Department of Engineering
MECH304 - Dynamic Control

Project Part 1 : Designing a PD Controller

May 15, 2022

Deniz Erdogan
Baris Aslan Turan
Defne Korkmaz

Supervisor: Prof.Cagatay Basdogan

Contents

1 Introduction

In this project, we are proposing a design of a PD controller to stabilize a ball on a tilting grooved system. The goal is to keep the ball in the vicinity of the origin that is around the shaft of the motor that controls the system. A sample illustration of this system can be seen from Figure 1. In order to accomplish this task, firstly, the transfer function of the plant is derived using Newton's law and transferred into frequency domain by Laplace transform.

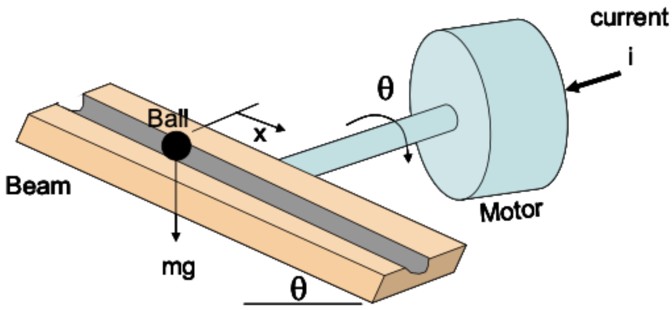
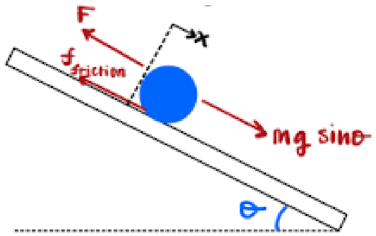


Figure 1: Illustration of the system

Following this, using tools like the *Root Locus method* or the sisotool() method of Matlab, the necessary PD parameters are derived in accordance with the design requirements that make the system not only stable but also compliant with various constraint such as using an upper bound for settling time or percent overshoot.

2 Deriving the Transfer Function of the Plant

In this section, the derivation of the plant transfer function will be investigated. To start with, we focused on the mechanical part of the system and obtained the transfer function between X and θ. Our calculations can be seen from Figure 2 below.



We are neglecting the rotational forces.

$$\Sigma F = mg\sin\theta - F - f_{fric} = 0$$

$$\Sigma F = mg\sin\theta - m\ddot{x} - \mu\dot{x} - \mu_{min}x$$

Now, we take the Laplace transform.

$$m\left(s^2X(s) - s\cancelto{0}{X(0)} - \cancelto{0}{X(0)}\right) + \mu\left(sX(s) - \cancelto{0}{X(0)}\right) + \mu_{min}X(s) - mg\sin(\Theta(s)) = 0$$

Assuming the initial conditions are zero, we can neglect the $X(0)$-like terms.

$$m(s^2X(s)) + \mu(sX(s)) + \mu_{min}X(s) - mg\sin(\Theta(s)) = 0$$

$$X(s)(ms^2 - \mu s + \mu_{min}) = mg\sin(\Theta(s))$$

Since we will be operating in smaller angles $\sin\theta \simeq \theta$

$$X(s)(ms^2 - \mu s + \mu_{min}) = mg\Theta(s)$$

$$\frac{X(s)}{\Theta(s)} = \frac{mg}{ms^2 - \mu s + \mu_{min}}$$ → TF between the angle and displacement

Figure 2: Calculations of the first part

Following these calculations, θ will be related to our current, I. After this step, two equations can be combined easily to obtain the transfer function between the current and X. Figure 3 shows the steps and derivations for these equations.

$KI(t) = T = J\ddot{\theta}$ (We assume that the torque is linearly proportional to current)

$KI(s) = T = J\theta(s)s^2$

$$\frac{\theta(s)}{I(s)} = \frac{K}{Js^2}$$

$$\frac{X(s)}{\theta(s)} = \frac{mg}{(ms^2 - \mu s + \mu_{min})}$$

$$\frac{X(s)}{I(s)} = \frac{Kmg}{Jms^4 - \mu Js^3 + \mu_{min}Js^2}$$

This is our plant transfer function

$$\xrightarrow{I(s)} \boxed{G_1(s)} \xrightarrow{\theta(s)} \boxed{G_2(s)} \xrightarrow{X(s)}$$

Figure 3: Derivation of the transfer function

Our transfer function is shown at the end of Figure 3. Plugging in the values from the project prompt, our transfer function becomes as follows:

$$T(s) = \frac{0.0981}{s^4 + 5s^3 + 20s^2} \tag{1}$$

3 Design of the PD Controller

In this part, we switch to Matlab in order to design our parameters for the PD controller with the help of necessary tools. To define our transfer function there, we first initialize our parameters. Figure 4 shows our initialization part at the beginning of our script.

```
%% Initializing Parameters
k = 1;
mu= 0.05;
mu_min = 0.2;
J = 100;
m = 0.01;
g = 9.81;
```

Figure 4: Parameter Initialization

Once these are defined, we then create our plant by defining the numerator and denominator and passing them to tf() function. The obtained transfer function will be fed into rlocus() and sisotool() methods to be manipulated further. As can be seen from Figure 5.

```
%% Defining plant
plant_num = [k*m*g];
plant_denum = [J*m, mu*J, mu_min*J, 0, 0];
plant = tf(plant_num, plant_denum)
```

Figure 5: Creating the Plant

When we see the Root Locus plot of this plant, we get the following figure. As can be seen, this system is not stable since it has 2 roots on the imaginary axis. Our aim is to introduce another *zero* to the system in order to obtain a configuration where all roots have negative real parts.

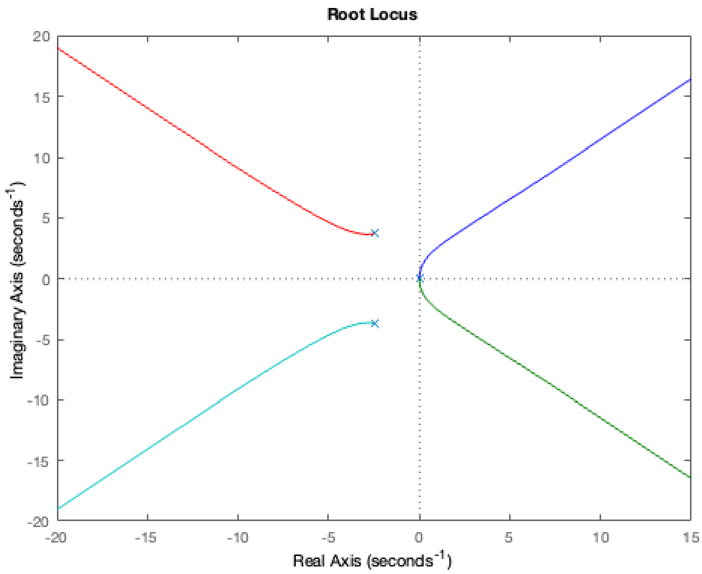


Figure 6: Root Locus Plot

Furthermore, the step response plot obtained from the sisotool() GUI confirms that the system is not stable. Figure 7 below shows this behavior.

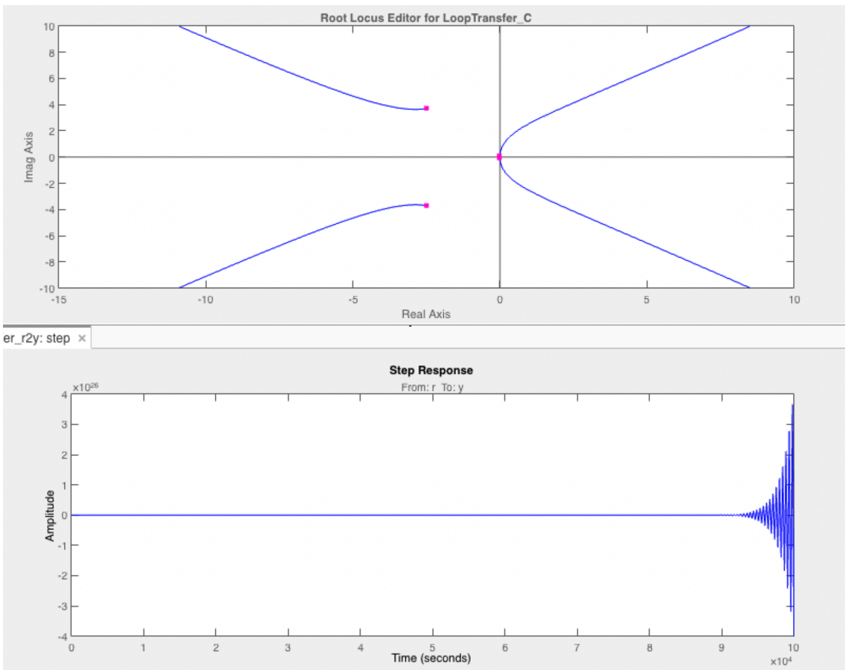


Figure 7: Sisotool Screen

The PD controller allows us to **introduce a new zero** to our system. This zero can be placed strategically in order to manipulate the *Root Locus plot* and obtain a configuration where all the root have negative real parts. There are 3 main places that our new zero can be placed. They are as follows:

- To the right of all poles
- Between the poles on the imaginary axis and the negative poles
- To the left of all poles

3.1 Zero at the Positive Side

When the zero is added to first mentioned location, this system does not benefit and continues to be unstable. A sample plot that shows this behavior is shown in Figure 8 below.

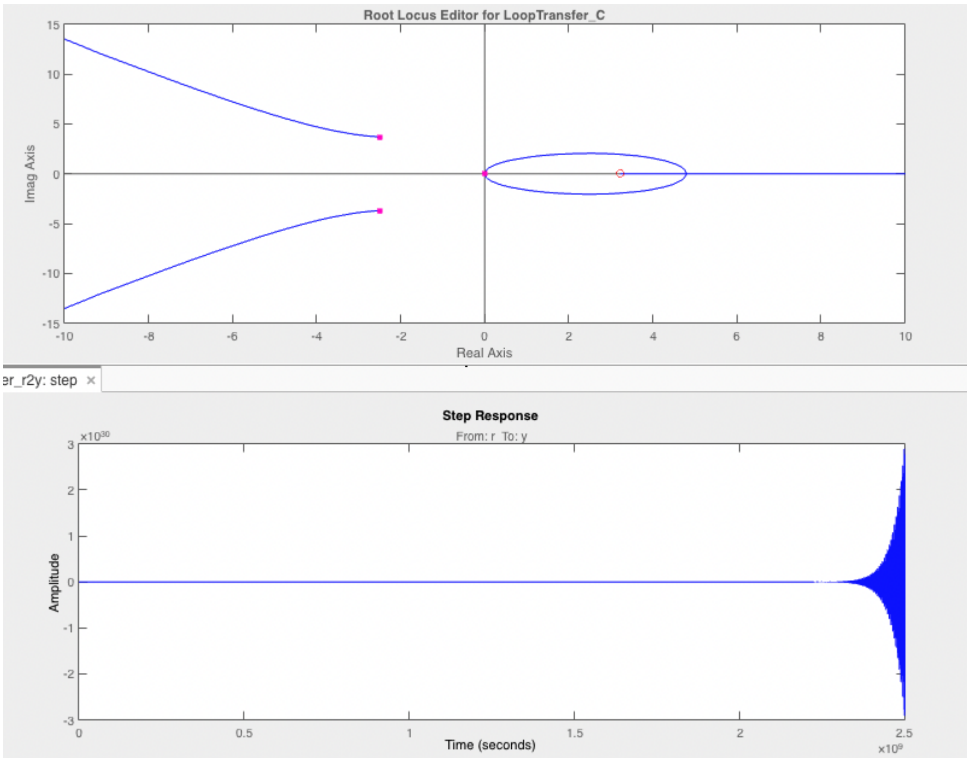


Figure 8: Zero added to the positive real part

3.2 Zero Between to the Pole Pairs

The second placement option for our zero actually allows a stable system. When the zero is placed between the two pole pairs, we obtain the option to select our K_d and K_p values to stabilize the system. A figure showing this behavior is given below on Figure 10. Although there is significant overshoot, this configuration can be stable and the design parameters can and will be further optimized throughout this report.

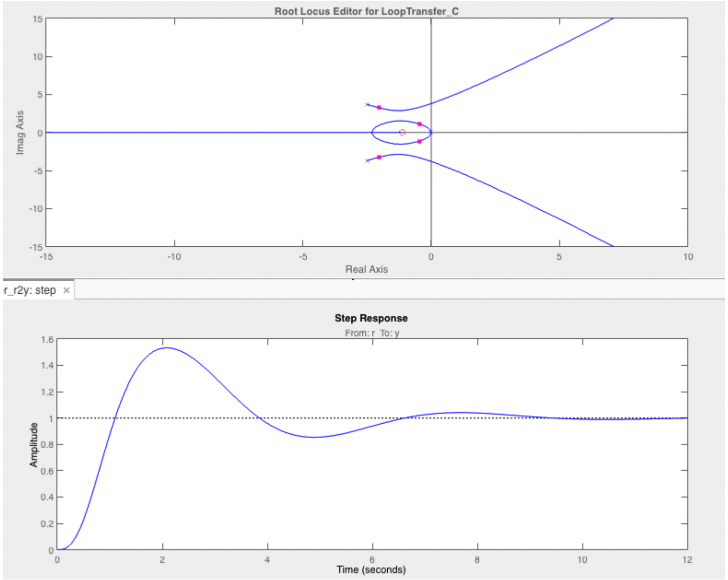


Figure 9: Zero added between the two pairs

3.3 Zero to the Left of Poles

This placement condition was first thought to not allow any stable configurations. However, it is then realized that there exists a small region to the left of the negative poles where the system in fact, can be stable.

A sample configuration for the unstable case is shown below.

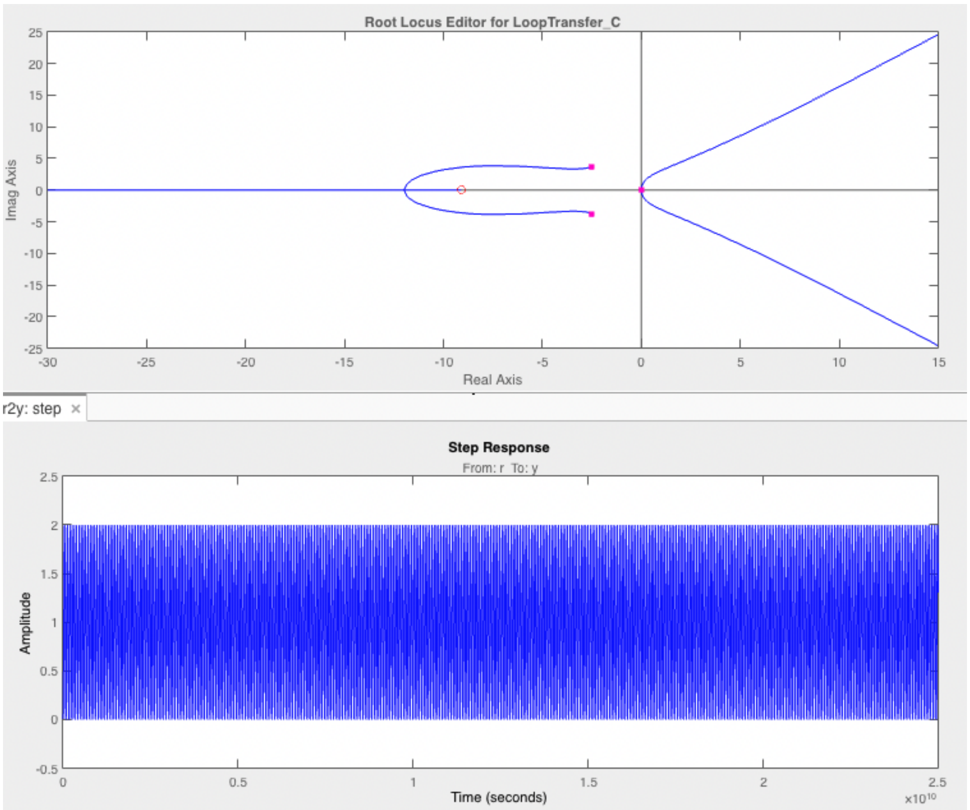


Figure 10: Zero added to the left the two pairs

3.4 Optimizing the Parameters

3.4.1 Smaller Overshoot

We continued by optimizing different design parameters individually. We also used the PID tuning tools to help us with this task. Our first optimization was keeping the overshoot at the minimum. This was observed when the zero value was close to the imaginary axis. This system is shown on the Figure 11 below.

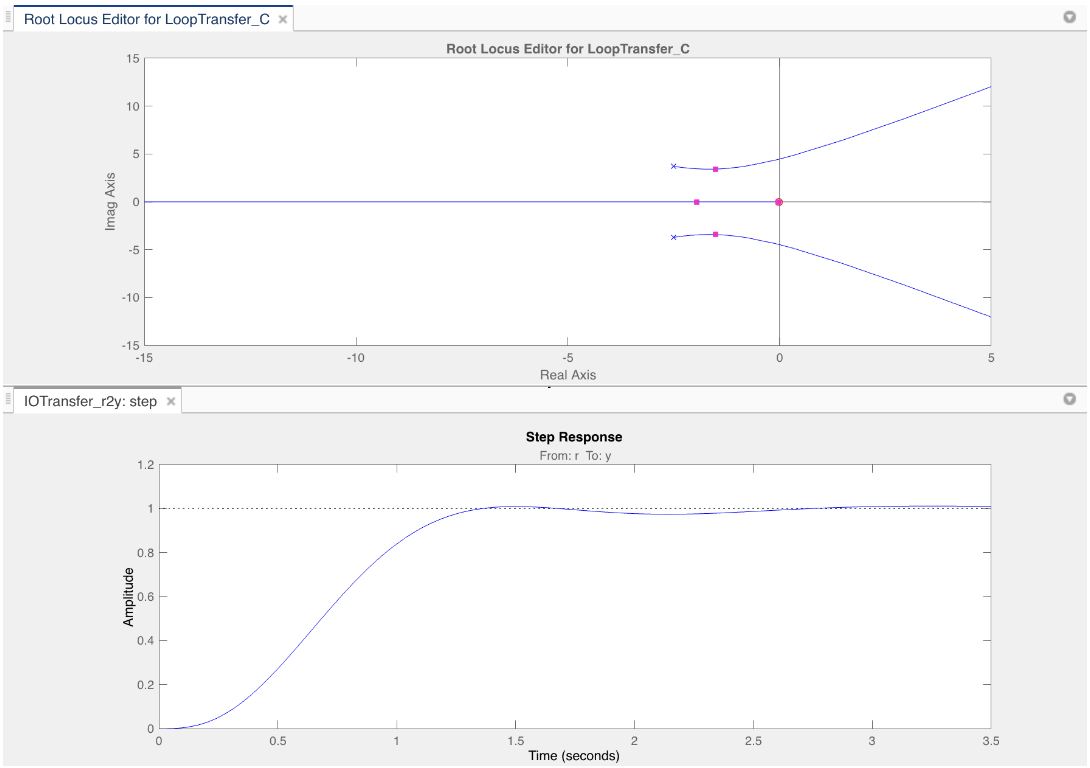


Figure 11: System with small overshoot

This system has a slower response time but it is very robust. The equation of the controller that corresponds to this system is as follows:

$$C(s) = 3.27 * (1 + 86s) \tag{2}$$

3.4.2 Aggressive but Oscillatory Configuration

We then optimized our system to have a more aggressive response. This, as expected, resulted in a oscillatory configuration that overshoots significantly more than the previous configuration. The response of this system is shown below:

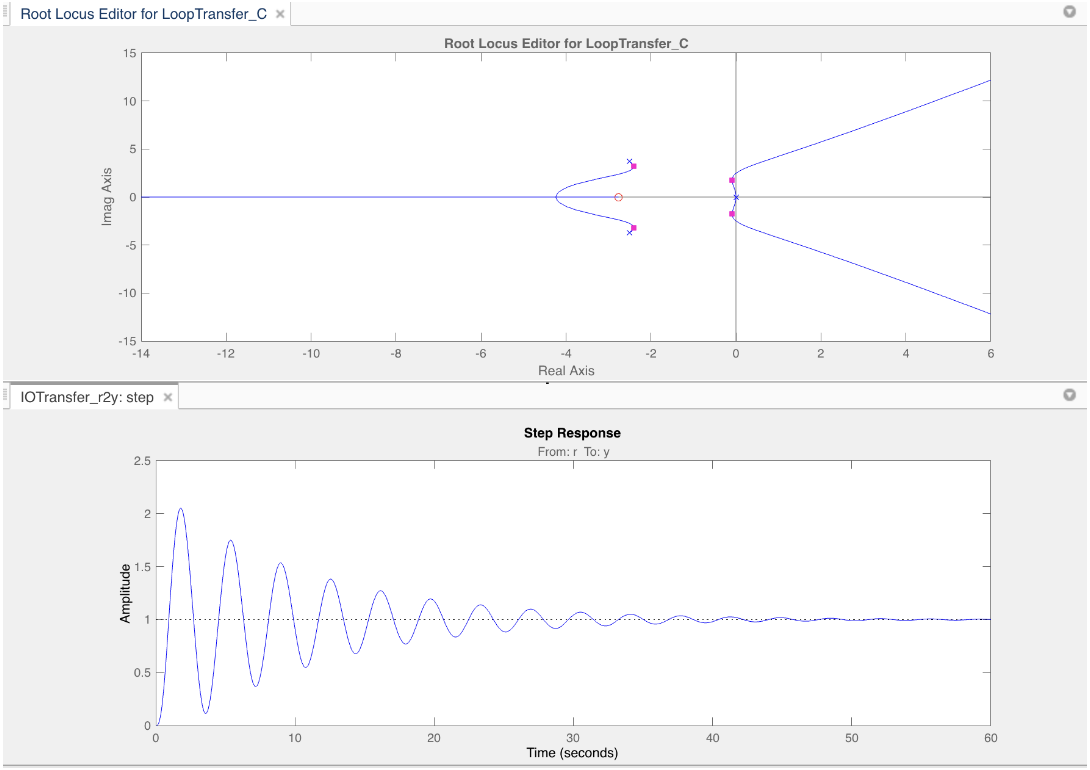


Figure 12: A more aggressive system

Note that the zero value is to the left of all poles in this configuration. Also, the controller that corresponds to this system is shown below:

$$C(s) = 502 * (1 + 36s) \tag{3}$$

3.4.3 Final Configuration

With our experience from the previous systems, we then continued with a new controller that we think is the best for our use in this project. It has very little overshoot, is fairly responsive and robust. A sample output of this configuration is shown below.

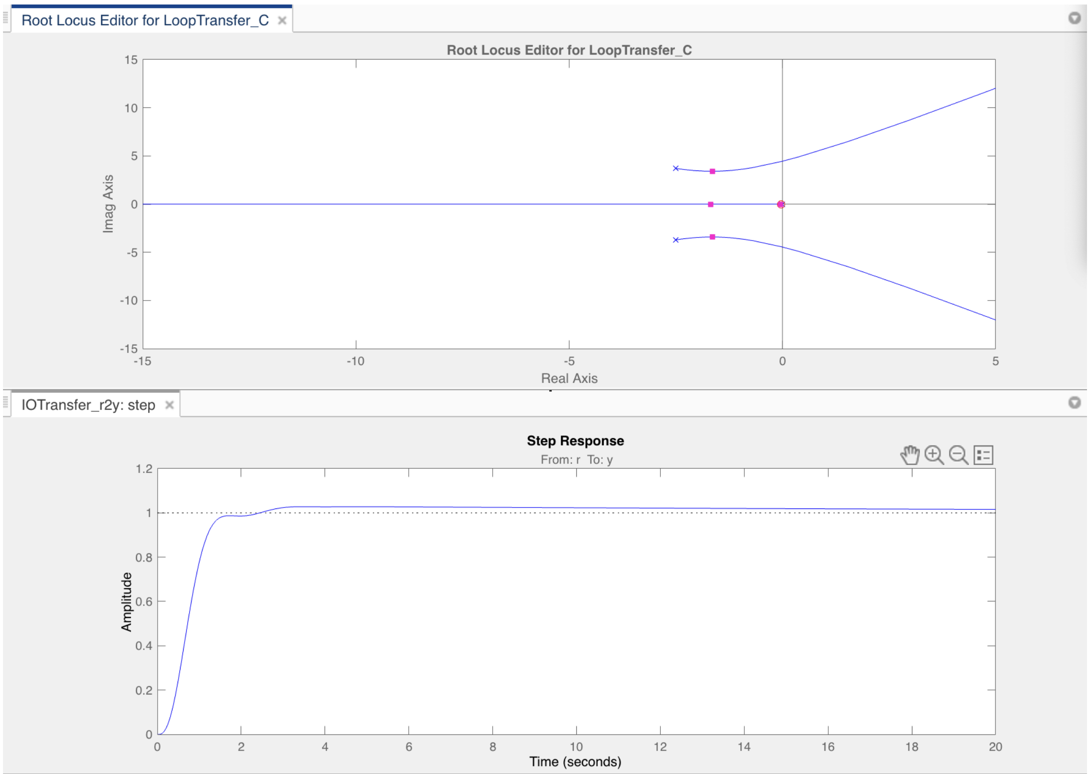


Figure 13: Proposed system

The controller that corresponds to this system is shown below:

$$C(s) = 9.75 * (1 + 26) \tag{4}$$

4 Conclusion

In conclusion, we tried different parameters for our controller and deduced that any **zero** that is introduced by the PD controller that is between 0 and -3.5 gives the opportunity to stabilize the system. However, the 3 controllers we have proposed work the best for our case.