EEE 342 Lab-1 Report
Spring 2025

Emir Arda Bayer
22201832

Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey

1. Introduction

The first lab's purpose was to design a PI motor by making use of a DC motor. First order approximations were applied to interpret the motor system, and the digital data from the motor was used in Simulink to examine the system behavior. A PI controller was later designed by using the found transfer function, and the responses of the system was evaluated.

2. Laboratory Content

2.1 Hardware Connection and Configuration

The first part was about applying the correct settings and changes to the digital environment in order to correctly examine the physical data. The raw data after making the correct adjustments were as in Figure 1.

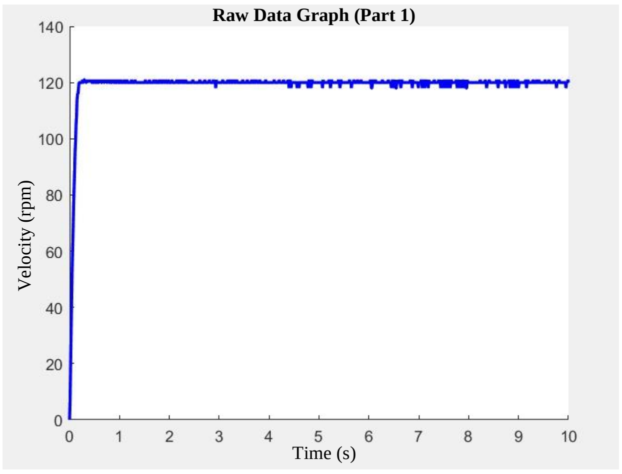

Figure 1: Raw Data

Having obtained the raw graph of velocity and time with noise, first a filtering process and then a first order approximation for the DC motor system will be made.

2.2 System Identification in Time Domain

The received data in Part 1 was later used to evaluate the DC motor by making a first order approximation. Here, the first step was to reduce the noise on the received data on Simulink, with the system as in Figure 2.

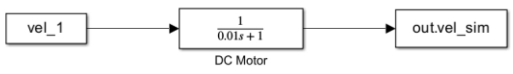

Figure 2: LPF for Noise Filtering

This filtered data was used to find the coefficients of the output equation from the preliminary part. The unit step function in the lab resulted in the value 6 being updated to 12.

$$y(t) = 12K(1 - e^{\frac{-t}{\tau}})u(t)$$

Using the mean function of MATLAB, the gain constant (12K) was found and saved. Again, similarly to the preliminary part, an instance from the filtered data between 0 and 0.1 s was evaluated and saved. As τ goes to infinity, y(t) equals 12K. Finding the mean between 2 and 10 seconds, K was found to be 13.75. The evaluated time instance, being the 10'th one, revealed the time constant to be 0.131. The code for the calculations were as:

```
steady = round(mean(out.velocity(out.tout > 2 & out.tout < 10)));
k = steady / 12;

chosen_t = out.tout(10);
chosen_v = out.velocity(10);
time_cons = -chosen_t / log(1 - chosen_v / (steady));
```

Figure 2: Part 2 MATLAB Calculations

The transfer function was found to be as below, and the outputs of the system was portrayed as in the graph in Figure 3.

$$Y(s) = \frac{13.75}{0.131s + 1}$$

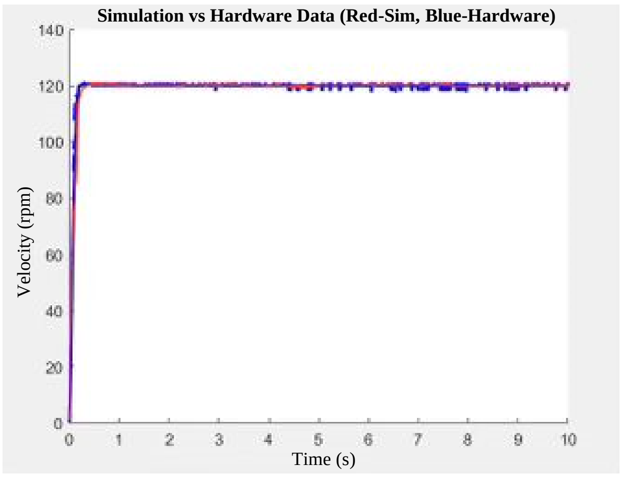

Figure 3: Hardware and Simulation Results

Due to noise in hardware results, there is a slight difference but graph verifies the approximation and implies that it is ready for the DC model.

2.3 PI Controller on Simulation System

As the mathematical model for the transfer function was obtained, three different PI controllers are to be designed to evaluate the changing behavior of the step response when the coefficients of the controller changes. In the manual, three requirements were given as below.

a) Steady state error is 0.
b) Maximum percentage overshoot is less than 8
c) Settling time is less than 0.8 seconds (%2 error bound).

The default values of the controller were already close to the given requirements, by trial and error the coefficients were chosen as $K_P = 0.5$ and $K_I = 0.8$. The output graph with these coefficients that satisfies all criteria is as below.

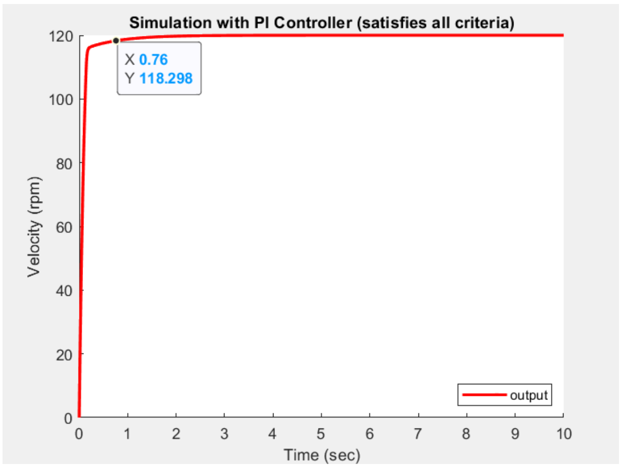

Figure 4: $K_P = 0.5$, $K_I = 0.8$

Two more PI controllers with different K_P and K_I values were created to understand how the step response changes. In the first one, $K_P = 10K_I$ and $20K_P = K_I$ in the second one. The output graphs for these coefficients are as below.

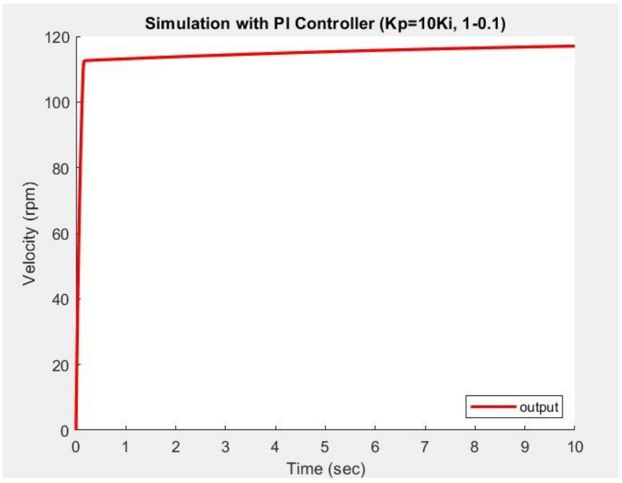

Figure 5: $K_P = 10K_I,\ K_P = 1,\ K_I = 0.1$

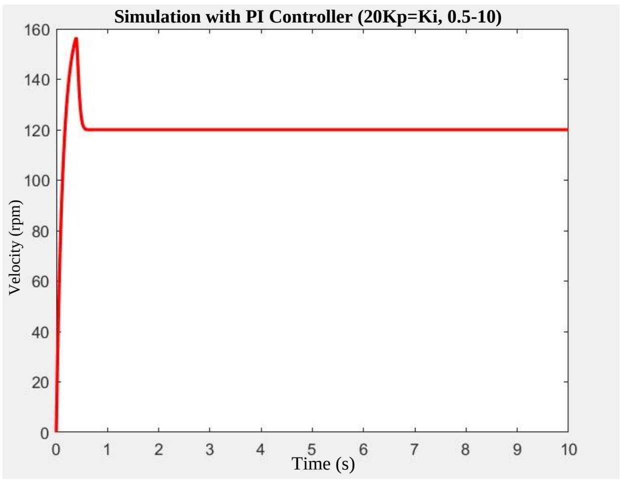

Figure 6: $20K_P = K_I,\ K_P = 0.5,\ K_I = 10$

Table 1 displays the overshoot, settling time and steady state error values for these PI controllers with changing coefficients.

	$K_P = 0.5$ $K_I = 0.8$	$K_P = 1$ $K_I = 0.1$	$K_P = 0.5$ $K_I = 10$
Overshoot (%)	0	0	29.2
Settling Time (s)	0.76	0.2	0.84
Steady State Error (s)	0	2.6	0

Table 1: Changing Behavior with Different Coefficients

The results demonstrate that as K_P increases settling time decreases, however results in higher steady state error. On the other hand, the increase of K_I imply less steady state error but more overshoot.

2.4 PI Controller on Physical System

The found coefficients in Part 3 that satisfy all criteria was used in Part 4 to implement a final design, and this system's close7.d-loop response was evaluated experimentally. The block diagram of the final system was as in Figure 7.

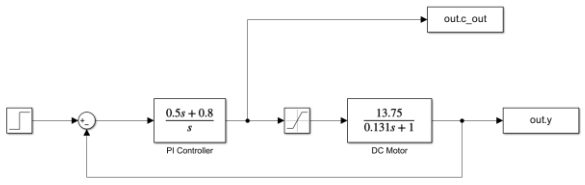

Figure 7: Block Diagram of Final Design

The graph that displays simulation and hardware data together is as follows:

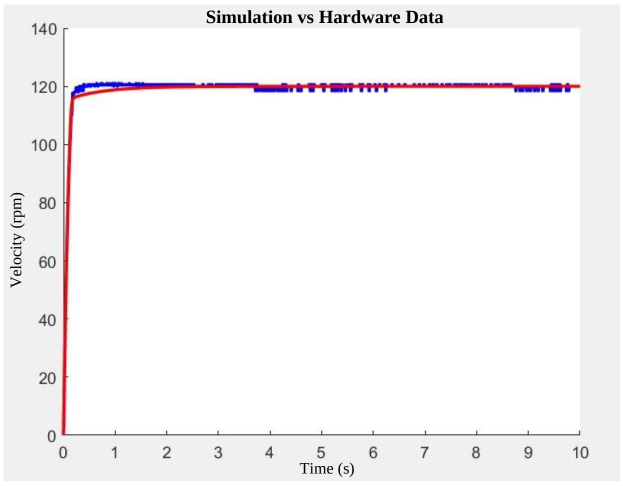

Figure 8: Hardware and Simulation Data

Comparing the hardware data to the simulation data, there is a minor difference resulting from the noise from the hardware one. Evaluating the data to criteria in Part 3, the steady state error and overshoot are 0%, and the settling time is 0.41 seconds.

3. Conclusion

The purpose of this lab was to model a DC motor by using a first order approximation and later make use of PI controllers in implementing a final design to satisfy given requirements. The final design was verified by comparing its output in hardware and simulation on Simulink. Regarding PI controllers, three systems were designed, two for understanding the effect of changing coefficients, and one final design that meets all criteria. This final design was then implemented and its output in hardware and simulation was compared to verify the system. In the final system, all criteria were met, the only difference between the implementations was the noise in the physical implementation that occurs during receiving digital data. System identification and simulation was useful to evaluate the changing behavior of the system when its inputs or the system's own qualities changes, such as the transfer function, to achieve a final design properly that satisfies all possible expectations. This lab was useful in terms of understanding PI controllers and be introduced to feedback control systems.

4. MATLAB Code

```
PART 1
vel_1=out.velocity;
plot(vel_1)

PART2
steady = round(mean(out.velocity(out.tout > 2 & out.tout < 10)));
k = steady / 12;

chosen_t = out.tout(10);
chosen_v = out.velocity(10);
time_cons = -chosen_t / log(1 - chosen_v / (steady));

figure;
plot(out.tout, out.velocity,'b','LineWidth',2);
hold on;
plot(out.tout, steady * (1 - exp(-out.tout/time_cons)),'r','LineWidth',2);
hold off;
xlabel('Time (sec)'); ylabel('Velocity (rpm)');
title('Velocity vs Time');
legend({'Hardware result','Simulation result'},'Location','SouthEast');


PART 3

figure;

hold on;
plot(out.tout, out.y,'r','LineWidth',2);
xlabel('Time (sec)'); ylabel('Velocity (rpm)');
title('Simulation with PI Controller (satisfies all criteria 0.5-0.8)');
legend({'output'},'Location','SouthEast');

figure;
hold on;
plot(out.tout, out.y,'r','LineWidth',2);
xlabel('Time (sec)'); ylabel('Velocity (rpm)');
title('Simulation with PI Controller (Kp=10Ki, 1-0.1)');
legend({'output'},'Location','SouthEast');
figure;
hold on;
plot(out.tout, out.y,'r','LineWidth',2);
xlabel('Time (sec)'); ylabel('Velocity (rpm)');
title('Simulation with PI Controller (Kp=20Ki, 0.5-10)');
legend({'output'},'Location','SouthEast');

PART4

part3_out = load('outy_part3.mat');

figure;
hold on;
plot(out.tout, out.velocity,'b','LineWidth',2);
plot(out.tout, part3_out.out.y,'r','LineWidth',2);
```