**Department of Electrical Engineering and   
Computer Science**

**Faculty Member:** Ma’am Neelma Naz **Dated:** 21/09/2022

**Semester:** 6th **Section:** BEE 12C

EE379: Control Systems

**Lab 11: PID controller Implementation for QNET DC Motor**

**Lab Instructor: Sir. Yasir Rizwan**

**Group Members**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Student Name | Reg. No. | Lab Report Marks / 10 | Viva Marks  / 5 | Total  /15 |
| Muhammad Ahmed Mohsin | 333060 |  |  |  |
| Imran Haider | 332569 |  |  |  |
| Zafar Azhar | 340908 |  |  |  |

Table of Contents

[2 Objectives 2](#_Toc134049054)

[3 Lab Task 1 2](#_Toc134049055)

[3.1 MATLAB Code 3](#_Toc134049056)

[3.2 Screenshot 4](#_Toc134049057)

[4 Lab Task 2 5](#_Toc134049058)

[4.1 MATLAB code: 5](#_Toc134049059)

[4.2 Screenshot: 7](#_Toc134049060)

[5 Lab Task 3 8](#_Toc134049061)

[5.1 MATLAB code: 8](#_Toc134049062)

[5.2 Screenshot: 10](#_Toc134049063)

[6 Conclusion: 12](#_Toc134049064)

**PID controller Implementation for QNET DC Motor**

# Objectives

The objectives of this lab are:

1. Understand the principles of PID control based on specific applications.

2. Design a PID controller using MATLAB and analyze its performance in terms of stability, settling time, and steady-state error for a given application.

3. Tune the three different gains of PID controller to get the desired response in each case.

# Lab Task 1

Design Problem 1 : Design of proportional controller for motor speed 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 : 1. %OS < 25% 2. Settling time is less than 0.5 seconds.

## MATLAB Code

%# Clear the workspace and command window

clear

clc

%# Define the numerator and denominator of the open-loop system

num = [25];

den = [1 7.5];

%# Create the transfer function of the open-loop system

sys = tf(num, den);

%#%% Open-loop system properties

%# Get the step response information of the open-loop system

sys\_param = stepinfo(sys);

%# Get the steady-state error of the open-loop system

SSError = abs(1-dcgain(sys));

%#%% Proportional controller

%# Plot the root locus of the open-loop system

rlocus(sys);

%# Define the proportional gain

Kp = 0.3;

%# Create the proportional controller

ctrl = zpk([],[],Kp);

%# Create the closed-loop system

sys\_cl = feedback(series(ctrl,sys),1);

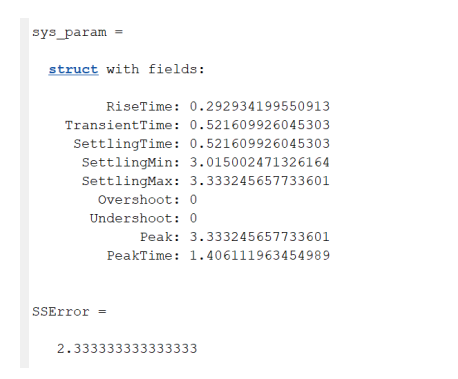
%# Get the step response information of the closed-loop system

sys\_param = stepinfo(sys\_cl);

%# Get the steady-state error of the closed-loop system

SSError = abs(1-dcgain(sys\_cl));

## Screenshot



Table

Description automatically generated with medium confidence

Chart, box and whisker chart

Description automatically generated

We observe that the overshoot is less that 25 percent and settling time is between 0.25 to 0.5 seconds.

# Lab Task 2

Design Problem 6 : Design and implementation of a PD controller for motor position Design a PD controller for the position control of the DC motor, to meet the following specifications : 1. OS < 25% 2. Settling time is less than 0.5 seconds.

## MATLAB code:

%# Clear the workspace and command window

clear

clc

%# Define the numerator and denominator of the open-loop system

num = [25];

den = [1 7.5 0];

%# Create the transfer function of the open-loop system

sys = tf(num, den);

%#%% Open-loop system properties

%# Get the step response information of the open-loop system

sys\_param = stepinfo(sys);

%# Get the steady-state error of the open-loop system

SSError = abs(1-dcgain(sys));

%#%% PD controller design

%# Display a message

display('Closed Loop System after PD Controller Design');

%# Define the zero location of the compensator

zero\_loc = -15;

%# Create the compensator transfer function

compensator = zpk(zero\_loc,[],1);

%# Plot the root locus of the open-loop system with the compensator

rlocus(series(compensator,sys));

%# Define the proportional gain and derivative gain of the controller

Kd = 1;

Kp = -Kd\*zero\_loc;

%# Create the proportional and derivative parts of the controller

ctrl\_p = zpk([],[],Kp); %proportional part of controller

ctrl\_d = zpk(0,[],Kd); %derivative part of controller

%# Create the controller transfer function

ctrl = parallel(ctrl\_p,ctrl\_d);

%# Create the closed-loop system

sys\_cl = feedback(series(ctrl,sys),1);

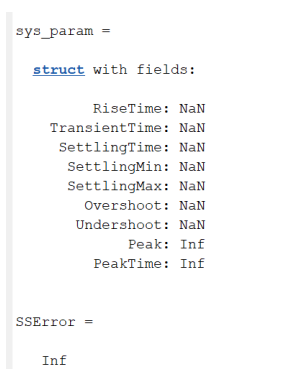
%# Get the step response information of the closed-loop system

sys\_param = stepinfo(sys\_cl);

%# Get the steady-state error of the closed-loop system

SSError = abs(1-dcgain(sys\_cl));

## Screenshot:



Graphical user interface, text, application

Description automatically generated

Diagram

Description automatically generated

In this we observe that the overshot in less than 25 percent and the settling time is less that 0.5 seconds as per requirement. Moreover, there were no restrictions on the steady state error.

# Lab Task 3

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. 1. %OS < 25 2. Settling time is less than 0.5 seconds 3. Zero steady state error for ramp input

## MATLAB code:

%# Clear the workspace and command window

clear

clc

%# Define the numerator and denominator of the open-loop system

num = [25];

den = [1 7.5 0];

%# Create the transfer function of the open-loop system

sys = tf(num, den);

%#%% Open-loop system properties

%# Get the step response information of the open-loop system

sys\_param = stepinfo(sys);

%# Get the steady-state error of the open-loop system

SSError = abs(1-dcgain(sys));

%#%% PID controller design

%# Display a message

display('Closed Loop System after PID Controller Design');

%# Define the zero locations of the compensator

zero\_loc\_pi = -1;

zero\_loc\_pd = -15;

%# Create the compensator transfer function

compensator = zpk([zero\_loc\_pi, zero\_loc\_pd],0,1);

%# Plot the root locus of the open-loop system with the compensator

rlocus(series(compensator,sys));

%# Define the proportional, integral, and derivative gains of the controller

Kd = 1;

Kp = -(zero\_loc\_pd+zero\_loc\_pi)\*Kd;

Ki = zero\_loc\_pd\*zero\_loc\_pi\*Kd;

%# Create the proportional, integral, and derivative parts of the controller

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

%# Create the controller transfer function

ctrl = parallel(parallel(ctrl\_p,ctrl\_i),ctrl\_d);

%# Create the closed-loop system

sys\_cl = feedback(series(ctrl,sys),1); % closed loop sys

%# Get the step response information of the closed-loop system

sys\_param = stepinfo(sys\_cl);

%# Get the steady-state error of the closed-loop system

SSError\_Ramp = abs(1-dcgain(sys\_cl));

%# Define the ramp input

ramp = tf([1],[1 0]);

%# Create the ramp-modified closed-loop system

sys\_cl1 = series(sys\_cl,ramp);

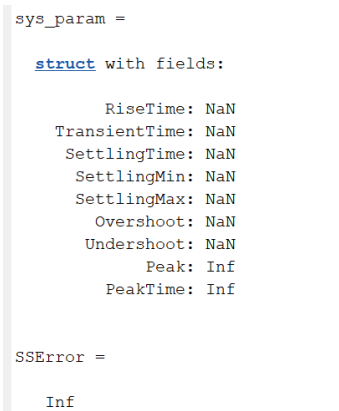
%# Get the step response information of the ramp-modified closed-loop system

sys\_param = stepinfo(sys\_cl1);

%# Get the steady-state error of the ramp-modified closed-loop system

SSError\_Ramp = abs(1-dcgain(sys\_cl1));

## Screenshot:



Text

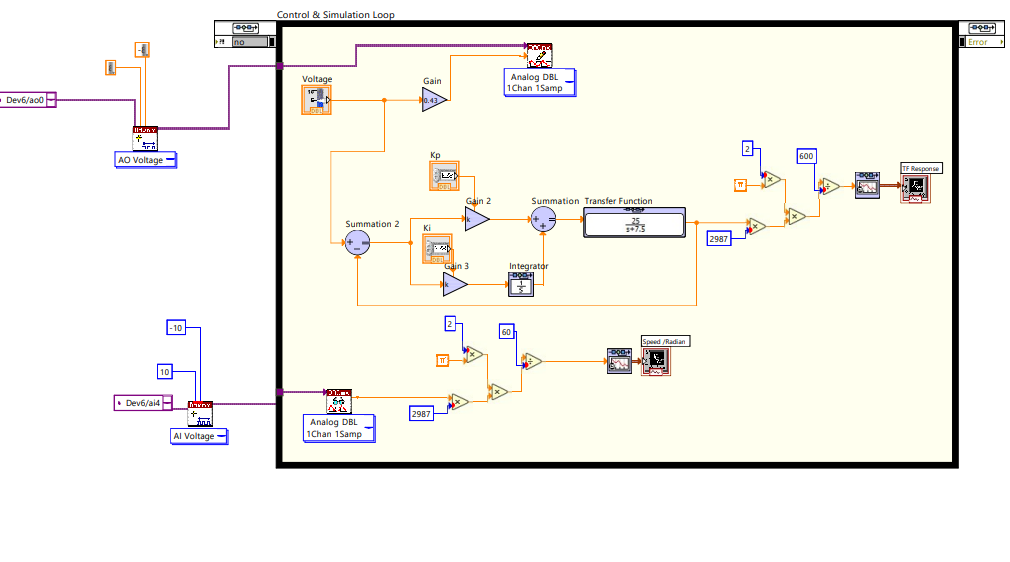
Description automatically generated

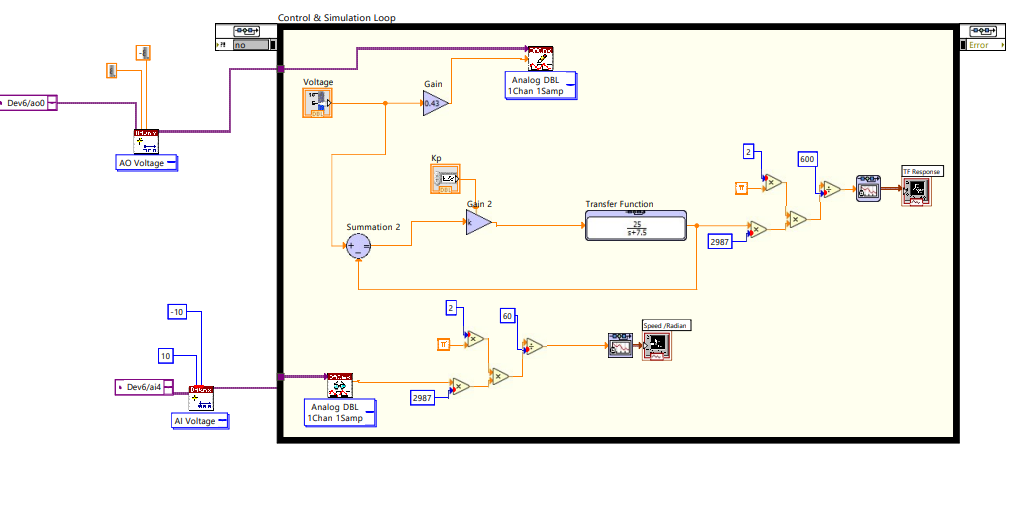
Diagram

Description automatically generated

In this we observe that the overshoot was less than 25 percent and the settling time is 0.311 seconds which is also less than 0.5 seconds. The steady state error is so small that it can be approximated to zero almost.

# Hardware components:





# Conclusion:

This lab was a valuable learning experience in the principles and practical applications of PID controllers. We learned how to implement different types of controllers using MATLAB, and obtained specific values for overshoot percentage, settling time, and steady-state error for each type of controller. Through these exercises, we also gained a deeper understanding of the importance of choosing the appropriate controller type for a given system, and the impact of controller parameters on system response. Overall, this lab provided a hands-on opportunity to apply the theoretical concepts of control systems and deepen our understanding of PID control.

Here are some of the key points from the lab:

PID controllers are a type of feedback controller that are widely used in a variety of applications.

PID controllers can be implemented using a variety of methods, including MATLAB.

The performance of a PID controller can be characterized by its overshoot percentage, settling time, and steady-state error.

The choice of PID controller type and parameters can have a significant impact on the performance of the system.

This lab provided a hands-on opportunity to learn about the principles and practical applications of PID controllers.