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: 26/04/2023

Semester: 6th Section: BEE 12C

EE-371: Linear Control Systems

Lab 11: PID Controller Implementation for QNET DC Motor

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:

- Design and implementation of controllers for speed control of DC motor
- Design and implementation of controllers for position control of DC motor

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 Exercise 1

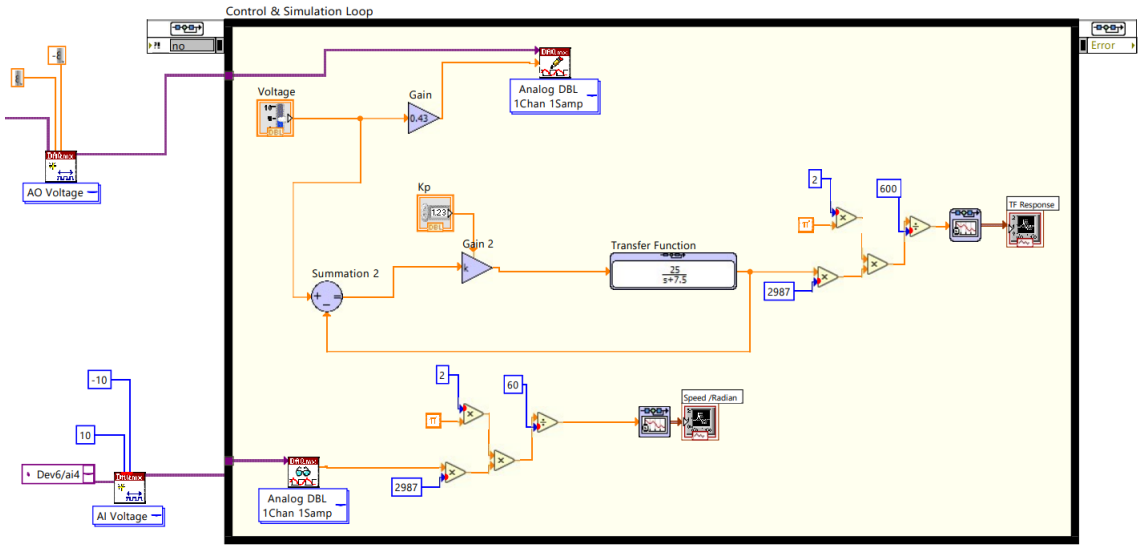
Using the techniques learnt in lab 10 and the model of QNET DC motor found in lab 3, design a simple proportional controller for the speed control of the DC motor. The controller should meet the following specifications:

- %OS
- Settling time is less than 0.5 second

```
num = 0.0334;
den = [1.566e-5, 1.11556e-3];
plant = tf(num, den);

Kp = 0.038305;
controller = zpk([], [], Kp); % proportional controller
sys = feedback(series(plant, controller), 1);
display(sys)
stepinfo(sys)
                                        Output
sys =
   81.698
  ---------
  (s+152.9)

Continuous-time zero/pole/gain model.

         RiseTime: 0.0144
    TransientTime: 0.0256
     SettlingTime: 0.0256
      SettlingMin: 0.4832
      SettlingMax: 0.5338
        Overshoot: 0
       Undershoot: 0
             Peak: 0.5338
         PeakTime: 0.0479
```

3.2 Exercise 2

Using the techniques learnt in previous labs implement the proportional controller on the actual plant (i.e., the QNET DC Motor). Using data acquisition, acquire the response of the control system to a step input and see whether the design specifications have been met or not. Load the motor by applying slight friction with your hands. Observe if it maintains the speed or not? How does this differ from open loop control? Comment on your observations in your lab report.

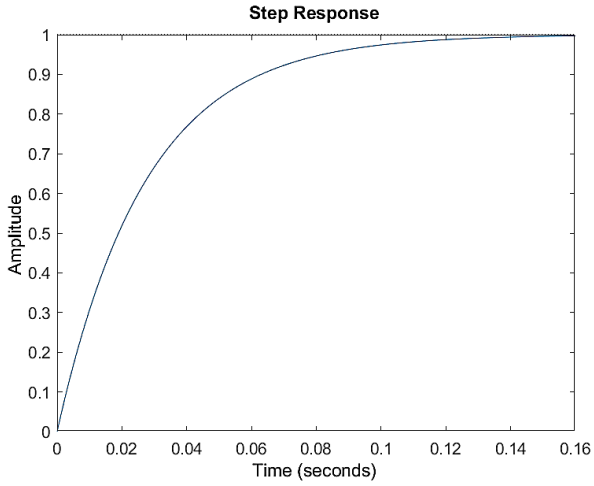


3.3 Exercise 3

Design a PI controller for the speed control of the DC motor. The controller should meet the following specifications:

- %OS
- Settling time is less than 1 seconds
- Zero steady state error for step input

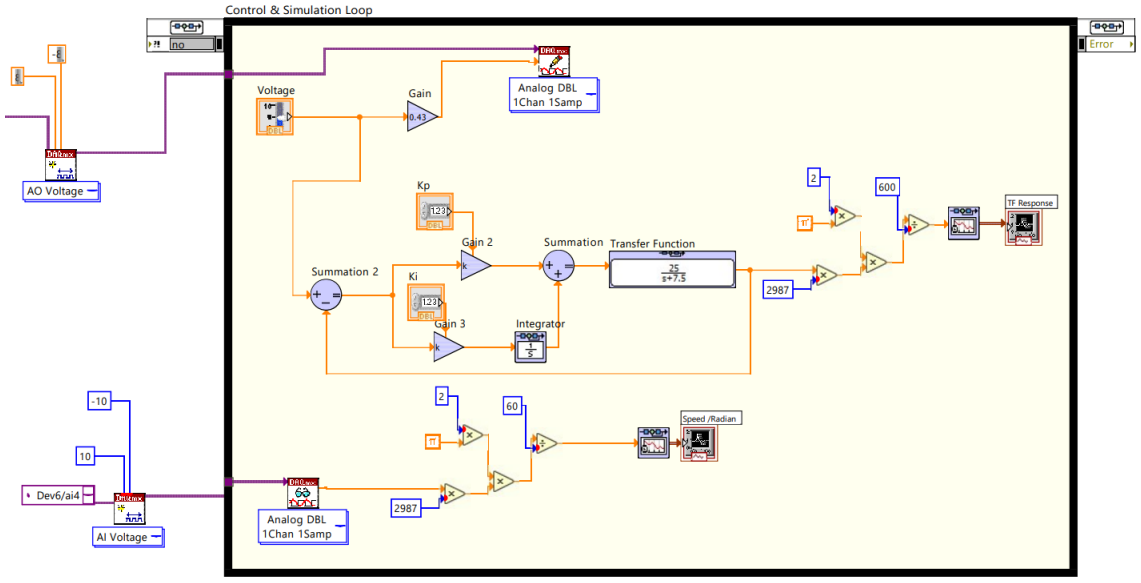
```
num = 0.0334;
den = [1.566e-5, 1.11556e-3];
plant = tf(num, den);

Kp = 0.01715;
Ki = 1.2221;
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, plant), 1); % closed loop sys
stepinfo(sys_cl)
err = abs(1 - dcgain(sys_cl));
step(sys_cl)
```




```
       RiseTime: 0.0600
  TransientTime: 0.1069
   SettlingTime: 0.1069
    SettlingMin: 0.9044
    SettlingMax: 0.9993
      Overshoot: 0
     Undershoot: 0
           Peak: 0.9993
       PeakTime: 0.2000
```

3.4 Exercise 4

Using the techniques learnt in previous labs implement the proportional controller on the actual plant (i.e., the QNET DC Motor). Using data acquisition, acquire the response of the control system to a step input and see whether the design specifications have been met or not. Load the motor by applying slight friction with your hands. Observe if it maintains the speed or not? How does this differ from open loop control? Comment on your observations in your lab report.



3.5 Exercise 5

Design a simple proportional controller for the position control of the DC motor, such that the closed loop systems is critically damped. Implement this controller on the QNET DC Motor. Using data acquisition, acquire the response of the control system to a step input and see whether the design specifications have been met or not. Load the motor by applying slight friction with your hands. Observe if it gets to the desired position or not? How does this differ from open loop control? Comment on your observations in your lab report.

```
num = 0.0334;
den = [1.566e-5, 1.11556e-3, 0];
plant = tf(num, den);

Kp = 0.5;
controller = zpk([], [], Kp); % proportional controller
sys = feedback(series(plant, controller), 1);
display(sys)
stepinfo(sys)
                                        Output
sys =

        1066.4
  ------------------
  (s+21.4) (s+49.84)

Continuous-time zero/pole/gain model.

         RiseTime: 0.1168
    TransientTime: 0.2090
     SettlingTime: 0.2090
      SettlingMin: 0.9032
      SettlingMax: 1.0000
        Overshoot: 0
       Undershoot: 0
             Peak: 1.0000
         PeakTime: 0.4971
```

The motor was able to reach the desired position, implying that the closed loop system with the controller allows for the plant model to be adaptive to the environment, whereas in open loop control, the motor was unable to reach the desired position upon application of friction.

3.6 Exercise 6

Design a PD controller for the position control of the DC motor, to meet the following specifications.

- %OS
- Settling time is less than 0.5 seconds

```
num = 0.0334;
den = [1.566e-5, 1.11556e-3, 0];
plant = tf(num, den);

Kd = 0.05;
Kp = 0.5;
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, plant), 1); % closed loop sys
stepinfo(sys_cl)
```

```
                                        Output
sys_cl =

     106.64 (s+10)
  -------------------
  (s+6.212) (s+171.7)

Continuous-time zero/pole/gain model.

         RiseTime: 0.2193
    TransientTime: 0.4794
     SettlingTime: 0.4794
      SettlingMin: 0.9001
      SettlingMax: 0.9980
        Overshoot: 0
       Undershoot: 0
             Peak: 0.9980
         PeakTime: 0.8472
```

3.7 Exercise 7

Note that in the transfer function of motor position vs voltage there is a pole at the origin. Therefore, the system type is 1. Consequently, the closed loop system will have zero steady state error for a step input. If the requirement is to have zero steady state error for step input, then we do not need a PI controller. If the input is a ramp, then there will be a non-zero steady state error. In that case we may have a PI compensator to increase the steady state error. Design a PID controller to meet the following specifications:

- %OS
- Settling time is less than 0.5 seconds
- Zero steady state error for ramp input

```
num = 0.0334;
den = [1.566e-5, 1.11556e-3, 0];
plant = tf(num, den);
```

```
Kd = 0.029039;
Kp = 1.1021;
Ki = 6.3464;
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, plant), 1); % closed loop sys
err = abs(1 - dcgain(sys_cl))
                                          Output
sys_cl =

     61.935 (s+30.87) (s+7.079)
  --------------------------------
  (s+113.5) (s^2 + 19.66s + 119.2)

Continuous-time zero/pole/gain model.

         RiseTime: 0.0585
    TransientTime: 0.4293
     SettlingTime: 0.4293
      SettlingMin: 0.9026
      SettlingMax: 1.0983
        Overshoot: 9.8325
       Undershoot: 0
             Peak: 1.0983
         PeakTime: 0.1826

err =
   6.6613e-16
```

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.