National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Ms. Neelma Naz Dated: 21/03/2023

Semester: 6th Section: BEE 12C

EE-371: Linear Control Systems

Lab 10: PID Controller Design

Lab Instructor: Yasir Rizwan

Group Members

Name	Reg. No	Lab Report Marks	Viva Marks	Total
		10 Marks	5 Marks	15 Marks
Danial Ahmad	331388			
Muhammad Umer	345834			
Tariq Umar	334943			

1 Table of Contents

2 Model Verification

2.1 Objectives

The objectives of this lab are:

- Learn how to design a PID controller using MATLAB.
- Learn how to implement a PID controller in LabVIEW.

2.2 Introduction

The PID controller is a widely used controller, accounting for about 80% of dynamic controllers for low-order systems. Its design involves determining the values of Kp, Ki, and Kd to ensure the controller is stable and exhibits certain performance characteristics for the entire system. While the proportional part of the controller is essential, the integral and derivative parts are optional. Typically, the design process begins with a proportional controller, which is the same one designed in a previous lab on root locus. However, it may not always be possible to achieve the desired transient behavior using only a simple proportional controller, as previously observed.

2.3 Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data.

3 Lab Procedure

3.1 Design Problem I

Settling time less than 2 seconds, %OS<20, and no conditions on the steady state error for a step input. Let's look at the characteristics of the open loop system. Use MATLAB to find the settling time, overshoot, and steady state error for the open loop system. Write these values in your logbook. Do these values satisfy the design requirements? If yes, the design is complete. We do not need a controller or feedback to meet the design requirement.

However, usually, it is not a good idea to use the plant in an open loop. An open loop system is usually less robust against disturbances as compared to a closed loop system. Therefore, even if the design requirements are satisfied with the open loop system, we should use a proportional controller with a small gain.

```
% open loop system
sys = tf(1, [1 10 20]);

% open loop system properties
stepinfo(sys) % step response information of the open loop system
abs(1 - dcgain(sys)) % steady state error for a step input ...
% to the open loop system

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

err =
0.9500
```

3.2 Design Problem II

Settling time less than 1 seconds, %OS<10, and no conditions on the steady state error for a step input. As already seen, the open loop system has a settling time of around 2 seconds.

Therefore, to improve the transient response we will have to design a controller. To improve the transient response the PD controller is used. However, a good starting point is to check whether a simple proportional controller is enough to satisfy the design requirements.

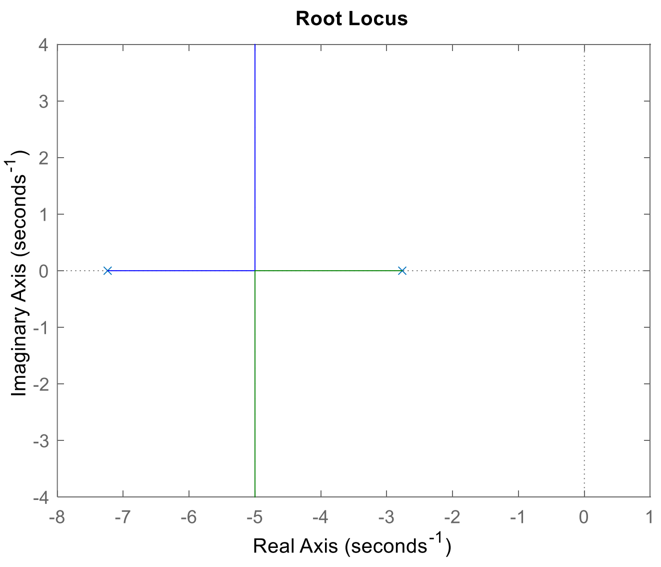
To design the proportional controller, we plot the root locus of the closed system. Try to choose a value of the gain so that you meet the design requirements. The value of the gain can be selected by using the rlocus() command in MATLAB. Once you have a value of gain K, you should test the transient response properties of the closed loop system to see if the design requirements are met.

```
% open loop system
sys = tf(1, [1 10 20]);
% open loop system properties
stepinfo(sys) % step response information of the open loop system
abs(1 - dcgain(sys)) % steady state error for a step input ...
% to the open loop system
% proportional controller
rlocus(sys)
Kp = 9; % proportional gain
ctrl = zpk([], [], Kp); % define the proportional controller TF
sys_cl = feedback(series(ctrl, sys), 1); % find the closed loop system
stepinfo(sys_cl)
err = abs(1 - dcgain(sys_cl))
```

Output

```
RiseTime: 0.5597
TransientTime: 0.9314
 SettlingTime: 0.9314
  SettlingMin: 0.2803
  SettlingMax: 0.3105
    Overshoot: 0.0388
   Undershoot: 0
         Peak: 0.3105
     PeakTime: 1.5750

err =
0.6897
```



3.3 Design Problem III

Settling time less than 0.5 second, %OS<10, and no conditions on the steady state error for a step input. Like the previous design similar problem, over here we are also concerned with only the transient response. We know that the open loop system doesn't meet these requirements. We can try a simple proportional controller, but as you may have observed in Design Problem 2, it is not possible to achieve a settling time of less than 0.5 seconds with a simple proportional controller. To further improve the transient response, we will try a PD controller.

If only the proportional and derivate parts of the PID controller are used, then it is called a PD controller and it has the following transfer function:

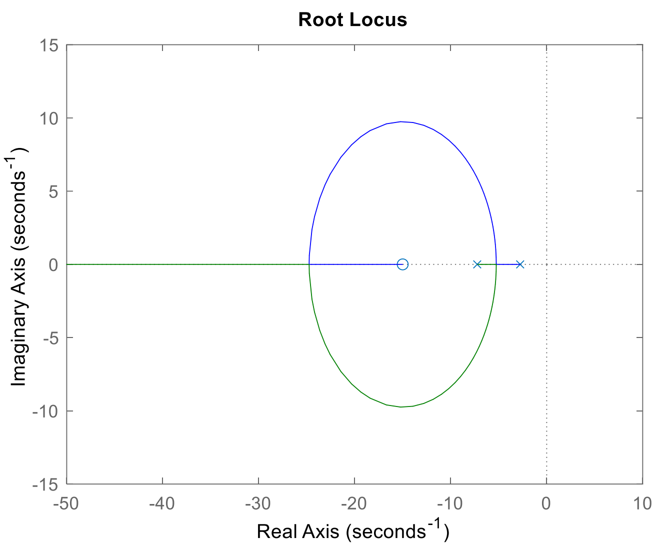
$$K_p + K_d s = K_d(s + K_p/K_d)$$

You can have different controllers based on the location of the compensator zero you choose. Remember that a good choice of the zero location will be such that the controller gains are not so large.

```
zero_loc = -15;
compensator = zpk(zero_loc, [], ...
    1);
rlocus(series(compensator, sys));
Kd = 25;
Kp = -Kd * zero_loc;
ctrl_p = zpk([], [], Kp); % proportional part of controller
ctrl_d = zpk(0, [], Kd); % derivate part of controller
ctrl = parallel(ctrl_p, ctrl_d); % controller TF
sys_cl = feedback(series(ctrl, sys), 1); % closed loop sys
stepinfo(sys_cl)
err = abs(1 - dcgain(sys_cl))

                                        Output
RiseTime: 0.0592
TransientTime: 0.2412
 SettlingTime: 0.2412
  SettlingMin: 0.8685
  SettlingMax: 1.0033
    Overshoot: 5.6858
   Undershoot: 0
         Peak: 1.0033
     PeakTime: 0.1395

err =
0.0506
```



3.4 Design Problem IV

Settling time less than 2 second, %OS<20, and zero steady state error for a step input. In this design problem the requirement is to eliminate the steady state error for a step input. For eliminating the steady state error, we can use a PI controller. If only the proportional and integral parts of the PID controller are used, then it is called a PI controller and it has the following transfer function:

$$K_p + \frac{K_i}{s} = \frac{K_p s + K_i}{s} = \frac{K_p(s + \frac{K_i}{K_p})}{s}$$

Try different values of Kp. If the design specifications are not met, change the location of zero and try again.

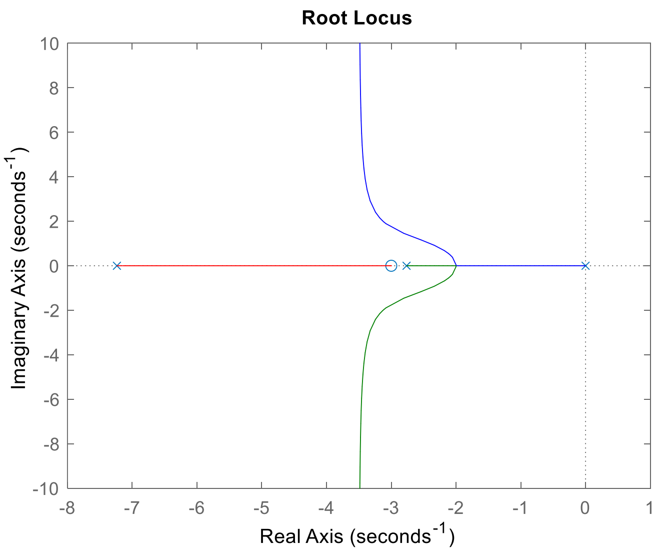
```
zero_loc = -3;
compensator = zpk(zero_loc, 0, 1);
rlocus(series(compensator, sys));
Kp = 25;
Ki = -Kp * zero_loc;
ctrl_p = zpk([], [], Kp); % proportional part of controller
ctrl_i = zpk([], 0, Ki); % integral part of controller
ctrl = parallel(ctrl_p, ctrl_i); % controller TF
sys_cl = feedback(series(ctrl, sys), 1); % closed loop sys
stepinfo(sys_cl)
abs(1 - dcgain(sys_cl))
```

Output

```
RiseTime: 0.4160
TransientTime: 1.2826
 SettlingTime: 1.2826
  SettlingMin: 0.9078
  SettlingMax: 1.0621
```

```
    Overshoot: 6.2095
   Undershoot: 0
         Peak: 1.0621
     PeakTime: 0.8774

err =
3.3307e-15
```



3.5 Design Problem V

Settling time less than 0.5 second, %OS<10, and zero steady state error for a step input. In this design problem we need to eliminate the steady state error for step input and improve the transient response characteristics. We will need all the parts of PID controller to meet the design specifications. The transfer function of the PID controller is given below:

$$K_p + K_d s + \frac{K_i}{s}$$

$$= \frac{K_d s^2 + K_p s + K_i}{s}$$

$$= \frac{K_d \left[s^2 + \frac{K_p}{K_d} s + \frac{K_i}{K_d}\right]}{s}$$

As we have already designed the PI and PD controllers, we can fix the location of zeros. If z1 and z2 are fixed, then K can be chosen with the help of the root locus. Once K is selected, then we can find the values of Kd, Kp and Ki in terms of z1, z2, and K.

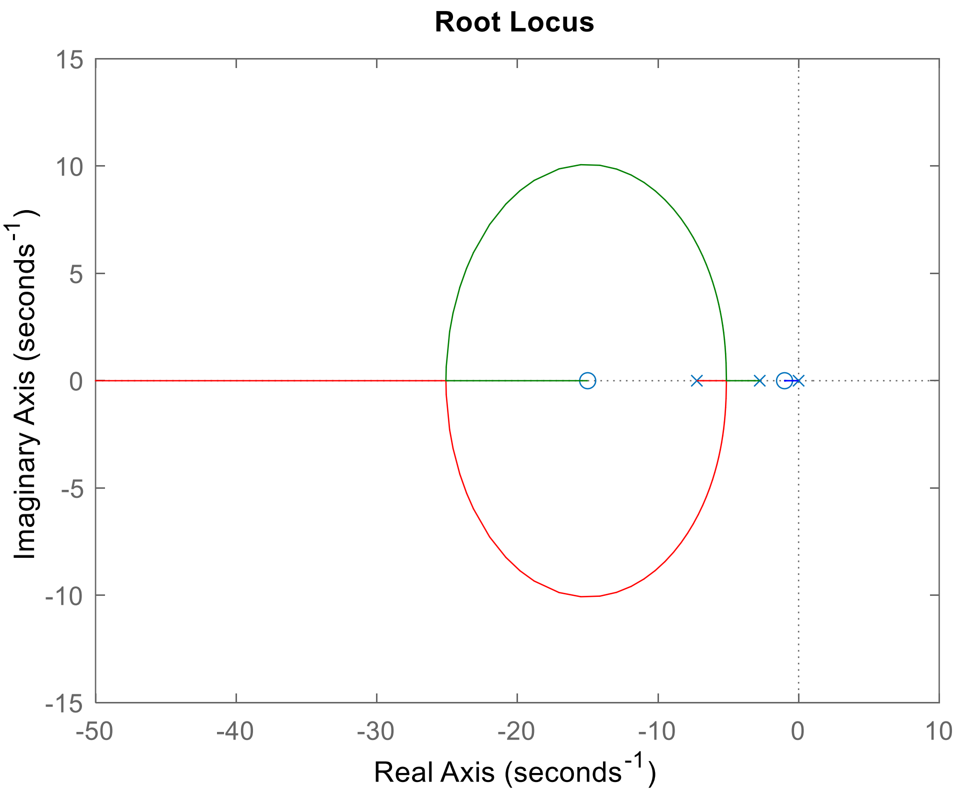
```
zero_loc_pi = -1;
zero_loc_pd = -15;
compensator = zpk([zero_loc_pi, zero_loc_pd], 0, 1);
rlocus(series(compensator, sys));
```

```
Kd = 50;
Kp = -(zero_loc_pd + zero_loc_pi) * Kd;
Ki = zero_loc_pd * zero_loc_pi * Kd;
ctrl_p = zpk([], [], Kp); % proportional part of controller
ctrl_i = zpk([], 0, Ki); % integral part of controller
ctrl_d = zpk(0, [], Kd); % derivative part of controller
ctrl = parallel(parallel(ctrl_p, ctrl_i), ctrl_d); % controller TF
sys_cl = feedback(series(ctrl, sys), 1); % closed loop sys
stepinfo(sys_cl)
abs(1 - dcgain(sys_cl))
```

Output

```
RiseTime: 0.0354
TransientTime: 0.1314
 SettlingTime: 0.1314
  SettlingMin: 0.9145
  SettlingMax: 1.0365
    Overshoot: 3.6509
   Undershoot: 0
         Peak: 1.0365
     PeakTime: 0.0876

err =
1.3323e-15
```



4 Conclusion

In conclusion, the proportional-integral-derivative (PID) controller is a widely used controller that is essential in ensuring stable and optimal performance of low-order systems. The design process involves determining the gains of Kp, Ki, and Kd to guarantee stability and specific performance characteristics of the entire system. While the proportional part of the controller is a critical component, the integral and derivative parts are optional. However, it is not always possible to achieve the desired transient behavior using only a proportional controller. Therefore, it is important to consider incorporating the integral and derivative parts in the design process when necessary. Overall, the PID controller is a powerful tool for controlling dynamic systems, and its effective application can lead to improved system performance and stability.