ME 586: Project 2 Report
DC Motor Speed Control

Harsh Savla & TJ Wiegman

2022-12-14

Abstract

In this lab, an STM32F100RB microcontroller programmed in C was used to control a DC electric motor. The overall system was first identified by measuring the step response of the motor, and Simulink was used to model the system's response with various inputs and controls. Effective K_p and K_i coefficients were found, and the system was successfully run with a PI controller. Disturbances were induced in the system through an external summing amplifier. The controller was able to stabilize itself when disturbed, but there was a drop in the output RPM. From this project, the authors not only learned to design a PI controller for a DC electric motor but also the significance of the effects of stray voltage and other physical parameters on the output of the overall system.

1 System Identification

First and foremost, the system was calibrated across input voltages ranging from -5 to 4.65 V, which produced output motor speeds from -2300 to +1300 RPM. Based on the information gathered during calibration, system identification was carried out for step inputs of 0 to 6.5 V.

Upon receiving the step input voltage, the motor responded as shown in Figure 1. A more detailed look at the curve, just before and after the step response was received, is shown in Figure 2. As the latter makes quite clear, this is a first-order system, as the first derivative has an obvious and sharp discontinuity upon receiving the step input at time 2 seconds.

Figure 1: The motor's step response. It is a fairly fast and noisy system.

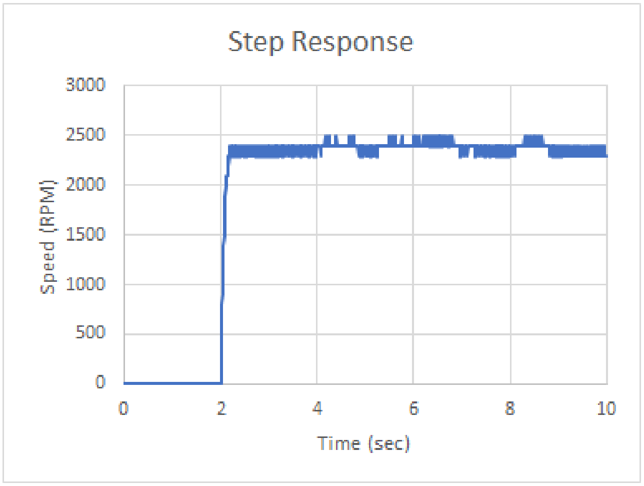
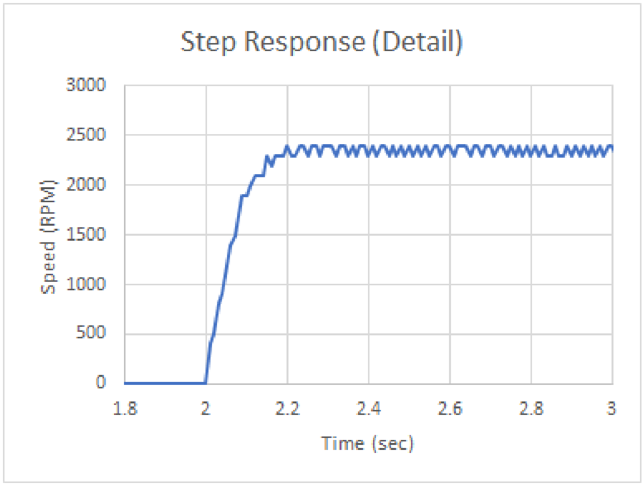


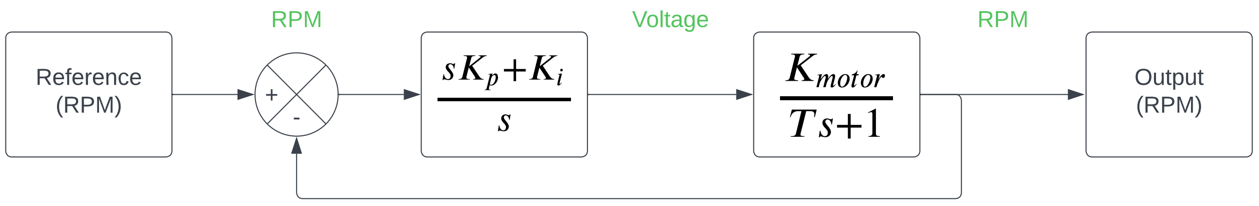
Figure 2: The motor's step response, zoomed in for better detail.



2 Controller Design

This system had the requirement to reject any external disturbances and have no steady-state error. In order to achieve these goals, a simple PI controller was sufficient. The system, with controller installed, was modeled as shown in Figure 3.

Figure 3: A PI controller gives input voltage to a first-order electric motor, which feeds its output speed back to the PI controller. Units for each signal are shown in green.



3 Simulation and Implementation

A simulink model of the system was created as shown in Appendix A.1. Using this simulation, several different controller coefficients K_p and K_i were tested. The values that were found to be effective were 0.01 for both. The simulated output for a 1000 RPM step input, using those 0.01 coefficients, is shown in Figure 4. As the simulation shows the system following extremely closely to the input signal, the same controller was tested on the physical system. Recorded output for the physical system, using the same input and controls, is shown in Figure 5.

To test how robust the system was to outside disturbances, the motor's control input was connected to a summing amplifier. This allowed the microcontroller's output signal to be added with an external disturbance (in this case, a manually activated $-3\,\mathrm{V}$ step) before reaching the motor. While the controller was not able to fully compensate for this disturbance, it did not destabilize and continued to produce a constant speed. The recorded output for that experiment is shown in Figure 6.

Figure 4: Simulated output for a 1000 RPM step input as the reference signal.

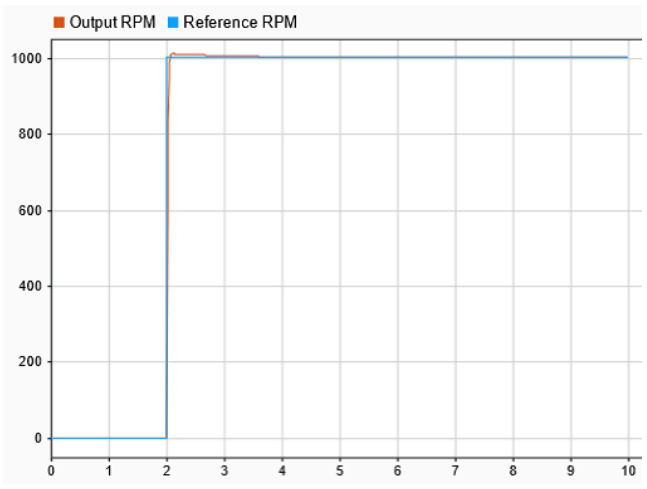


Figure 5: Recorded output for a 1000 RPM step input as the reference signal on a physical system.

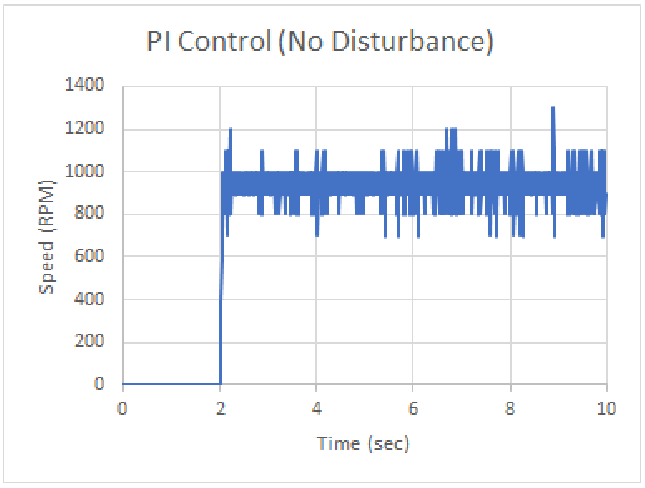
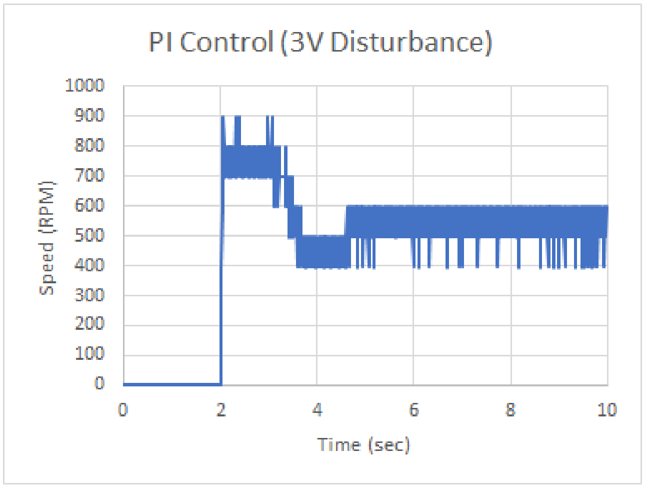


Figure 6: Recorded output for a 1000 RPM step input, with reactions to an external disturbance.

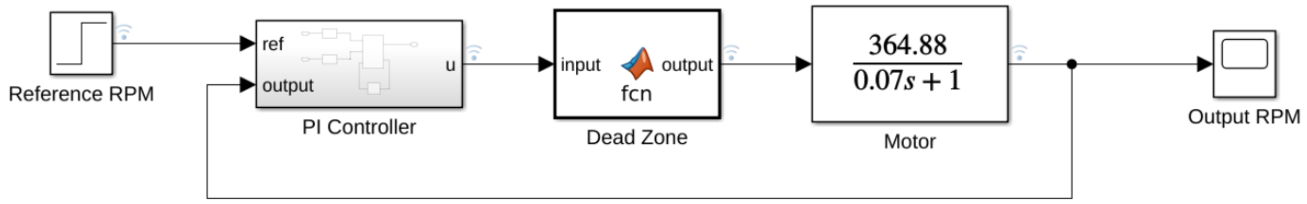


4 Conclusion

This lab gave excellent practice on the common tasks of system identification and controller design for a practical electromechanical system. The most important learning outcome from this project was though we simulate the controller and include all the physical aspects in the system, there may be a possibility of unaccounted disturbances and effects which may not produce desired results. Initially, without including the disturbances, our controller seemed to working flawlessly. But after the addition of the summing amplifier to the system, we observed that the controller was able to reject the disturbances but at the expense of the drop in the output RPM.

A Appendix

A.1 Simulink System Model (With PI Controller)



A.2 PID Controller (Timer Interrupt) Code

```
// Define statements
#define EXTI_PR 0x40010414
#define ZEROOUT 1.6 // determined empirically
#define OUTPUT_MAX 10.0
#define OUTPUT_MIN 0.0
#define samplerate 10 // milliseconds

// Global variables (also used in main function)
int ram_ptr[1000]; // measured data stored in memory
int stored_actual = 0; // number of values actually written
```

```c
int start_record = 0;
float Kp = 0.01;
float Ki = 0.01;
float Kd = 0;
float error_prior = 0;
float integral = 0;
int clicks = 0;
int outclicks = 0;
int desclicks = 0;
int *p = (int *) EXTI_PR;

// Timer interrupt routine
void timehand(void){
   // Setup
   float error;
   float derivative;
   float output;
   float reference = 0;

   // Wait to change until 2 seconds
   if (stored_actual > 200) {
      reference = desclicks;
   }

   // Convert to RPM and reset clicks
   outclicks = clicks*(1000/samplerate);
   clicks = 0;

   // Save RPM to memory
   if (stored_actual < 1000) {
      if (start_record == 1) {
            ram_ptr[stored_actual] = outclicks;
            stored_actual = stored_actual + 1;
      }
   }

   // PID Control
   error = reference - outclicks;
   integral = integral + (Ki * error * (samplerate / 1000)); // convert ms to seconds
   derivative = (error - error_prior) / (samplerate / 1000);
   error_prior = error; // save for next D term
   output = (Kp * error) + integral + (Kd * derivative) + ZEROOUT; // in volts

   // Bounded output
   if (output > OUTPUT_MAX) output = OUTPUT_MAX;
   if (output < OUTPUT_MIN) output = OUTPUT_MIN;

   // Convert to counts and output via DAC
   output = output*(2048/10) + 2048;
   d_to_a(0, output);
}

// External interrupt handler
void inthand(void){
   *p = 2; // reset pending register
   clicks=clicks+1;
}
```