MAE 443/543 Continuous Control Systems

Final Lab Report

Ben Cutri

Prof.: Dr. Tarunraj Singh

TA: Adrian Stein

Fall 2021

Introduction

The purpose of this report and the labs is to introduce students to the basic concepts of designing and utilizing continuous control systems. First, an introductory lab was run to familiarize students with using LabVIEW software, which can model and simulate control systems using VI’s (virtual instruments). Then, several more labs were run to simulate an electric motor, compare that simulation to the data acquired from an actual motor, and design a control system capable of modifying the system's output to meet some specified design criteria. Several concepts introduced in class were made of use in this lab report included the design of a system transfer function to represent the motor, and coding in MATLAB software to design a PD controller that was then implemented into LabVIEW to modify the motor simulation.

Summary of Experiments

Lab 1:

The first lab was intended to introduce users to using LabVIEW and creating basic VI's (virtual instruments). The lab explained how to open the LabVIEW program, what the main windows are (front panel and block diagram), and introduced the concept of palettes. Then, the lab gave three basic assignments to start working with the program and actually start using the functions and tools offered in the program. These assignments are detailed below:

Assignment 1:

The goal of this assignment was to create a VI capable of performing basic mathematics like addition, multiplication, etc. It introduced several tools from the controls and functions palette including numeric controls/indicators, boolean operations, and

programming structures. The block diagram can be seen in **Figure 1** below:

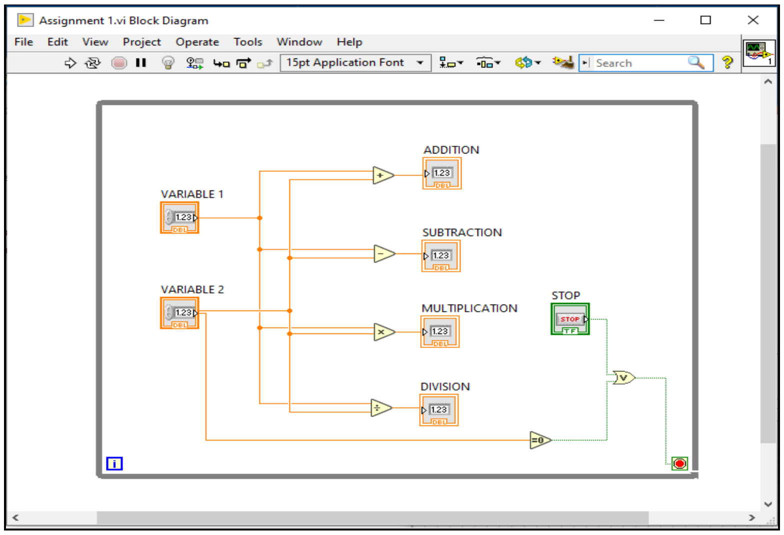


Figure 1: Block Diagram of Lab 1 - Assignment 1

Assignment 2:

The goal of this assignment was to create a random number generator with an oscilloscope-like display. It introduced waveform charts, knob controls, and how to "wire" time delays into code. The block diagram can be seen in **Figure 2** below:

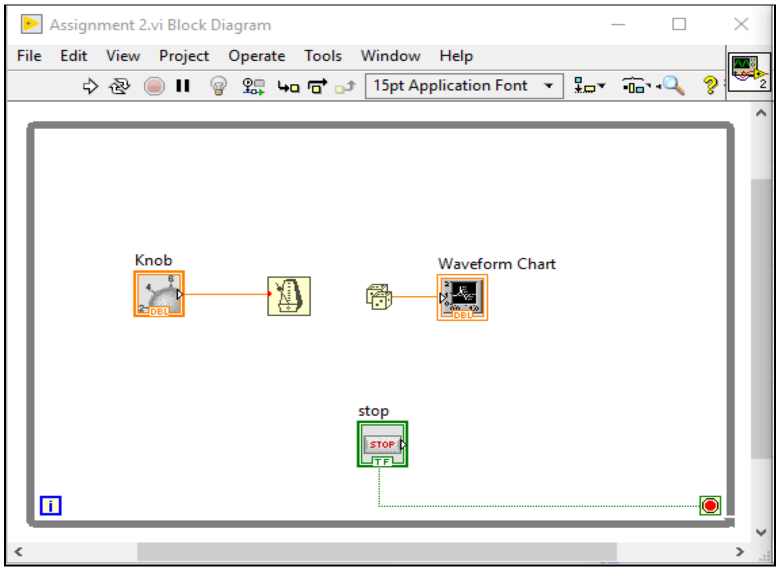


Figure 2: Block Diagram of Lab 1 - Assignment 2

Assignment 3:

This assignment's goal was to introduce the concept of SubVI's and develop a VI that lets a user choose between different types of signals (square, triangle, and sinusoidal) and specify the amplitude, frequency, and sampling time interval. Based on the information provided by the user, the subVI should generate the desired signal and display it on a chart on the panel of the main VI. The block diagram is seen in **Figure 3** below:

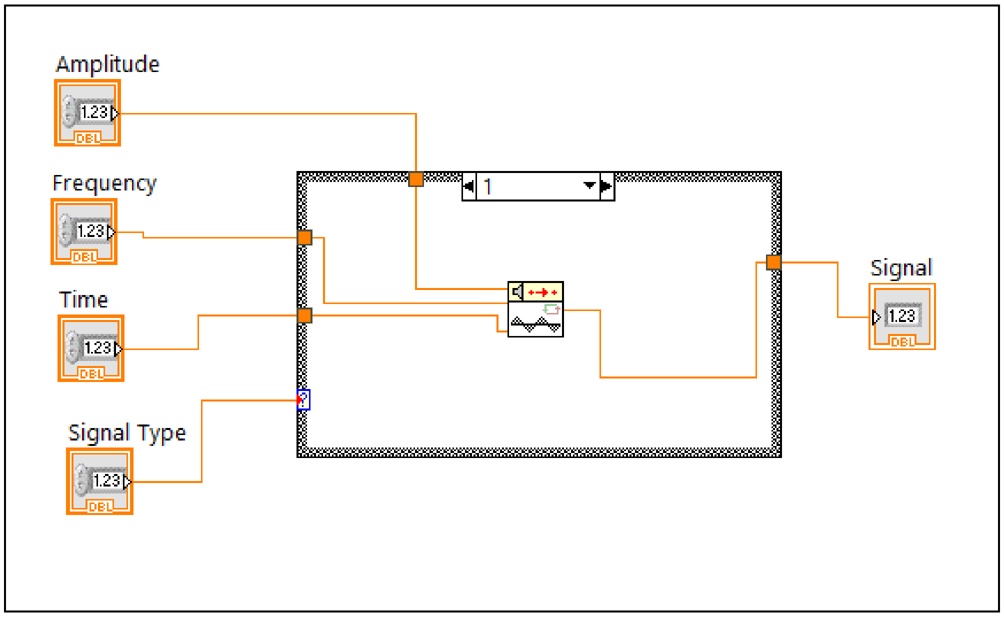


Figure 3: Block Diagram of Lab 1 - Assignment 3

Lab 2:

The second lab focused on creating a model in LabVIEW that could simulate an electric motor. First, the lab introduces several formulas and diagrams that were intended to create a thorough understanding of the underlying mechanics. Then, once the electrical and mechanical equations and the transfer function for the system were realized, a main and sub-VI were created with those system parameters to fully represent the motor numerically. **Figure 4** below shows the block diagram of the MainVI for this lab:

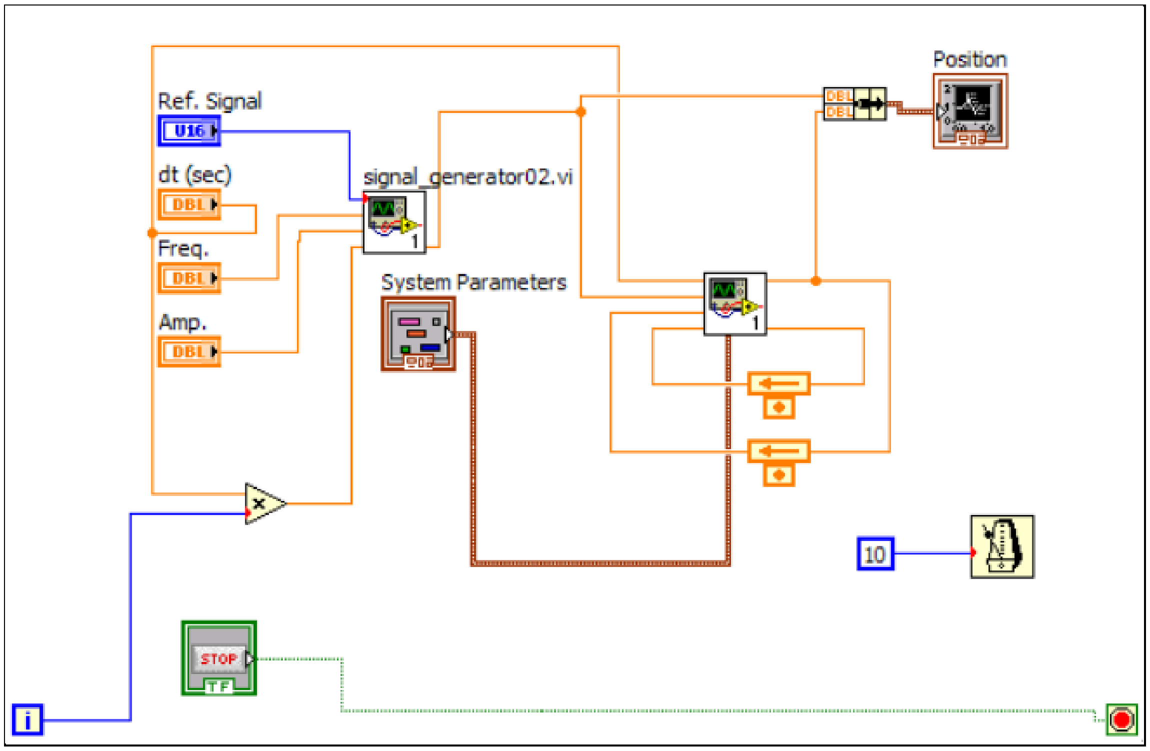


Figure 4: Block Diagram of Lab 2 - MainVI

The main VI allows a user to input several control items including signal type, sampling time, signaling frequency, signal amplitude, and the four system parameters that were used to represent the motor. And because of the large number of given control items, data clusters in LabVIEW were utilized to easily access the data outputs. These were useful for the construction of the sub VI, as shown in **Figure 5** below:

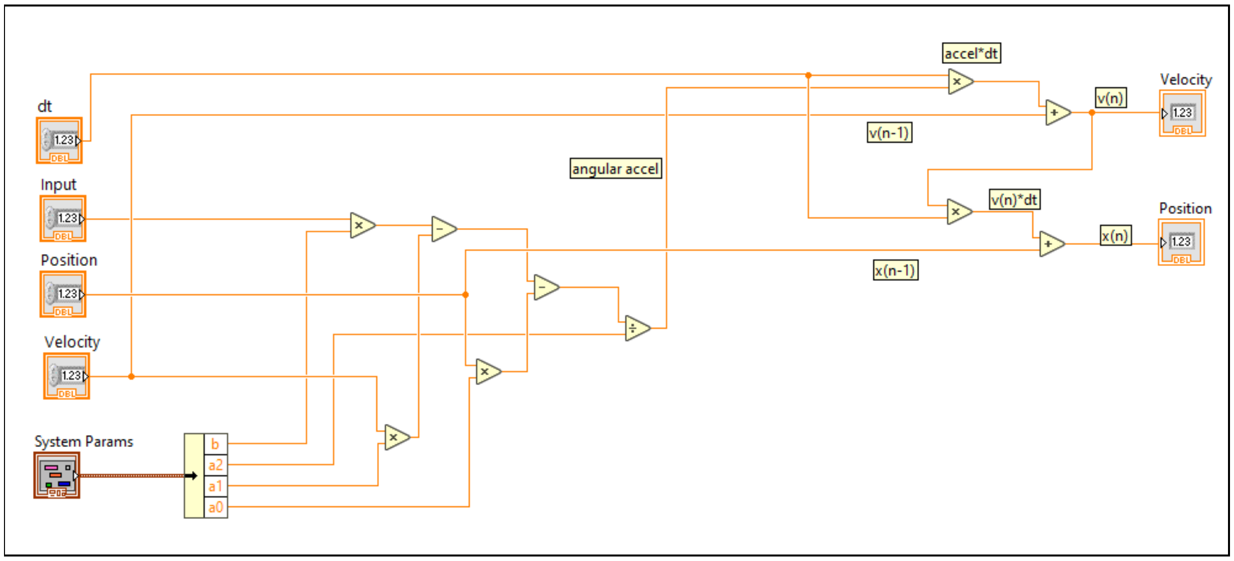


Figure 5: Block Diagram of Lab 2 - subVI

Lab 3:

The main goal of this experiment was to use LabVIEW to input a known signal into a plant and acquire the outputs. The specific plant considered was the SRV02 DC motor unit (a standard servo plant instrumented with a continuous turn potentiometer to measure output/load angular position). This plant was connected to a PC for data gathering, as well as a power supply to use the motor. The power module used in this experiment was a UPM 2405/1503. The UPM (Universal Power Module) is a power amplifier that is required to drive the motor. Using the previously created subVI's in past labs as well as the provided subVI's, **Figure 6** below is the final constructed block diagram for the MainVI:

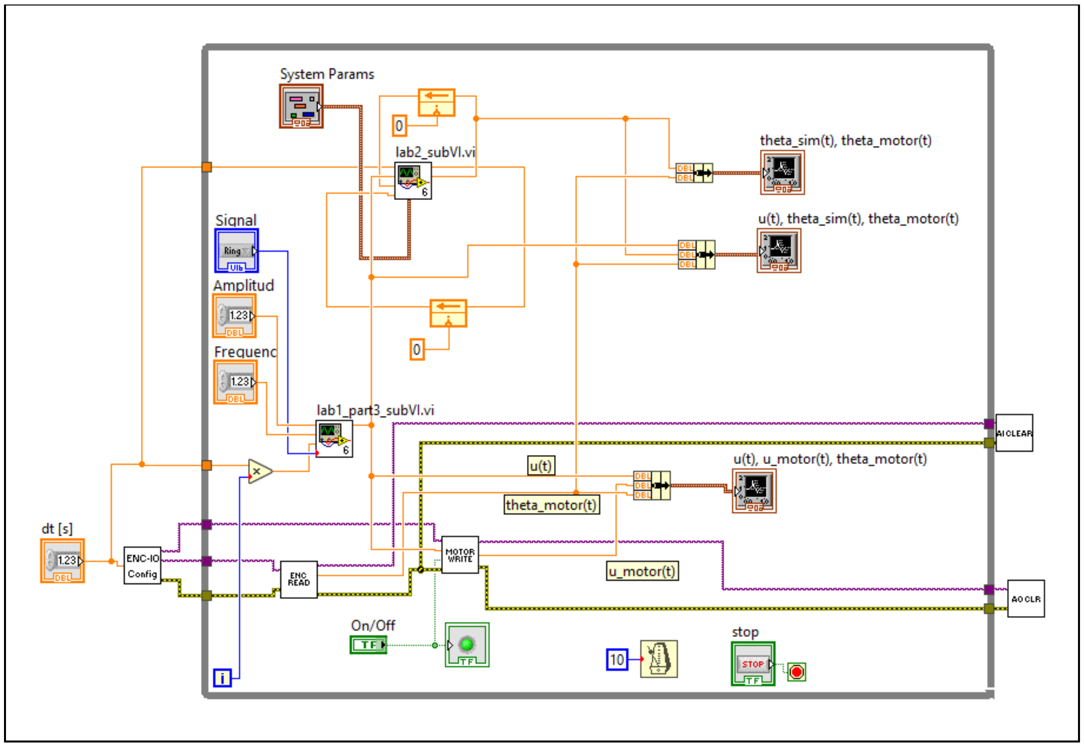


Figure 6: Block Diagram of Lab 3 - MainVI

The MainVI receives a signal, amplitude, and frequency from the user. Then, using the lab 1 and lab2 subVI's as well as the provided/calculated system parameters, the MainVI outputs and graphs various waveforms for the servo. These waveforms are a square, triangle, and sinusoidal output comparing the simulated results of the MainVI to the actual servo.

Lab 4:

The main goal of this experiment was to introduce the fundamentals of controls using a PD controller. In previous experiments, the model of the servo plant was mathematically created; and now, utilizing different tuning parameters of the controller, a PD controller was designed and simulated to meet the required specifications. The controller was evaluated for performance and compared to the "actual" servo motion and data, and tuned to be in line with the real world. **Figure 7** below is the final constructed block diagram for the MainVI:

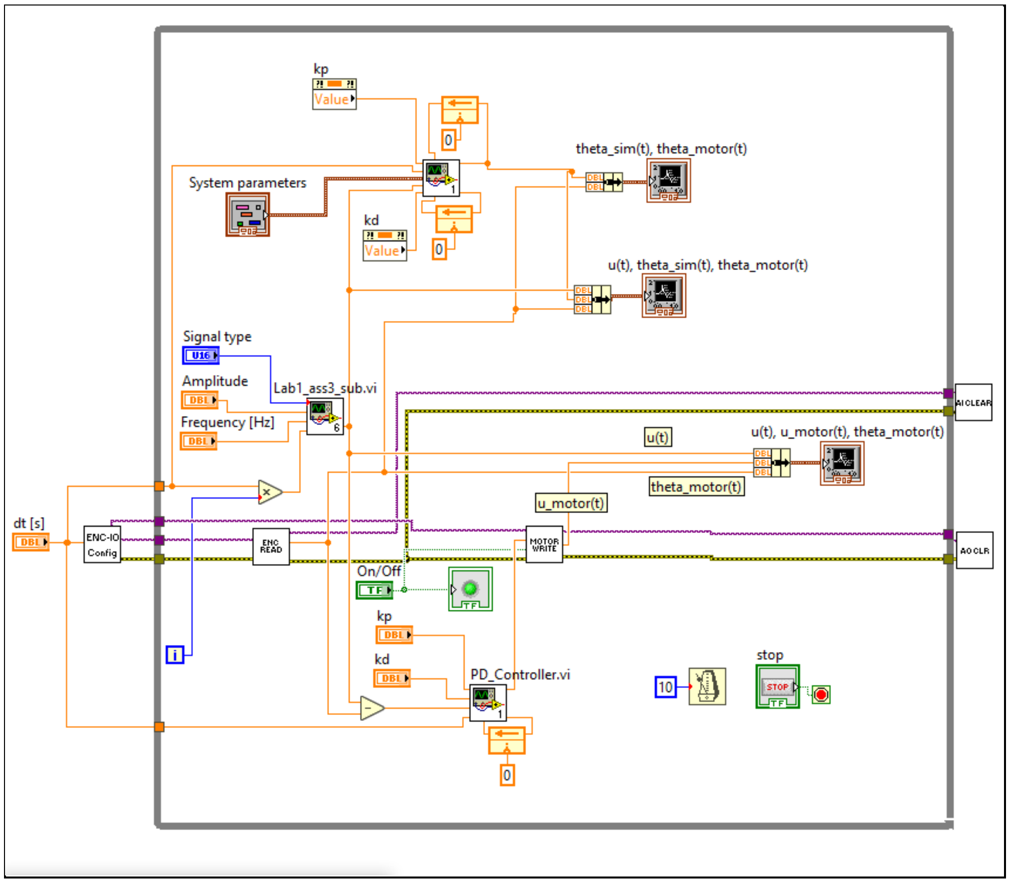


Figure 7: Block Diagram of Lab 4 - MainVI

Similar to **Figure 6,** the block diagram simulates the square, triangle, and sinusoidal waveform outputs and compares them to the actual servo. But controller gains were tweaked inside the program to match desired system performance. Additionally, the calculation of the closed-loop natural frequency and damping ratio (using model parameters and controller gains) was incorporated into the system.

Controller Design Procedure

Lab 3:

In designing the controller for the motor, several sub VI's were needed including subVI's for the IO configuration, encoder, motor, and AI/AO. Using these sub VI's as well as the ones from lab one (**Figure 3**) and lab two (**Figure 5**), a main VI was created to develop and run data acquisition for the physical motor. All of the provided subVI's are explained below in both their individual function as well as how they contribute and help create the mainVI.

ENC-IO-Config.vi:

This subVI specifies the encoder, analog input, and the analog output device configurations. It has one input, the sampling time Ts. The ENC (encoder) task output from this subVI will be the input task into the SRV02-Encoder-Read subVI and the AO (analog out) task output from this subVI will be the input task into the SRV02-Motor-Write-NI subVI.

SRV02-Encoder-Read.vi:

This subVI acquires the motor shaft position information outputted by the encoder. Inputs to this are the ENC task and the error message outputted by the ENC-IOConfig subVI. The outputs of this subVI are the task, the error message and the shaft position.

SRV02-Motor-Write-NI.vi:

SRV02-Motor-Write-NI is the subVI that inputs the periodic signal into the motor. The inputs to this subVI are the AO task output from the ENC-IO-Config subVI, the error output from the SRV02-Encoder-Read subVI, and the periodic signal and a boolean command which will let the user decide whether the motor should be on or off. The outputs of this subVI consist of the AO task, the error message, and the input signal into the motor.

AI-Clear-MX.vi and AO-Clear-MX.vi

These two subVIs are used to clear the data acquisition tasks and the error messages at the end of each simulation. The ENC task and the error message from the ENC-IO-Config.vi are the inputs into the AI-Clear-MX subVI. The inputs into the AO-Clear-MX subVI are the task out and the error message from the SRV02-Motor-Write-NI.vi.

Lab 4:

From lab 2, the transfer function for relating the load shaft position to the armature input voltage is given below in **Equation 1:**

$$\frac{\theta_l(s)}{V_m(s)} = \frac{\eta_g \eta_m K_t K_g}{J_{\text{eq}} R_m s^2 + (B_{\text{eq}} R_m + \eta_g \eta_m K_t K_g^2 K_m)s} \tag{1}$$

Where Jeq = Jl+ηgJmK2 g and for a complete listing of the system parameters, refer to ***Table 1.***

Symbol	Description	Nominal Value SI units
B_{eq}	Viscous damping coefficient	4.0×10^{-3}
η_g	Gearbox efficiency	0.9
η_m	Motor efficiency	0.69
$J_{eq} = J_l + \eta_g J_m K_g^2$	Moment of inertia at the load	2.0×10^{-3}
J_m	Motor moment of inertia	3.87×10^{-7}
K_g	Gear ratio	$70(14 \times 5)$
K_m	Back-emf constant	7.67×10^{-3}
K_t	Motor-torque constant	7.67×10^{-3}
R_m	Armature resistance	2.6

Table 1: Motor Parameters

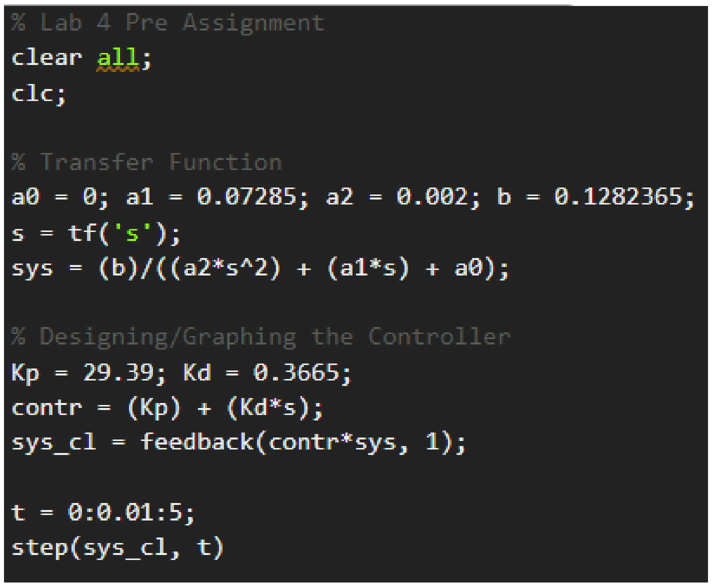
According to the design specifications, proportional and derivative gains (for the PD controller) needed to be found that can satisfy the following design specifications:

- $M_p \leq 5\%$
- $T_p \approx 0.10$ sec

These proportional and derivative gains were found using MATLAB Simulink (Kp and Kd respectively). Gains that satisfy the mentioned requirements were found to be:

- Kp = 29.39
- Kd = 0.3665

Shown below in **Figures 8 and 9** is the MATLAB code and graph of the transfer function:



```
% Lab 4 Pre Assignment
clear all;
clc;

% Transfer Function
a0 = 0; a1 = 0.07285; a2 = 0.002; b = 0.1282365;
s = tf('s');
sys = (b)/((a2*s^2) + (a1*s) + a0);

% Designing/Graphing the Controller
Kp = 29.39; Kd = 0.3665;
contr = (Kp) + (Kd*s);
sys_cl = feedback(contr*sys, 1);

t = 0:0.01:5;
step(sys_cl, t)
```

Figure 8: MATLAB Code for Lab 4 Pre-lab Assignment

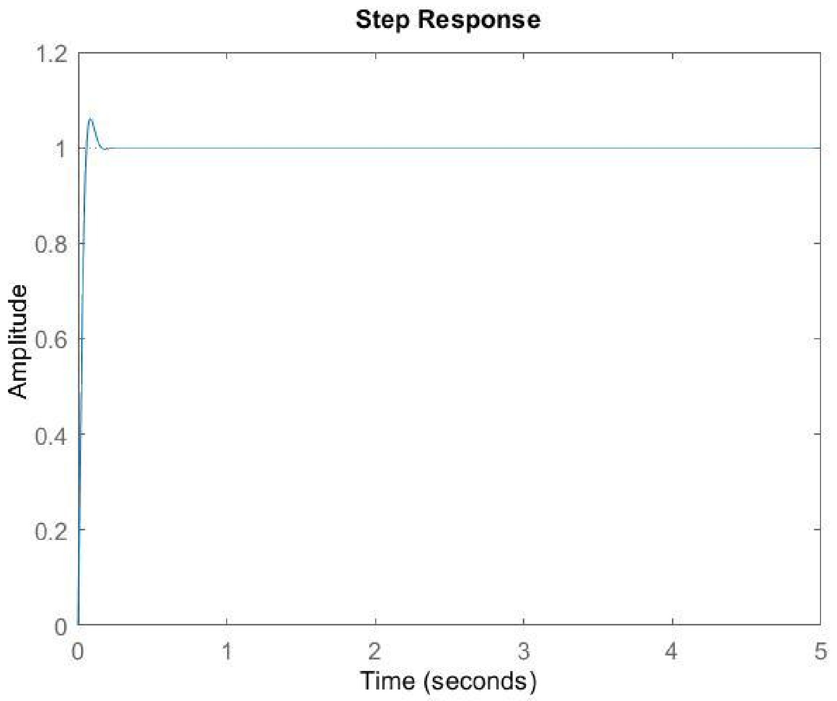


Figure 9: Graph of Step Function for Lab 4 Pre-lab Assignment

These gains were calculated using MATLAB's Automatic PID Tuning (the pidtuner function). And as is shown in the graph, the PD controller successfully meets the system criteria and can now be applied to our LabVIEW model.

Controller Implementation

To implement the designed controller, a filter was utilized (instead of a PD block in LabVIEW) to eliminate high frequencies from damaging the motor. The cut-off frequency was specified to be 20 for this controller and several subVI's were utilized including the Low Pass filter.vi, Lab 1 assignment 3.vi, and the Lab 2 motor simulation.bi (with the integrator). The lab 2 motor simulation was changed to include the Low Pass filter and controller gains. Shown below in **Figure 10** is the PD controller as implemented into the motor simulation (with a closed loop for the real position of the motor shaft):

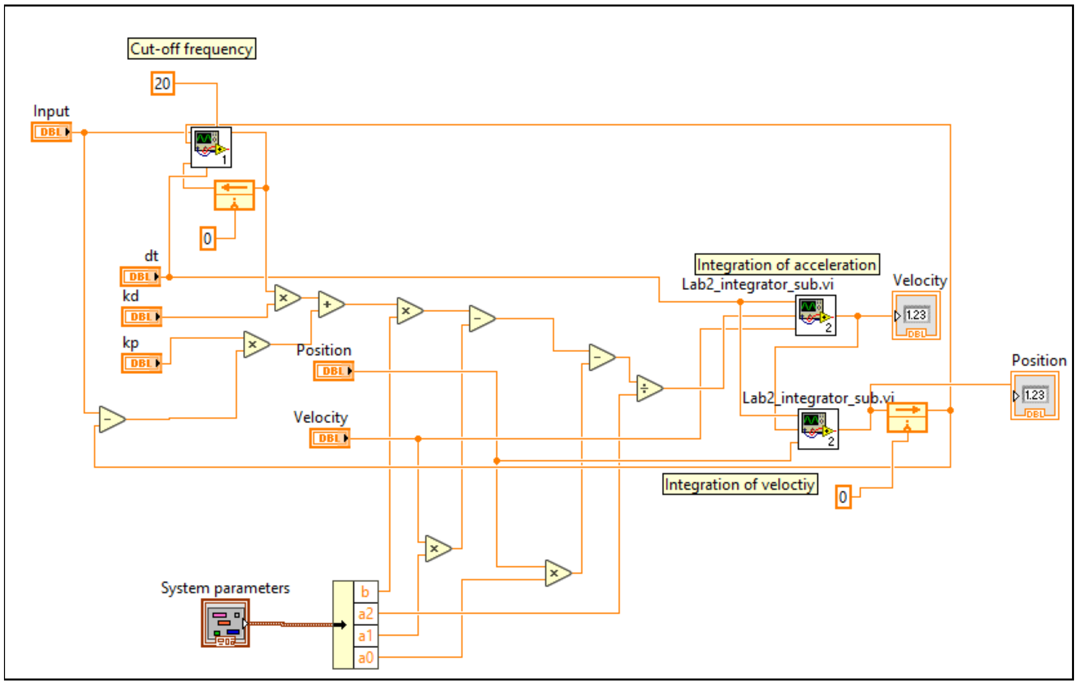


Figure 10: PD Controller as Implemented into the Motor Simulation with a Closed Loop

Once the controller was implemented and designed, as mentioned previously, the gains were tweaked so that the system response yielded the design requirements. Calculations for the natural frequency and damping ratio are shown below:

$$M_p = e^{\frac{-\pi\varepsilon}{\sqrt{1-\varepsilon^2}}} = 6.26$$

Solving for damping ratio ε…

ε = 0.308438

For the given system parameters…

$$a_2 = 0.002, a_1 = 0.07285, \ a_0 = 0$$

$$2\varepsilon w_n = 0.07285 = 2(0.308438)w_n$$

Solving for natural frequency w_n…

$$w_n = 8.4677 \ rad/sec$$

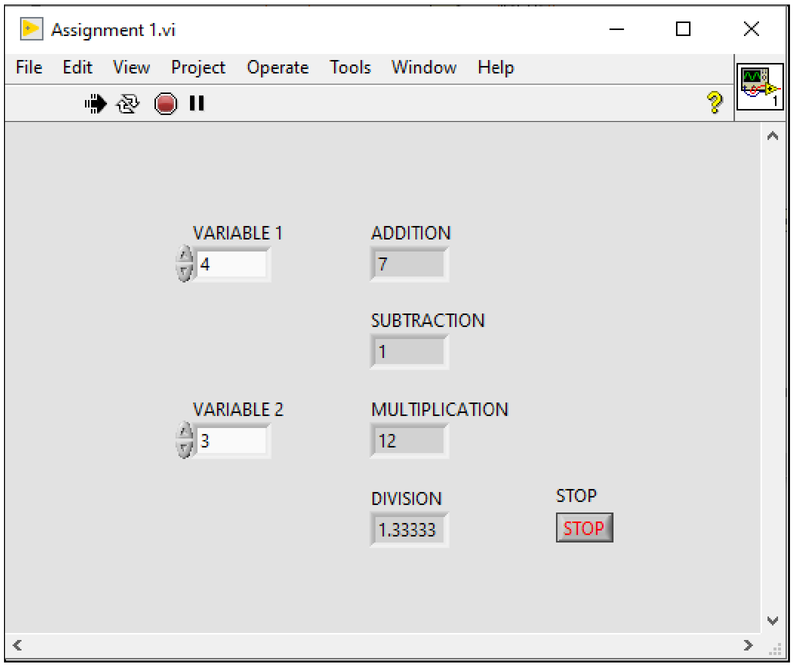
Results

Lab 1:

The results for Lab 1 are broken down into the three given assignments, each having their own front panel screenshots as described below:

Assignment 1:

The result of this assignment was a simple mathematical calculator that could add, subtract, multiply, and divide any two numbers (VARIABLE 1 with VARIABLE 2). The user can change the variables as they like, and the calculator stops running the function if they try to divide by zero. This is seen in **Figure 11** below:



VARIABLE 1: 4
VARIABLE 2: 3
ADDITION: 7
SUBTRACTION: 1
MULTIPLICATION: 12
DIVISION: 1.33333

Figure 11: Front Panel Results of Lab 1, Assignment 1

Assignment 2:

The result of this assignment was a waveform chart depicting a series of randomly generated numbers. Higher numbers are displayed as a higher amplitude on the chart and the amount of generated numbers was controlled by the knob. Each value on the knob represents a millisecond (ranging from 0ms to 300ms) and turning the knob increases the delay of the randomly generated number. This is seen in **Figure 12** below:

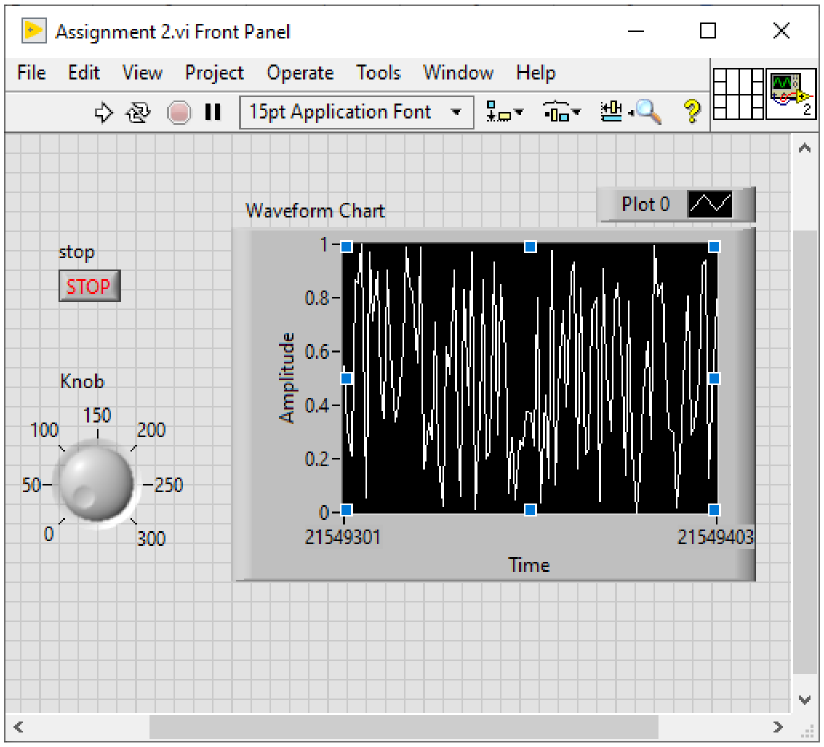


Figure 12: Front Panel Results of Lab 1, Assignment 2

Assignment 3:

The result of this assignment was a VI that allows the user to input amplitude, frequency, time, and signal type and outputs the generated wave onto a waveform chart. This is seen in **Figure 13** below:

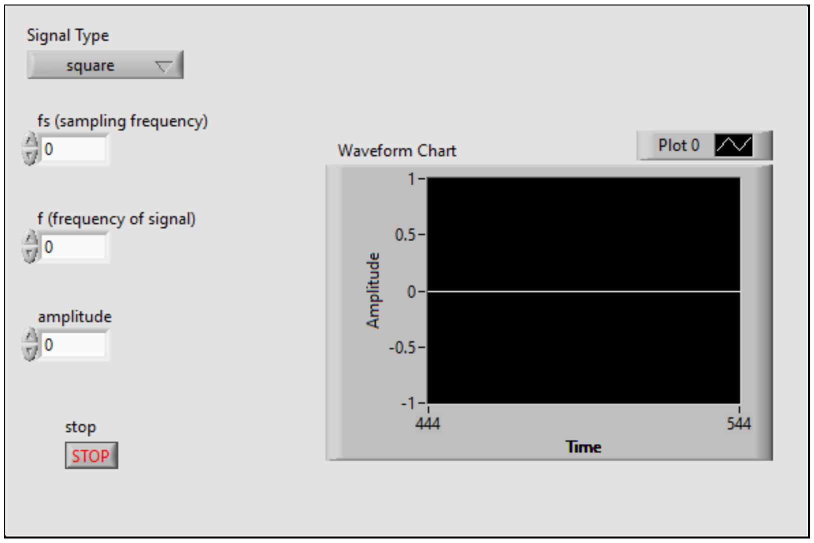


Figure 13: Front Panel Results of Lab 1, Assignment 3

Lab 2:

The result of this lab was a subVI that outputs the velocity and position, assuming zero for the initial velocity and position. Using the inputs, the subVI calculates the angular acceleration. And by integrating the acceleration, the angular position and velocity could be determined. Integration was implemented in the motor-subVI using the finite-difference method, and using a feedback node, the integrated signals are fed back into the system. These VI's are seen in **Figures 4 and 5** above, and seen below in **Figure 14** is the final resulting front panel for the motor subVI for Lab 4:

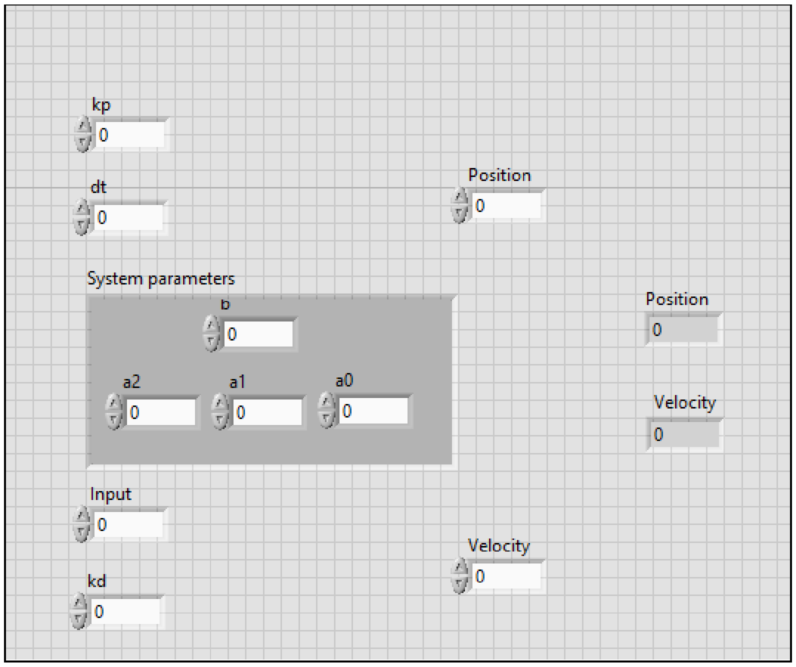
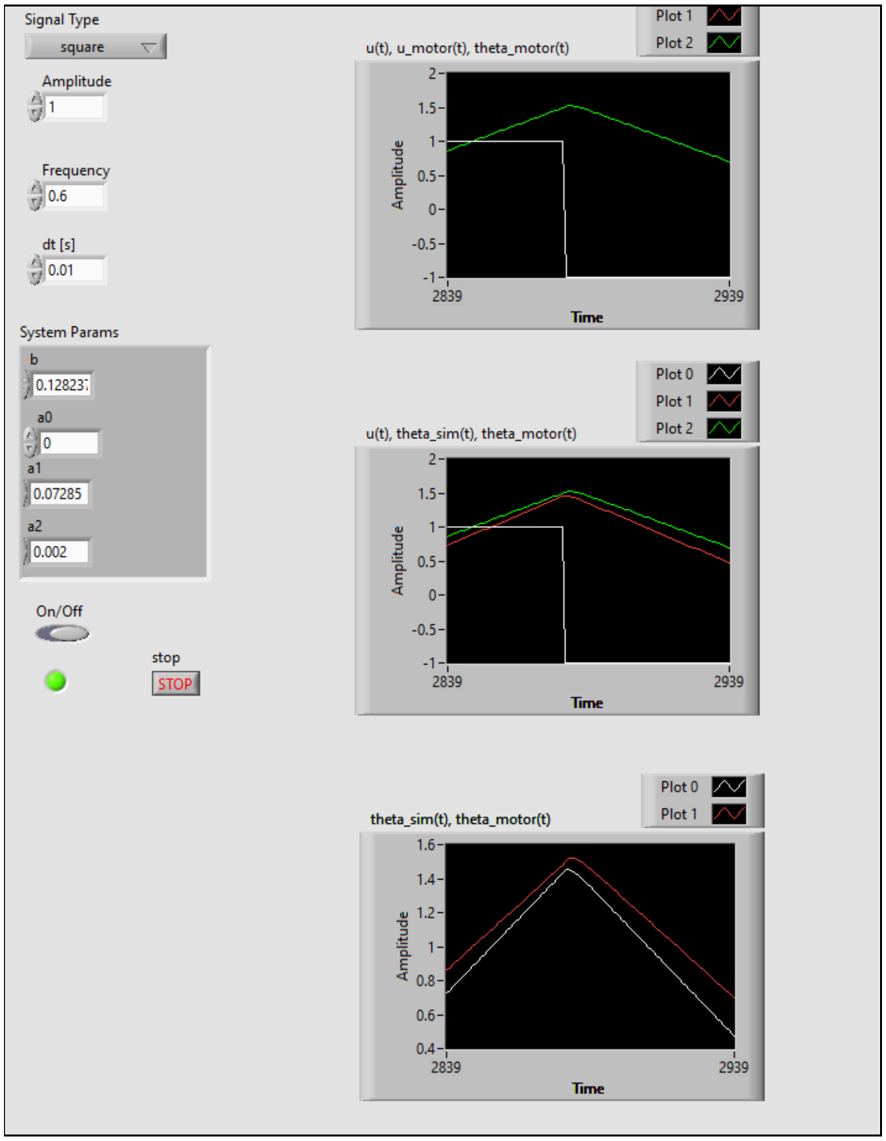


Figure 14: Front Panel Results of Lab 2 as used with Lab 4

Lab 3:

The results of this lab were a set of output graphs that show both the simulated signal and actual motor signal. The user inputs the amplitude, frequency, dt, and system parameters b, a0, a1, and a2 and the VI outputs three sets of graphs for a square, triangle, and sinusoidal wave based on the selected signal type. **Figure 15, 16, and 17** show the front panel results for lab 3 below for the mainVI with the three signal types.



Signal Type: square

Amplitude: 1

Frequency: 0.6

dt [s]: 0.01

System Params

b: 0.128237

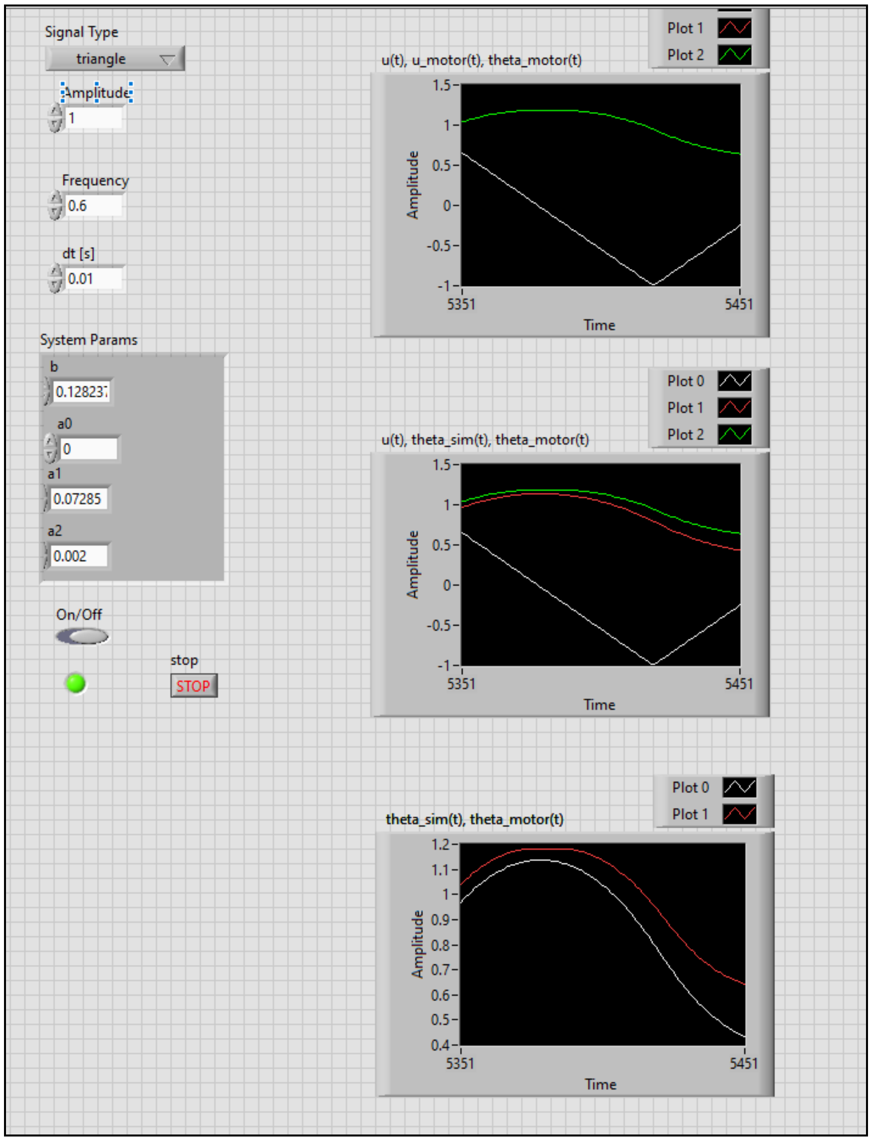
a0: 0

a1: 0.07285

a2: 0.002

On/Off
stop
STOP
u(t), u_motor(t), theta_motor(t)
Plot 1
Plot 2
u(t), theta_sim(t), theta_motor(t)
Plot 0
Plot 1
Plot 2
theta_sim(t), theta_motor(t)
Plot 0
Plot 1
Amplitude
Time
2839
2939

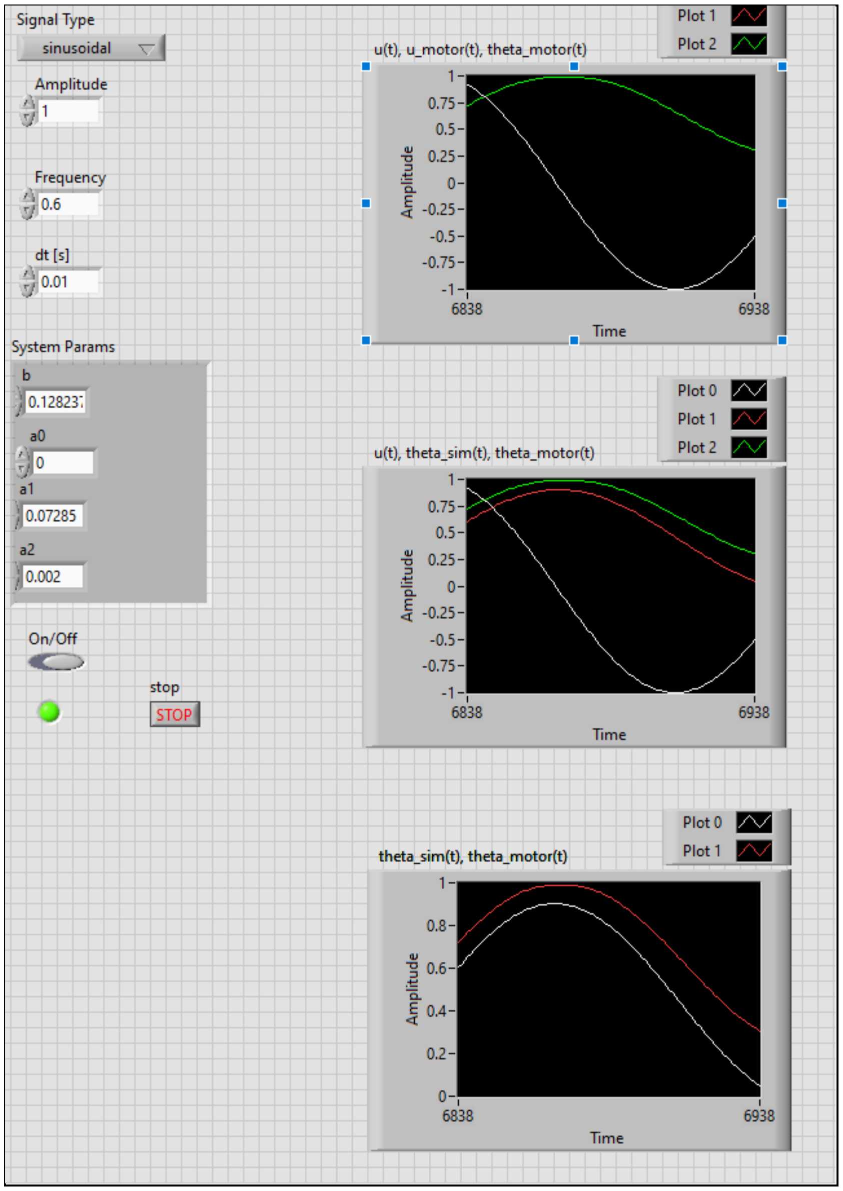
Figure 15: Front Panel Results of Lab 3 for a Square Wave


Plot 0
Plot 1
Plot 2
u(t), u_motor(t), theta_motor(t)
u(t), theta_sim(t), theta_motor(t)
theta_sim(t), theta_motor(t)
Amplitude
Time
5351
5451
stop
STOP
On/Off

Signal Type: triangle
Amplitude: 1
Frequency: 0.6
dt [s]: 0.01

System Params
b: 0.128237
a0: 0
a1: 0.07285
a2: 0.002

Figure 16: Front Panel Results of Lab 3 for a Triangle Wave


Signal Type: sinusoidal

Amplitude: 1

Frequency: 0.6

dt [s]: 0.01

System Params

b: 0.12823

a0: 0

a1: 0.07285

a2: 0.002

Figure 17: Front Panel Results of Lab 3 for a Sinusoidal Wave

Lab 4:

The results of this lab were a set of output graphs that show both the simulated signal and actual motor signal but now with the controller applied to the VI. Like in Lab 3, the user inputs the amplitude, frequency, dt, and system parameters b, a0, a1, and a2 and the VI outputs three sets of graphs for a square, triangle, and sinusoidal wave based on the selected signal type. And as mentioned in the controller design and implementation sections of this report, these values were selected to meet the design requirements of the system ($Mp \leq 5\%$ and $Tp \approx 0.10$ sec). **Figure 18, 19, and 20** show the front panel results for lab 3 below for the mainVI with the three signal types.

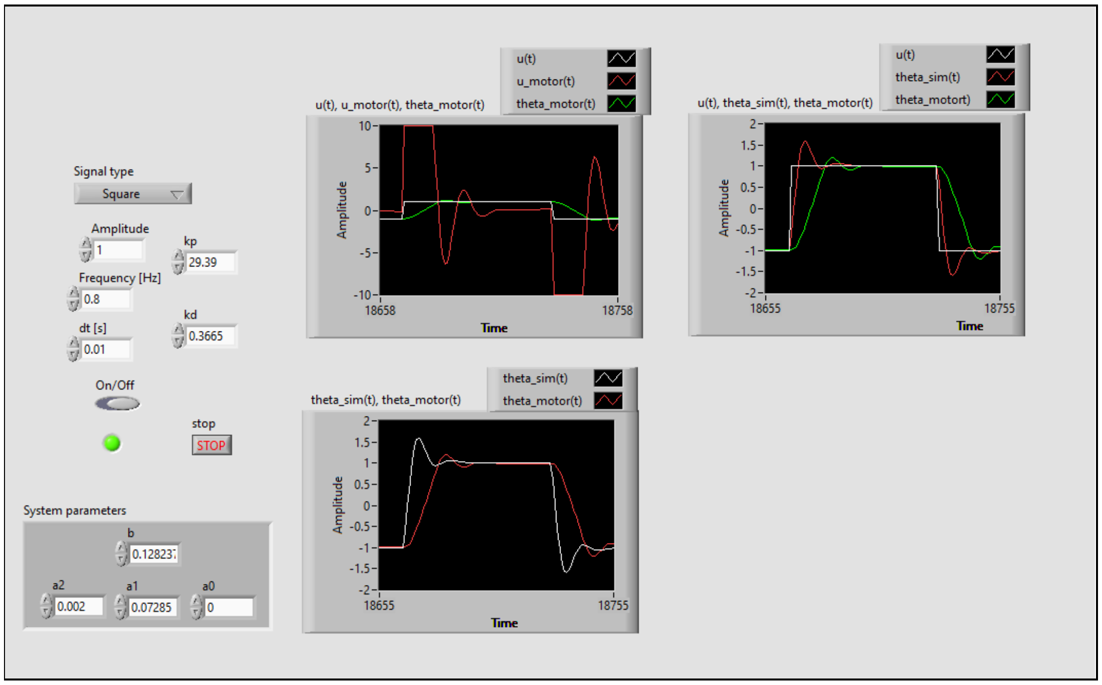
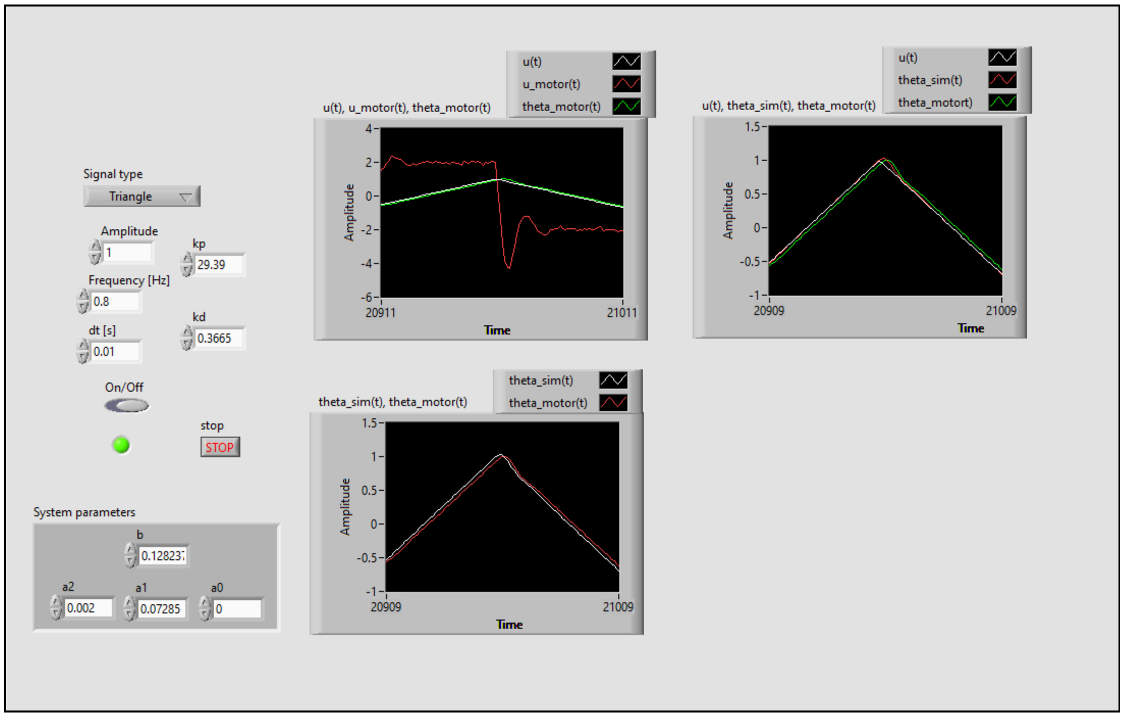

Signal type: Square
Amplitude: 1
kp: 29.39
Frequency [Hz]: 0.8
kd: 0.3665
dt [s]: 0.01
On/Off
stop: STOP
System parameters
b: 0.12823
a2: 0.002
a1: 0.07285
a0: 0
u(t), u_motor(t), theta_motor(t)
u(t)
u_motor(t)
theta_motor(t)
Amplitude
Time
18658
18758
u(t), theta_sim(t), theta_motor(t)
theta_sim(t)
theta_motort)
18655
18755
theta_sim(t), theta_motor(t)

Figure 18: Front Panel Results of Lab 3 for a Square Wave


Signal type: Triangle

Amplitude: 1

Frequency [Hz]: 0.8

dt [s]: 0.01

kp: 29.39

kd: 0.3665

On/Off

stop: STOP

System parameters

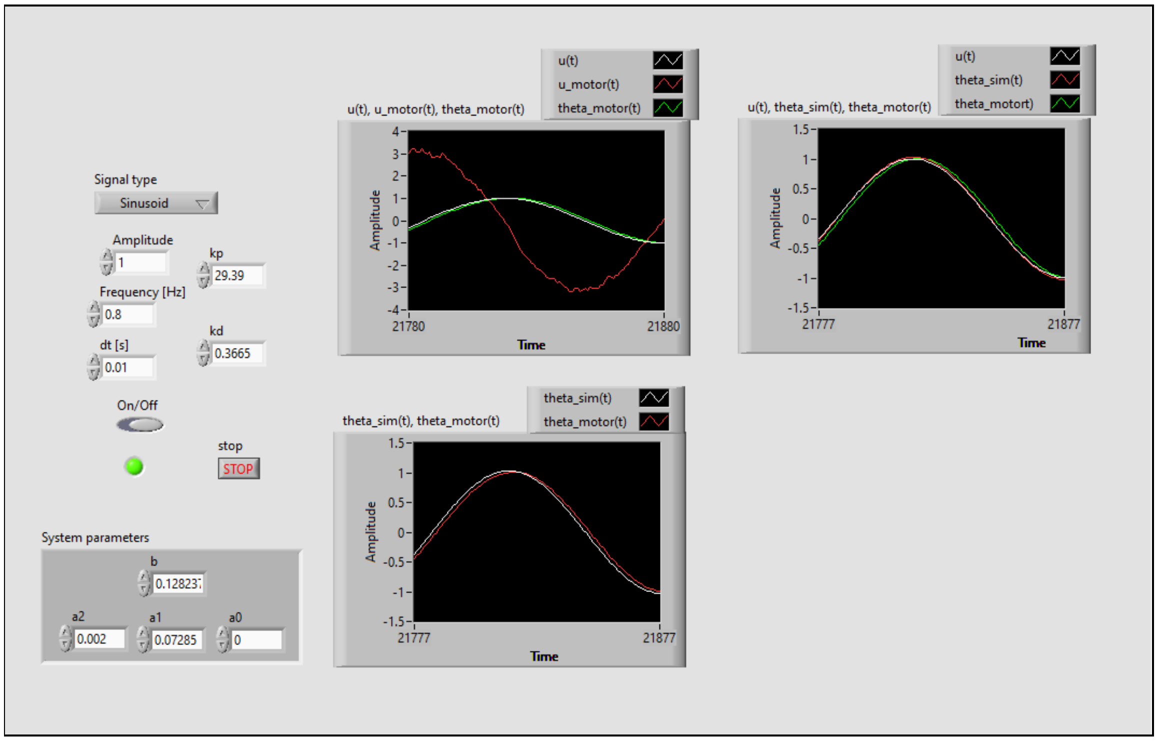
b: 0.12823

a2: 0.002

a1: 0.07285

a0: 0

Figure 19: Front Panel Results of Lab 3 for a Triangle Wave


Signal type: Sinusoid

Amplitude: 1

Frequency [Hz]: 0.8

dt [s]: 0.01

kp: 29.39

kd: 0.3665

On/Off

stop: STOP

System parameters

b: 0.12823

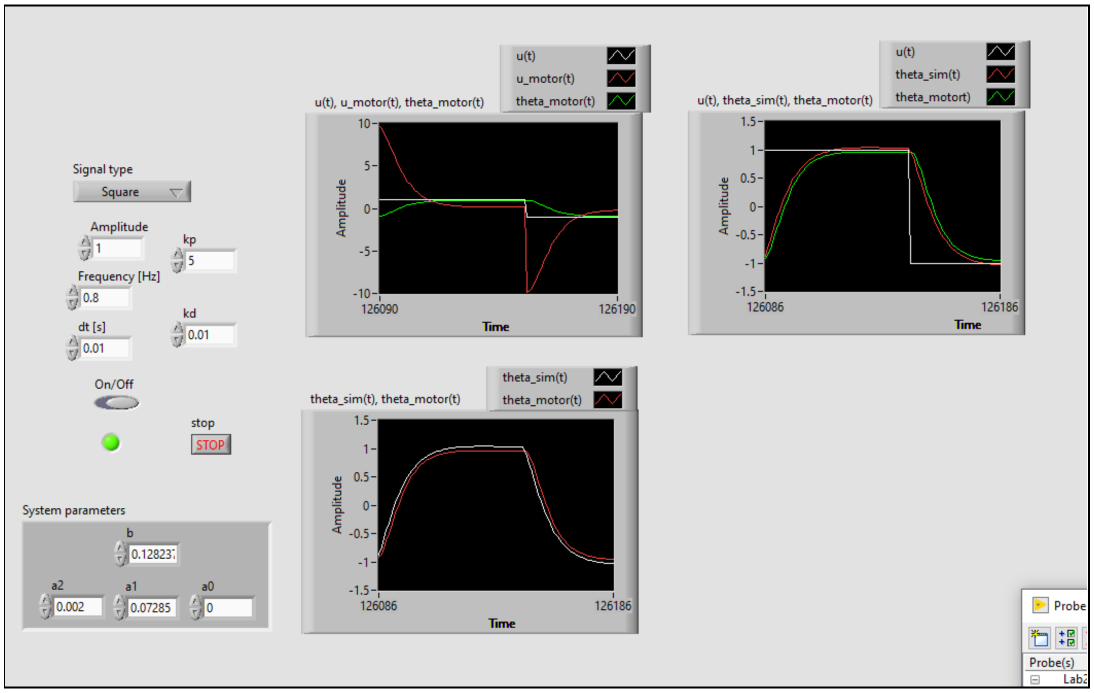
a2: 0.002

a1: 0.07285

a0: 0

Figure 20: Front Panel Results of Lab 3 for a Sinusoidal Wave

To properly simulate and ultimately represent the actual motor, the parameters needed to be tuned slightly. **Figures 21, 22, and 23** show the front panel results for lab 3 below for the mainVI with the three signal types using the tuned parameters to fully align the simulated results with the actual results:


Signal type: Square
Amplitude: 1
kp: 5
Frequency [Hz]: 0.8
kd: 0.01
dt [s]: 0.01
System parameters: b 0.128237; a2 0.002; a1 0.07285; a0 0

u(t), u_motor(t), theta_motor(t)
u(t)
u_motor(t)
theta_motor(t)
Amplitude
Time
u(t), theta_sim(t), theta_motor(t)
theta_sim(t)
theta_motort)
theta_sim(t), theta_motor(t)
On/Off
stop
STOP
Probe
Probe(s)

Figure 21: Front Panel Results of Lab 3 for a Square Wave

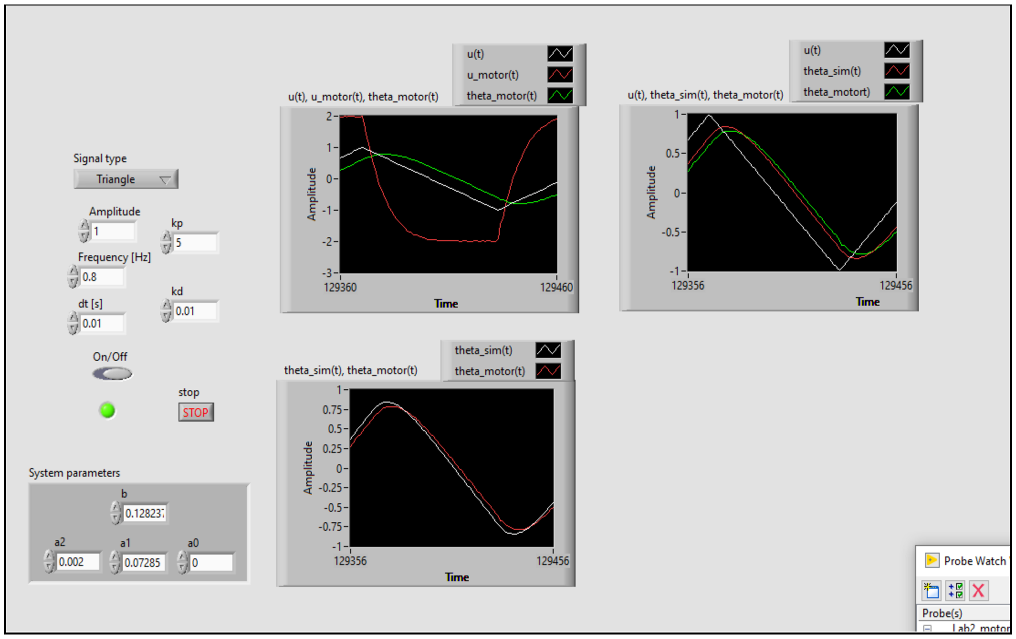
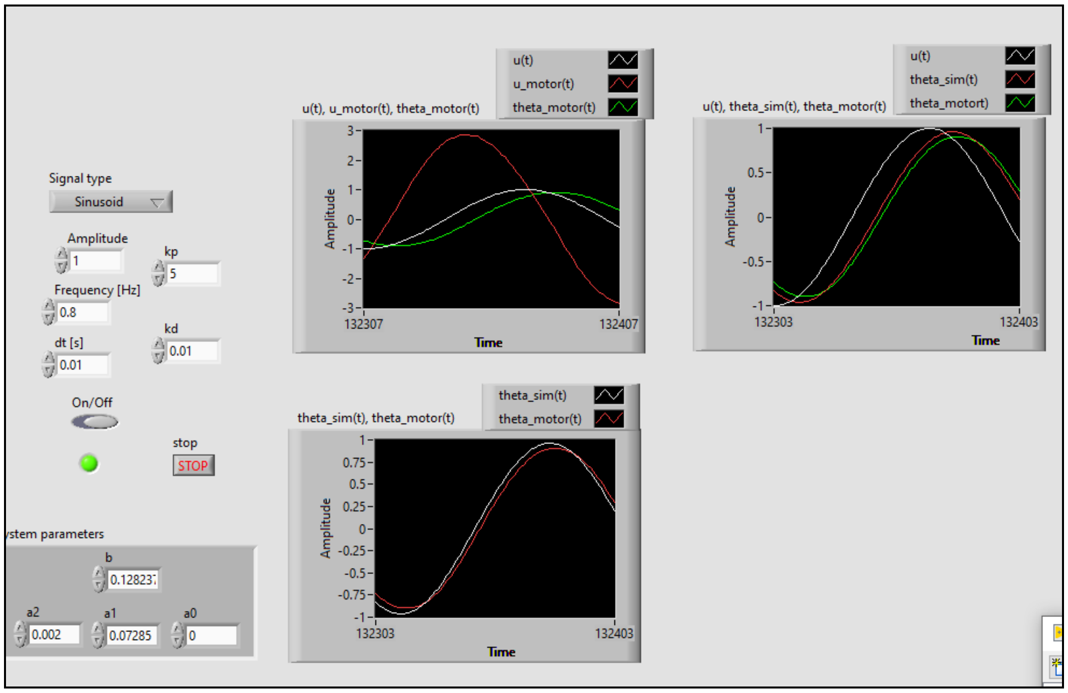

Signal type: Triangle
Amplitude: 1
kp: 5
Frequency [Hz]: 0.8
kd: 0.01
dt [s]: 0.01
System parameters: b 0.128237; a2 0.002; a1 0.07285; a0 0

Figure 22: Front Panel Results of Lab 3 for a Triangle Wave



Signal type: Sinusoid

Amplitude: 1

kp: 5

Frequency [Hz]: 0.8

kd: 0.01

dt [s]: 0.01

ystem parameters

b: 0.12823

a2: 0.002

a1: 0.07285

a0: 0

Figure 23: Front Panel Results of Lab 3 for a Sinusoidal Wave

Discussion

The MainVI created to simulate the motor worked as expected alongside the provided and designed subVI's. The MainVI output several different waveforms, and these waveforms followed their expected shapes (triangle, square, and sinusoidal). And these waveforms from the simulation matched with the actual motor using the designed controller. This designed PD controller successfully met the system requirements, and the Kp and Kd terms had a significant effect on the settling time and overshoot present in the simulated values. But as can be seen in **Figures 20-23,** the Kp and Kd values needed to be adjusted from the expected values in simulation. This is mainly due to errors within the actual motor that were not accounted for in the MATLAB simulation, including resistance in the wires and the motor's tolerance. Ultimately, the provided results show an accurate simulation of the motor with the PD controller successfully applied to the system in LabVIEW.

Questions:

After simulating your controller with your calculated Kp and Kd, did the response match what you had expected? What was the percentage overshoot and Tp ?

The response with the calculated Kp and Kd matched the expected control results, and the system reached the desired value of the step response significantly faster and more effectively with the implementation of the PD controller. The percentage overshoot was 6.26%, and the peak rising time was 0.0409 seconds, meeting the system design specifications.

After implementing your controller, did the actual system response match your simulated results? If not, what reasons could you conclude were responsible for the discrepancies?

The actual system results were slightly different than the simulated results. One potential reason for this is due to resistance in the wires that connect to the physical servo. They create a small amount of lag and reduce the amplitude of the signal. Another source of error that created a difference in the simulated and actual results was the motor tolerance. The motor likely has some error regarding the location of any given step, which would affect the output of the system.

Include your final Kp and Kd after fine-tuning the controller. Are they different from the calculated gains? If yes, why?

The final Kp and Kd values were 0.5 and 0.01 respectively. These values were different from the calculated gains from the simulation in MATLAB. This is due to not taking errors in the actual motor output into consideration (errors as mentioned above).

What are the benefits of introducing an integral term in the controller? Why did we omit the integral term in our controller?

The addition of an integral term to a controller (Ki) tends to reduce the steady-state error. If there is a persistent, steady error, the integrator builds and builds, thereby increasing the control signal and driving the error down. However, a drawback of the integral term is that it can make the system more sluggish and oscillatory since when the error signal changes sign, it may take a while for the integrator to "unwind."

During the course of these experiments, were there any problems or limitations encountered? If so, what were they and how were you able to overcome them?

The biggest problem encountered over the labs was the lack of experience debugging LabVIEW in order to correctly model the system and enact the PD controller. Even a small mistake in the wiring or implementation of the subVI’s could ruin the entire system and create weird and unexpected errors. To overcome this, our TA introduced ways to run only certain sections of the VI to limit the debugging complexity and ultimately diagnose any problems.

Conclusion

In conclusion, these labs set out to design and create a continuous control system to accurately model a real motor’s output. This was done through the design of a PD controller in LabVIEW that was implemented into a transfer function to modify the system and meet the design criteria. Ultimately, the implementation was a success and the PD controller and transfer function accurately represented the real motor’s output. However, the calculated Kp and Kd (according to the MATLAB simulation) did not match the real Kp and Kd values used. This was due to various sources of error in the motor including errors due to resistance in the wires that connect to the physical servo as well as the motors tolerance.