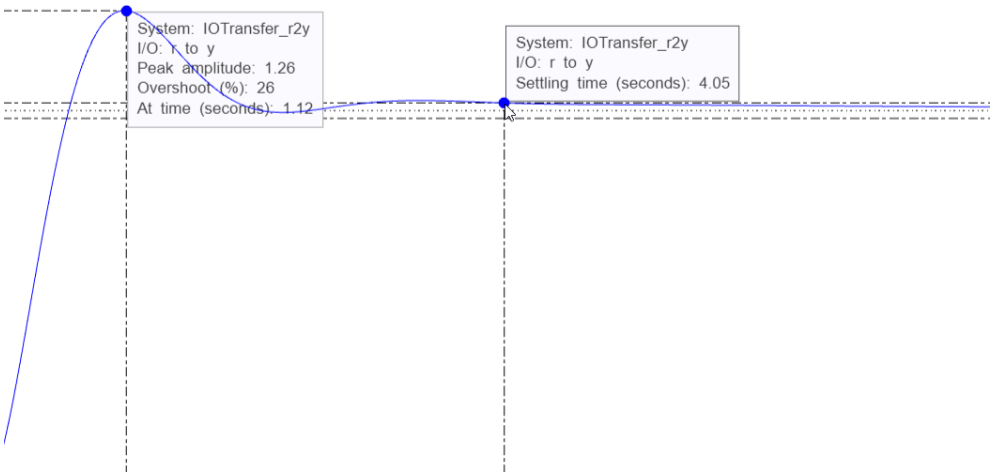
```
K_old = (k*15.25*z)/(1.2*p);
K_new = 34;
z2 = -xpoint/12
z2 = 0.1667
p2 = (z2*K_old)/K_new
p2 = 0.0197
```

```
num4 = [k*15.25 (k*15.25)*(z+z2) k*15.25*z*z2];
den4 = [1 1.2+p+p2 (1.2*(p+p2))+(p*p2) 1.2*p*p2 0];
g3 = tf(num4,den4);
sisotool(g3)
```

The final transfer function of the system, after designing the Lead-Lag controller, will be as follows:

$$G(s) = \frac{15.25(s+1.65)(s+0.16)}{s(s+1.2)(s+4.01)(s+0.02)}$$

If we plot the step response after designing the Lead-Lag controller, we will obtain the following results:



It is observed that by refining the design of the Lead controller, we were able to improve the system's step response after applying the Lead-Lag controller to some extent. The settling time is very close to the desired specification, but there is still a slight amount of overshoot that exceeds the allowable limits.

Next, we will analyze another set of conditions to further optimize the system's performance.

6-2-2: 3-second Settling Time and 13% Overshoot

First, we define the system specifications.

```
Mp = 0.13;
zeta = -log(Mp)/(sqrt(pi^2 + log(Mp)^2))
zeta = 0.5446
```

```
ts = 3;
Wn = 4/(ts*zeta)
Wn = 2.4481
```

```
beta = acos(zeta);
beta = rad2deg(beta);
xpoint = -zeta*Wn
xpoint = -1.3333
tan_beta = tan((beta*pi)/180);
ypoint = tan_beta*zeta*Wn
ypoint = 2.0531
```

Now, to find the zero and pole using the graphical method in the Lead design, we draw the following figure.

```
teta_p = 43;
teta_z = 83;
x = ypoint/(tan((44.5*pi)/180));
p = x + 1
p = 3.0893
y = ypoint/(tan((77.5*pi)/180));
z = y + 1
z = 1.4552
```

```
s = -1.33 + 2.05j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2);
k = abs(den2/num2)
k = 0.4329
```

```
num3 = [k*15.25 k*15.25*z];
den3 = [1 1.2+p 1.2*p 0];
g2 = tf(num3,den3);
sisotool(g2)
```

Then, we design the Lag controller as follows:

```
K_old = (k*15.25*z)/(1.2*p);
K_new = 34;
z2 = -xpoint/12
z2 = 0.1111
p2 = (z2*K_old)/K_new
p2 = 0.0085
```

```
num4 = [k*15.25 (k*15.25)*(z+z2) k*15.25*z*z2];
den4 = [1 1.2+p+p2 (1.2*(p+p2))+(p*p2) 1.2*p*p2 0];
g3 = tf(num4,den4);
```

If we plot the step response after designing the Lead-Lag controller, we will have:

In this case, it can be observed that the overshoot has improved, while the settling time has deviated from the desired value.

6-3: Optimal Lead-Lag Design with Pole Cancellation

This time, for the same settling time of 2 seconds and an overshoot of 12%, we will design the Lead controller using the pole removal method.

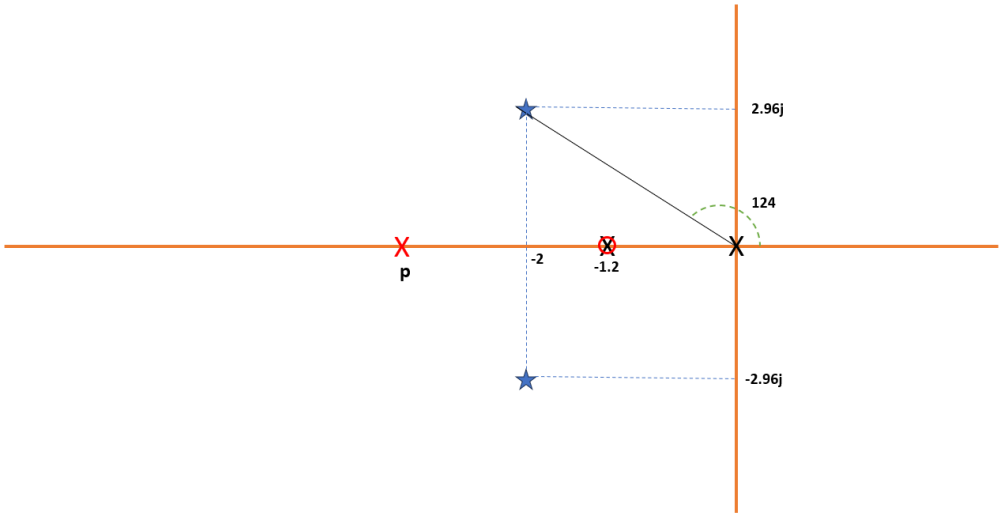
We have:

```
Mp = 0.12;
zeta = -log(Mp)/(sqrt(pi^2 + log(Mp)^2))
zeta = 0.5594
```

```
ts = 2;
Wn = 4/(ts*zeta)
Wn = 3.5752
```

```
beta = acos(zeta);
beta = rad2deg(beta);
xpoint = -zeta*Wn
xpoint = -2
tan_beta = tan((beta*pi)/180);
ypoint = tan_beta*zeta*Wn
ypoint = 2.9634
```

We want to design the controller using the pole removal method, so the zero of the Lead Controller should be placed at point 1.2. To find the pole of the Lead controller, we refer to the following shapes and relationships.



$$-\theta_p = 180 + 124 \rightarrow \theta_p = 56$$

Now we will calculate the zeros and poles of the Lead controller.

```
z = 1.2;
teta_p = 54;
x = ypoint/(tan((teta_p*pi)/180));
p = x + 1
p = 3.1530
```

```
s = -2 + 2.96j ;
num2 = 15.25*(s+z);
den2 = (s+p)*s*(s+1.2);
k = abs(den2/num2)
k = 0.7441
```

Now we will apply a Lag controller on this Lead controller, and its zero and pole will be calculated as follows.

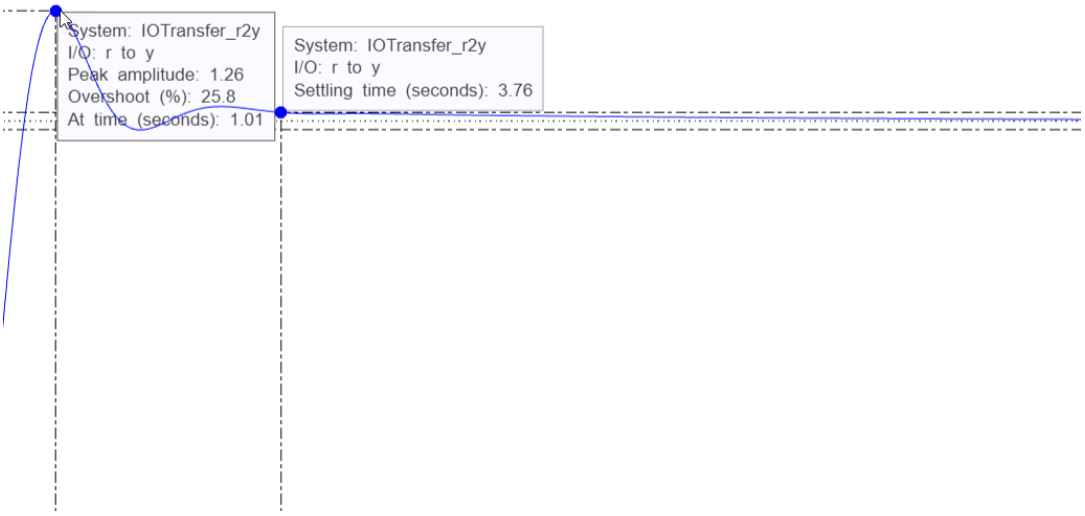
```
K_old = (15.25)/(p);
K_new = 34;
z2 = 2/12
z2 = 0.1667
p2 = (z2*K_old)/K_new
p2 = 0.0237
```

As a result, the overall transfer function of the system after designing the Lead-Lag controller is as follows.

$$G(s) = \frac{15.25(s + 0.16)}{s(s + 3.15)(s + 0.02)}$$

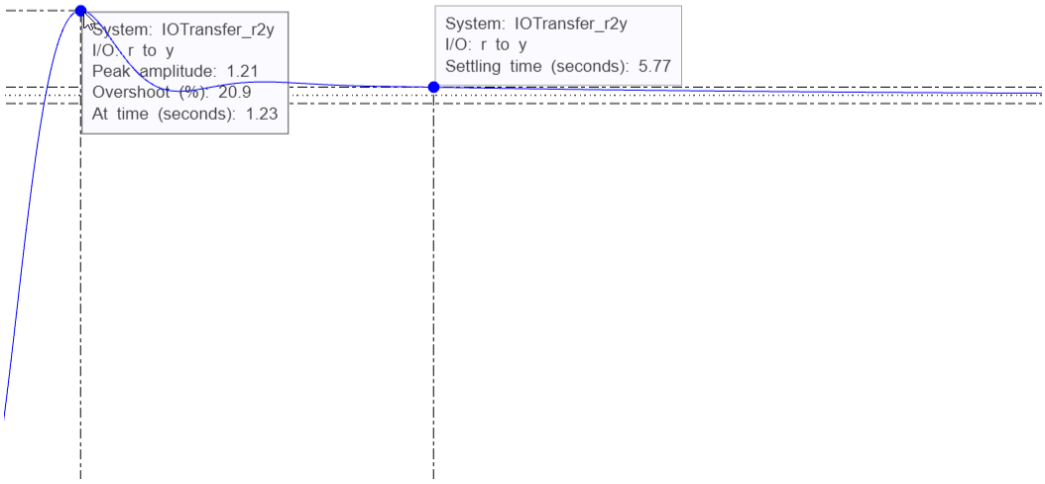
Now, if we plot the step response of the above transfer function, we will have:

Now we want to examine how changing the gain of the transfer function affects the final step response. By increasing the value of *k*, the step response will be as follows:



It can be seen that by increasing *k*, the settling time decreases, but the overshoot increases.

Now, we will reduce the gain to further examine the system's behavior and find the optimal response.



Chapter 7:
Conclusion

7-1: Final Lead-Lag Comparison (Practical)

Method	Overshoot (%)	Settling Time (s)
Graphical Method (2-12)	26	4.05
Graphical Method (3-13)	21.3	6.34
Pole Cancellation	24.5	4.06
Pole Removal with Gain Increase	25.8	3.76
Pole Removal with Gain Decrease	20.9	5.77

If we compare the second and fifth methods, it is clear that the fifth method is more suitable in terms of both settling time and overshoot.

There is no difference in settling time between the first and third methods, but the overshoot of the third method is less. Therefore, the third method is better between these two.

Thus, the final comparison is between the three last methods. Which response is better entirely depends on the system. Whether the system wants to reach the response in the least time possible or to have less error. In any case, if less error is desired, the last method is the most appropriate. However, if the speed of reaching the response is more important, the fourth method can be used.

7-2: Summary of Project Performance

Initially, we considered the desired points requested by the problem as the target points. We then attempted to create a Lead controller to bring the root locus of the system to these target points, while also ensuring that the settling time and overshoot were close to the problem's specified values. This was done using four different methods.

After that, to reduce the steady-state error of the system to a ramp input, a Lag controller was used. We observed that applying the Lag controller decreased the steady-state error to the ramp input; however, this application caused changes in the step response, resulting in a deviation from the desired criteria.

To address this issue, we initially pursued the idea of reducing the zero of the Lag controller, which indeed provided a very suitable step response on paper. However, the problem was that it was impractical to construct such a controller in reality.

Therefore, another approach was employed: to consider the criteria more stringently from the beginning when designing the Lead controller so that after applying the Lag controller, the step response would not deviate significantly from the problem's specifications.