CAL POLY
College of Engineering

M E M O R A N D U M

From: Christopher Ng
Patrick Michael
Mechanical Engineering Department
ME 305-06

Date: 12/14/2023

To: Marius Tali
Espinoza-Wade

Subject: **DC Motor Control**

Overview

The lab setup as seen in Figure 1 contained a 24V battery supply, a PWM to regulate the voltage provided, an encoder and a motor. The goal of this lab was testing and determining relevant motor and sensor parameters like K_{PWM}, $K_c\Delta t$, τ_m etc. All these values have been used to test the performance of the proportional-plus-integral (PI) controller once set to run with certain characteristics. We conducted 2 different tests, a closed loop test and an open loop test to compare values. An open loop test does not receive feedback from the motor which forces the system to apply only the known inputs to the motor. This is good to help us get a finer understanding of the basic behavior of the motor. A closed loop test on the other hand allows exchange of feedback between the motor and the control system. We do both these tests to get an understanding of the performance/ accuracy of the PI controller by measuring how differently these two tests respond in ways like measuring the time constant and the output voltage or V_{ACT}.

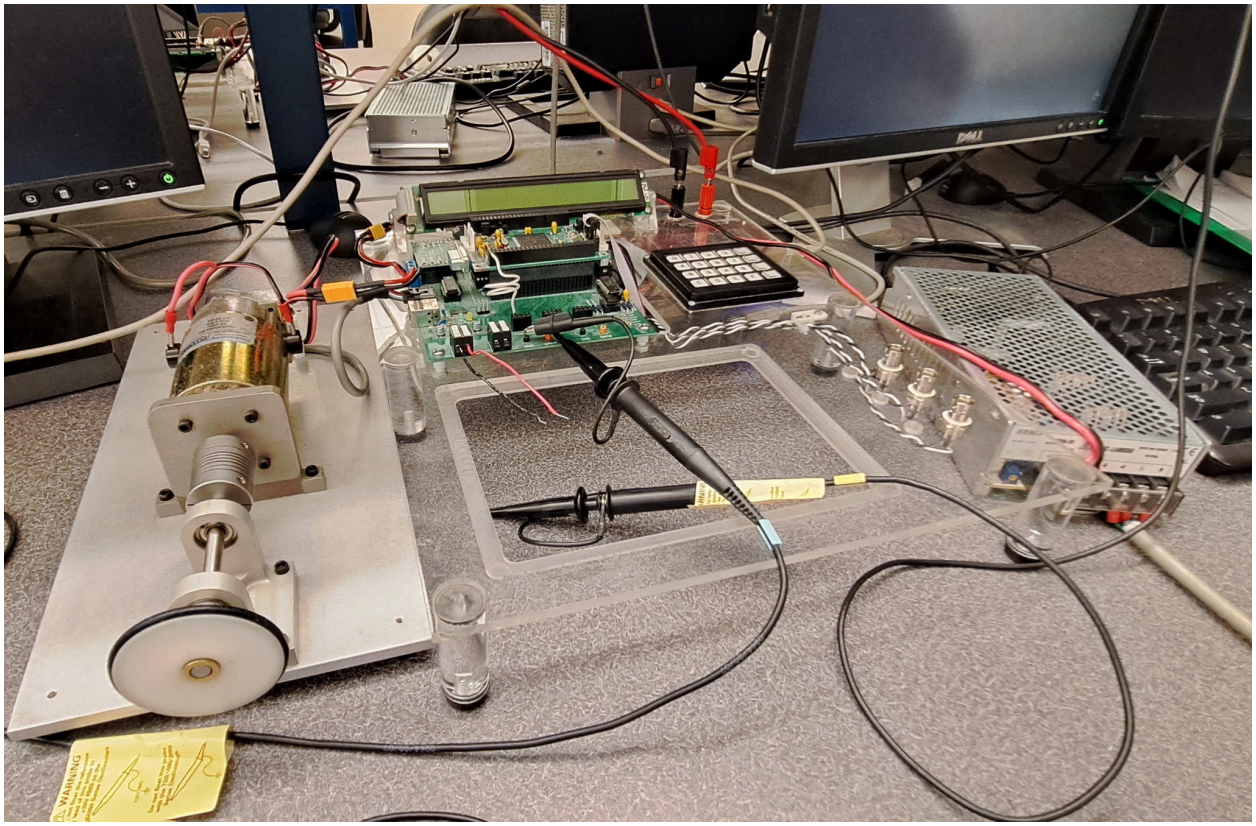

Figure 1. DC Motor Experimental Setup

Motor Parameterization

Here we conducted two different tests, one for an open loop and the other for a closed loop. Both were at steady state and conducted in a single step manner. As the pulse is portrayed on DAC A, we use a normal trigger and we have somewhat of a logarithmic function. We use this to measure the time constant τ_m . We do this by moving the right cursor to the top most flat surface and the left cursor to the bottom most flat surface. We obtain ΔV, which we then multiply by 0.63 to get the time constant from the step response. We then move the right cursor to the 63% value and the left cursor to where we can see the logarithmic curve start to form.

Derivations and Calculations

Overall System Overview

To obtain the values for k_m and τ_m, we first break the system into its components as seen in Figure 2. From our system, an input speed, V_{REF}, is provided and enters either a closed or open loop system depending on the closed or open switch, respectively. This system is otherwise composed of a proportional and integral controller, where k_p and k_i are values that the user has access to set as multiples of 1024, a pulse width modulator to convert the controller values to a percentage of the 24V power supply, the DC motor to convert the voltage to an angular velocity, and an encoder that supplies us the ability to find the real angular velocity produced by the motor.

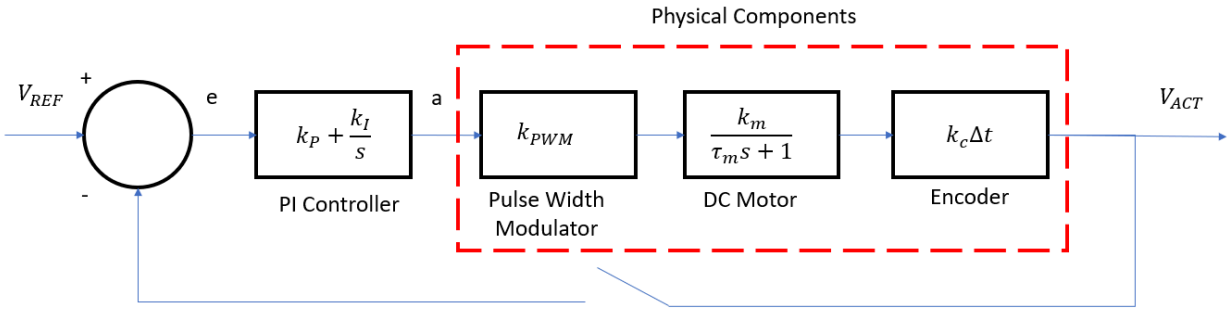

Figure 2. Overall System Block Diagram

Pulse Width Modulator

We use the pulse width modulator (PWM) to control the average voltage supplied to the motor which controls the motor's speed. This is done by manipulating the duty cycle of the pulse signal. Our source voltage, V_{REF}, provides 24V, however if our goal is to output 12V using our PWM, the duty cycle needs to be changed. We first find out what is a 100% duty cycle by dividing the clock speed which is 10MHz clock speed in our case, by the desired frequency which is 16 kHz, a range that is out of hearing range and won't create motor stutter. From this we get 625 duty counts, a value that we are required to saturate our effort to. Seen in Equation 1.1.1, dividing the reference voltage by 625 we get a pulse width modulator constant, k_{PWM}, a factor that helps us convert the duty cycle to the desired output voltage.

$$k_{PWM} = \frac{V_{REF}}{duty\ count} \tag{1.1.1}$$

DC Motor

We can then look at just the DC motor block and see how the formula in the block is derived from the physical motor system. Seen in Figure 3, an RL circuit is connected through an electrical to rotational transducer with motor constant k_v to power a dampened disk wheel. From this model, we can compose a simple first order transfer function.

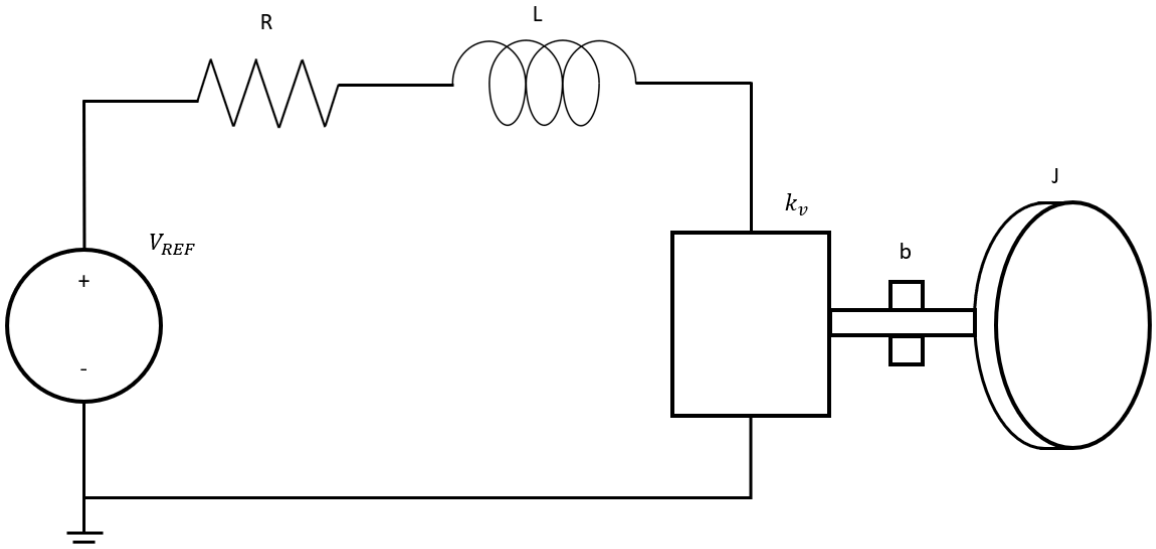

Figure 3. DC Motor Component Model

We first start by producing the elemental Equations 1.2.1 to 1.2.6 and constraint Equations 1.2.7 and 1.2.8

$$v_R = i_R R \tag{1.2.1}$$

$$i_L = \frac{1}{L}\int v_L dt \tag{1.2.2}$$

$$\tau_b = b\Omega_b \tag{1.2.3}$$

$$\Omega_J = \frac{1}{J}\int \tau_J dt \tag{1.2.4}$$

$$v_1 = k_v \Omega_2 \tag{1.2.5}$$

$$\tau_2 = -\ k_v i_1 \tag{1.2.6}$$

$$v_{REF} = v_R + v_L + v_1 \tag{1.2.7}$$

$$\tau_J = \tau_2 - \tau_b \tag{1.2.8}$$

Using these equations, we're able to derive the transfer function for the disk's angular velocity, Ω_J, now known as Ω_{ACT}, and the reference voltage, V_{REF}, as seen in Equation 1.2.9. This equation can then be simplified to Equation 1.2.10.

$$\frac{\Omega_{ACT}}{v_{REF}} = \frac{\frac{k_v}{Rb}}{(\frac{L}{R}s+1)(\frac{J}{b}s+1)+\frac{k_v^2}{Rb}} \tag{1.2.9}$$

$$\frac{\Omega_{ACT}}{v_{REF}} = \frac{\frac{k_v}{Rb}}{(\frac{L}{R}\frac{J}{b})s^2+(\frac{L}{R}+\frac{J}{b})s+\frac{Rb+k_v^2}{Rb}} \tag{1.2.10}$$

We can further simplify the equation by comparing the magnitude of the time constant in the electrical domain, *L/R*, to the time constant of the mechanical domain, *J/b*. We find that the system in the electrical domain reacts much quicker than the system in the mechanical domain, with time constants of 0.1ms and 10ms, respectively. This allows us to conclude that *L/R* is negligible compared to *J/b*. After implementing this approximation into Equation 1.2.10 and further simplifying to the canonical form for a first order system transfer function, we are provided with Equation 1.2.11 as the transfer function of the motor.

$$\frac{\Omega_{ACT}}{v_{REF}} = \frac{\frac{k_v}{Rb+k_v^2}}{(\frac{k_v J}{Rb+k_v v^2})s+1} \tag{1.2.11}$$

One more step can be taken to complete the transfer function by substituting the Equations 1.2.12 and 1.2.13 for our condensed motor parameters k_m and τ_m into Equation 1.2.11 to get our final Equation 1.2.14. Since these parameters are made of multiple unknown variables, this experiment will be used to experimentally determine them for our given motor.

$$k_m = \frac{k_v J}{Rb+k_v v^2} \tag{1.2.12}$$

$$\tau_m = \frac{k_v}{Rb+k_v^2} \tag{1.2.13}$$

$$\frac{\Omega_{ACT}}{v_{REF}} = \frac{k_m}{\tau_m s+1} \tag{1.2.14}$$

Encoder

An encoder will also be required to convert the produced angular velocity to a readable value. In the hardware provided for this lab, we have a 500 slit encoder and contains 2 sets of offset LEDs and photodetectors for counters. This makes a quadrature counter counting 4 counts per slit and gives us 2000 counts per rotation. This is then represented as the encoder constant, k_c, as 2000/2π BDI/rad. Using a Δt which is 0.002 s/BTI, we can then use Equation 1.3.1 to find the relationship between an input shaft speed, Ω, in rad/s and get a V_{ACT} in BDI/BTI. This relationship finds that the overall scaling produces a $K_c\Delta t$ of 0.637.

$$V_{ACT} = \Omega k_c \Delta t \tag{1.3.1}$$

Testing for Motor Parameters

For this experiment, two systems using the model in Figure 2 will be tested with a step input voltage: an open loop system where the switch in the figure remains open, and a closed loop system where the switch in the figure is closed. These will be used to evaluate the motor parameters shown previously in the DC motor section using measurements of the time constant, τ_m, and actual motor speed, V_{ACT}, with variable inputs of desired motor speed, V_{REF}, a proportional scaling constant, k_p, and an integral scaling constant, k_I, of zero.

To simplify the constants k_{PWM}, k_m, and $k_c\Delta t$, we will combine them into one constant, K, as seen in Equation 1.4.1 to produce a simplified block diagram in Figure 4.

$$K = k_{PWM} k_m k_c \Delta t \tag{1.4.1}$$

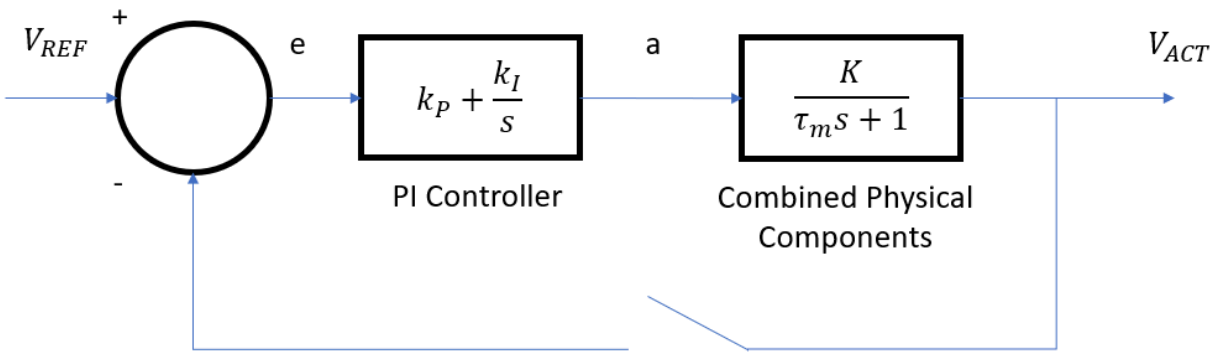

Figure 4. Simplified Overall System Block Diagram

Open Loop Test

From Figure 4, a new block diagram can be created by removing the branch with the switch and applying a k_I of zero as seen in Figure 5.

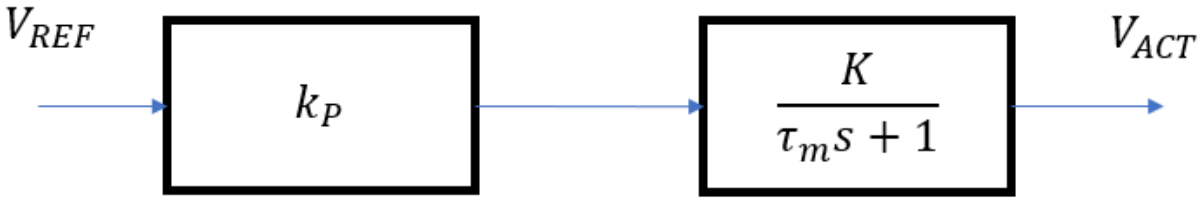

Figure 5. Open Loop System Block Diagram

For the simple block diagram, a transfer function can be easily made comparing the output V_{ACT} to the input V_{REF} as seen in Equation 1.5.1.

$$\frac{V_{ACT}}{V_{REF}} = \frac{k_p K}{\tau_m s+1} \tag{1.5.1}$$

Two tests will be required to determine both *K*, then k_m from Equation 1.4.1, and τ_m. One at steady state, as time approaches infinity, for k_m, and one during the motor's transient phase, while the motor speed is ramping up to its steady state value, for τ_m. Both of these tests will be subjected to a step input so that they reach a constant value.

In the steady state model, we will be using the definition of *s* as the frequency or *1/t*, which as time approaches infinity, will create an *s* value of zero. Our Equation 1.5.1 then becomes Equations 1.5.2 and 1.5.3 to solve for *K*.

$$\frac{V_{ACT}}{V_{REF}} = k_p K \tag{1.5.2}$$

$$K = \frac{V_{ACT}}{k_p V_{REF}} \tag{1.5.3}$$

And after incorporating Equation 1.4.1 into Equation 1.5.3, we get the final Equation 1.5.4 for k_m.

$$k_m = \frac{V_{ACT}}{k_p k_{PWM} k_c \Delta t V_{REF}} \tag{1.5.4}$$

The Equations 1.5.3 and 1.5.4 show that *K*, and subsequently k_m, are a function of the input speed, V_{REF}, output speed, V_{ACT}, input k_p, and constants, but since they are composed of the physical system's constants, should remain constant for any set of these input values.

For the transient model, we have to examine the system's response to a step input to determine τ_m. Using Laplace transformations on Equation 1.5.1, where $V_{REF}(t)$ is treated as a step input with the amplitude of V_{REF}, we retrieve equation 1.5.5 for $V_{ACT}(t)$ in the time domain.

$$V_{ACT}(t) = k_p K V_{REF}(1 - e^{-t/\tau_m}) \tag{1.5.5}$$

Plotted, this is seen to be similar to Figure 6, where for a log curve to a steady state value, one time constant can be found at 0.63 of the difference from the initial value to the steady state value.

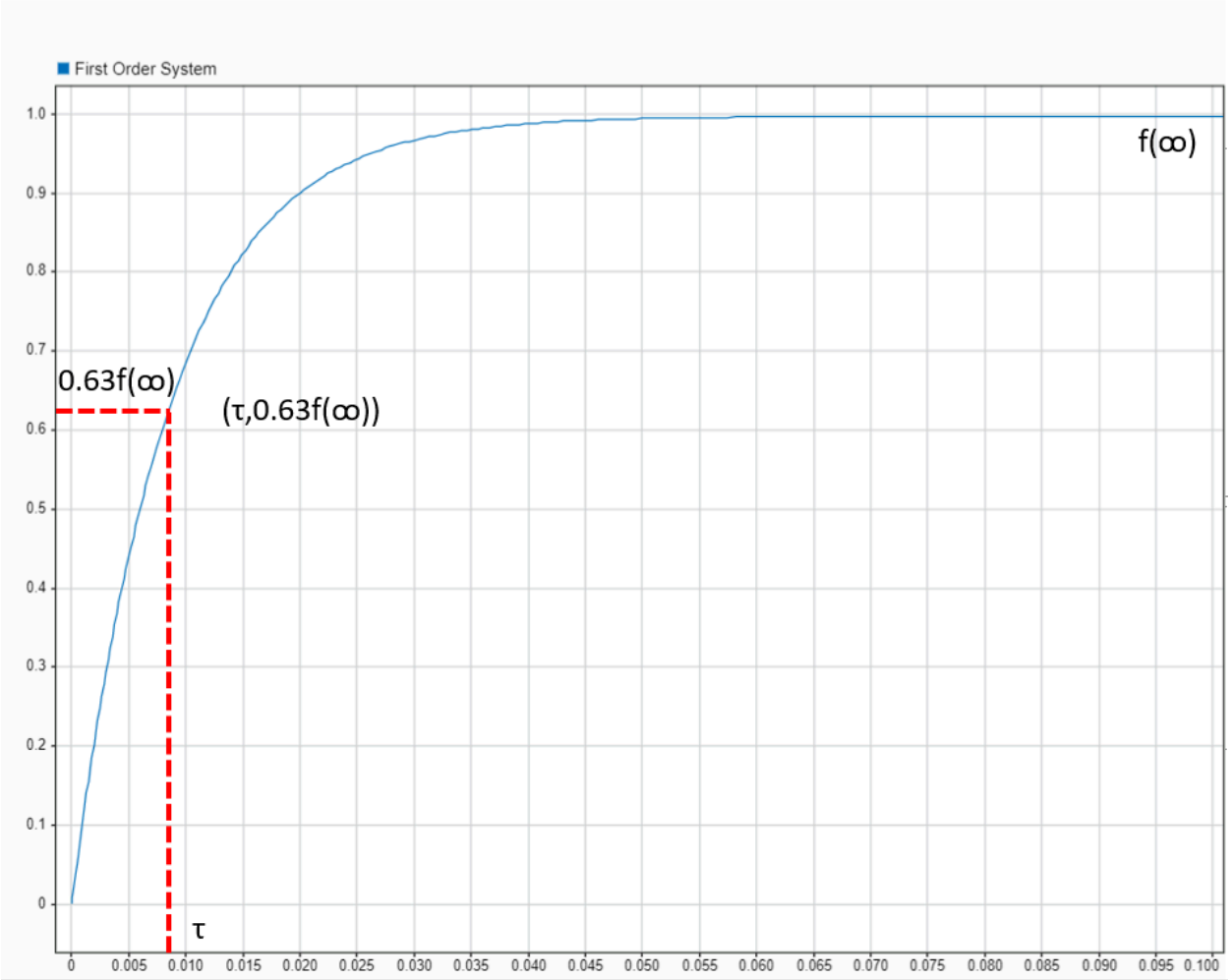

Figure 6. Log Response for a Step Input

The final values that we find for both k_m and τ_m from these methods should be found to be constant since they are derived from constants of the physical system in Equations 1.2.12 and 1.2.13.

Closed Loop Test

Similar to the open loop system, Figure 7 shows a new block diagram can be created from Figure 4 that includes the branch with the closed switch and again applying a k_I of zero.

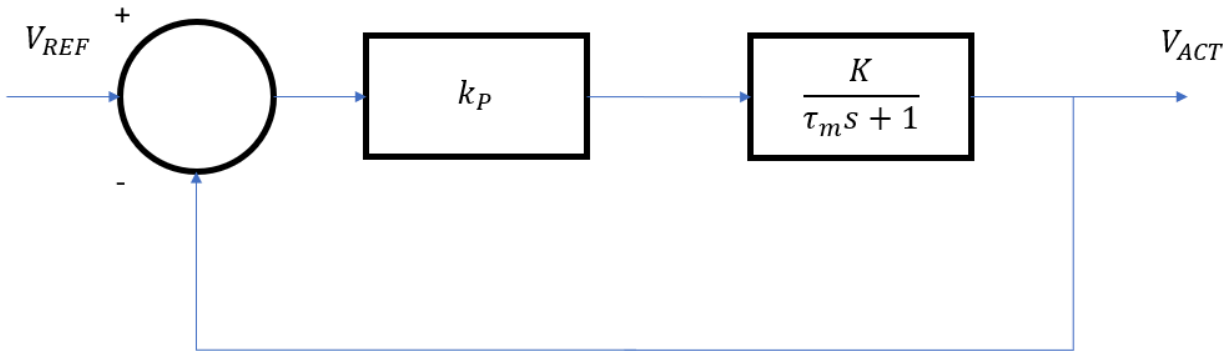

Figure 7. Closed Loop System Block Diagram

After introducing this closed loop feedback, the system becomes more complex with the transfer function no longer being derived from just multiplication blocks. Instead, this system takes the difference in V_{ACT} and V_{REF}, or error, and scales the input off of this error. To get a transfer function for this system, we'll apply a block diagram simplification seen in the closed loop model in Figure 8 that converts to an open loop model in Figure 9.

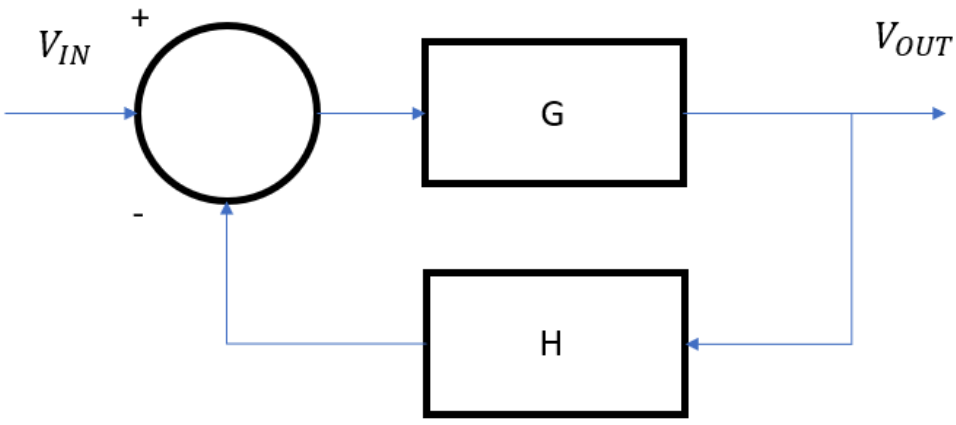

Figure 8. Example Closed Loop System

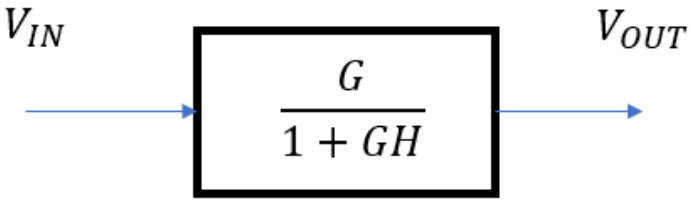

Figure 9. Example Closed Loop System Transfer Function

In our case, G includes both the proportional and integrated controller and combined physical system while H is just 1. What this produces is the block in Figure 10 which can be simplified into Equation 1.6.1.

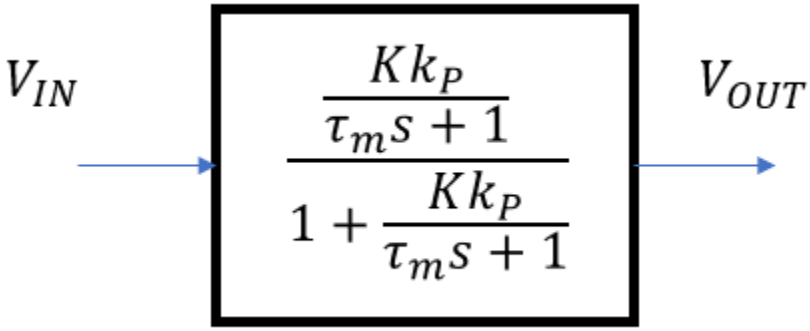

Figure 10. Simplified Closed-to-Open Loop System Block Diagram

$$\frac{V_{ACT}}{V_{REF}} = \frac{\frac{k_p K}{1+k_p K}}{\frac{\tau_m}{1+k_p K}s+1} \tag{1.6.1}$$

Once again, two tests will be required to determine both k_m and τ_m. One at steady state, as time approaches infinity, for k_m, and one during the motor's transient phase, while the motor speed is ramping up to its steady state value, for τ_m. Both of these tests will be subjected to a step input so that they reach a constant value.

After applying an *s* value of zero into Equation 1.6.1 for the steady state model, we get Equation 1.6.2 which then becomes Equations 1.6.3 and 1.6.4 to solve for *K* and k_m.

$$\frac{V_{ACT}}{V_{REF}} = \frac{k_p K}{1+k_p K} \tag{1.6.2}$$

$$K = \frac{V_{ACT}}{k_p(V_{REF} - V_{ACT})} \quad (1.6.3)$$

$$k_m = \frac{V_{ACT}}{k_p k_{PWM} k_c \Delta t(V_{REF} - V_{ACT})} \quad (1.6.4)$$

The Equations 1.6.3 and 1.6.4 vary a little from the closed loop system, but they still show that *K* and k_m are a function of the input speed, V_{REF}, output speed, V_{ACT}, input k_p, and constants and should remain constant for any set of these input values.

For the transient model, we again have to examine the system's response to a step input to determine τ_m. Using the exact same Laplace transformations on Equation 1.6.1, where $V_{REF}(t)$ is treated as a step input with the amplitude of V_{REF}, we retrieve equation 1.6.5 for $V_{ACT}(t)$ in the time domain.

$$V_{ACT}(t) = \frac{k_p K}{1+k_p K} V_{REF}(1 - e^{-t/\frac{\tau_m}{1+k_p K}}) \quad (1.6.5)$$

The key difference that we observe between Equations 1.5.5 and 1.6.5, the open and closed loop transient response, is that our time constant that we measure in the is no longer just τ_m. Instead, we must use Equation 1.6.6 to calculate τ_m from our measured time constant.

$$\tau_m = \tau_{measured}(1 + k_p K) \quad (1.6.6)$$

Procedure

The procedure we used to test the parameters of the motor started off by setting the normal trigger for the screen to populate the second an event occurs. We then ran the motor to which a transient curve appears on the oscilloscope. We turned the cursors to time mode and proceeded to measure τ_m. As the pulse is portrayed on DAC A, we use a normal trigger and we have somewhat of a logarithmic function. We use this to measure the time constant τ_m . We do this by moving the right cursor to the top most flat surface and the left cursor to the bottom most flat surface. We obtain *ΔV*, which we then multiply by 0.63 to get the time constant from the step response. We then move the right cursor to the 63% value and the left cursor to where we can see the logarithmic curve start to form. The time constant value we get from doing that is the correct value τ_m in an open loop test. However, for a closed loop test we have to used equation 1.6.6 to get the final value of τ_m .

For the comparison of the *K* and k_m we first had to use the values V_{REF} and V_{ACT} . We chose the V_{REF} for our motor and we received V_{ACT} by waiting for the motor to run at steady state and then read the value of V_{ACT} as presented on the LCD screen. After

doing that we used Equation 1.5.3 to calculate *K* in an open loop test and Equation 1.5.4 to calculate k_m also in an open loop test. In a closed loop test however we used equation 1.6.3 to calculate *K* and equation 1.6.4 to calculate k_m .

Experimental Data

Table 1. Open Loop Test V_{ref} = 200

Experimental							
V_{ref}	$1024*k_p$	k_p	τ_m (ms)	V_{ACT}	K	k_m	
200.000	500.000	0.488	11.800	19.000	0.195	7.954	
	1000.000	0.977	10.000	36.000	0.184	7.535	
	1500.000	1.465	11.300	50.000	0.171	6.977	
Average			11.033			7.489	

Table 2. Open Loop Test V_{ref} = 300

Experimental							
V_{ref}	$1024*k_p$	k_p	τ_m (ms)	V_{ACT}	K	k_m	
300.000	500.000	0.488	12.000	29.000	0.198	8.093	
	1000.000	0.977	9.300	55.000	0.188	7.675	
	1500.000	1.465	8.200	76.000	0.173	7.070	
Average			9.833			7.613	

Table 3. Open Loop Test V_{ref} = 400

Experimental							
V_{ref}	$1024*k_p$	k_p	τ_m (ms)	V_{ACT}	K	k_m	
400.000	500.000	0.488	11.200	40.000	0.205	8.373	
	1000.000	0.977	9.200	73.000	0.187	7.640	
	1500.000	1.465	8.200	101.000	0.172	7.047	
Average			9.533			7.686	

Table 4. Average For All τ_m, K, and k_m for Open Loop Test

Overall Average			10.133		0.186	7.596

Table 5. Closed Loop Test V_{ref} = 200

Experimental							
V_{ref}	$1024*k_p$	k_p	$\tau_m/(1+Kk_p)$ (ms)	τ_m (ms)	V_{ACT}	K	k_m
200.000	500.000	0.488	10.400	11.429	18.000	0.203	8.281
	1000.000	0.977	8.000	9.697	35.000	0.217	8.880
	1500.000	1.465	7.600	10.133	50.000	0.228	9.303
Average				10.420			8.821

Table 6. Closed Loop Test V_{ref} = 300

Experimental							
V_{ref}	$1024*k_p$	k_p	$\tau_m/(1+Kk_p)$ (ms)	τ_m (ms)	V_{ACT}	K	k_m
300.000	500.000	0.488	9.200	10.185	29.000	0.219	8.960
	1000.000	0.977	8.400	10.286	55.000	0.230	9.398
	1500.000	1.465	7.500	10.000	75.000	0.228	9.303
Average				10.157			9.220

Table 7. Closed Loop Test V_{ref} = 400

Experimental							
V_{ref}	$1024*k_p$	k_p	$\tau_m/(1+Kk_p)$ (ms)	τ_m (ms)	V_{ACT}	K	k_m
400.000	500.000	0.488	10.600	11.745	39.000	0.221	9.045
	1000.000	0.977	7.600	9.297	73.000	0.229	9.346
	1500.000	1.465	6.800	9.067	100.000	0.228	9.303
Average				10.036			9.231

Table 8. Average For All τ_m, K, and k_m for Closed Loop Test

Overall Average				10.204		0.222	9.091

Results

After collecting values and calculating which cannot be collected, we found that the τ_m, k_m , and *K* values for both closed loop and open loop test were similar with very minor differences especially between the k_m values of the two tests. For the final averages we got τ_m of 10.133ms, *K* of 0.186, and k_m of 7.596 for an open loop test. For the closed loop test, we got τ_m of 10.204, *K* of 0.222, and k_m of 9.091.

Error Analysis

The τ_m value we got from doing the open loop test could be much more accurate when it comes to using it in calculations. The reason for that is we have to use equation 1.6.6 to find the τ_m of the closed loop test. Also, some error was probably introduced when trying to read values off the oscilloscope. The graph was very blocky which made it very difficult to accurately detect where the function began to form. This definitely introduced some error in retrieving the value for τ_m .

Controller Design

Here we are trying to calculate the damping coefficient, ζ, and the natural frequency ω_n for any input values of k_p and k_I. The damping coefficient is important in order to understand the level of damping or the resistance in the motion of the system caused by any forces like vibration and oscillation. Depending on the parameters we want the system to perform by, we want to be able to set k_p and k_I to get a specific ζ. An overdamped system can cause the motors to respond slower to commands, while an underdamped system may result in a lot of damage caused by oscillatory or vibrational forces.

The natural frequency is also crucial in this experiment to give us a higher understanding of the oscillatory behavior. It helps us avoid things like damage caused by high resonance which is also caused by high levels of vibration in the system. Knowing both values definitely helps with further design of controllers.

Derivations and Calculations

To determine the system's natural frequency, ω_n, and the damping coefficient, ζ, for any given k_p and k_I, we first have to set up the system to be a second order system. Returning to Figure 4, we can simplify the closed loop system with a non-zero k_I as seen in Figure 11. By then simplifying this closed loop system, we produce the block in Figure 12.

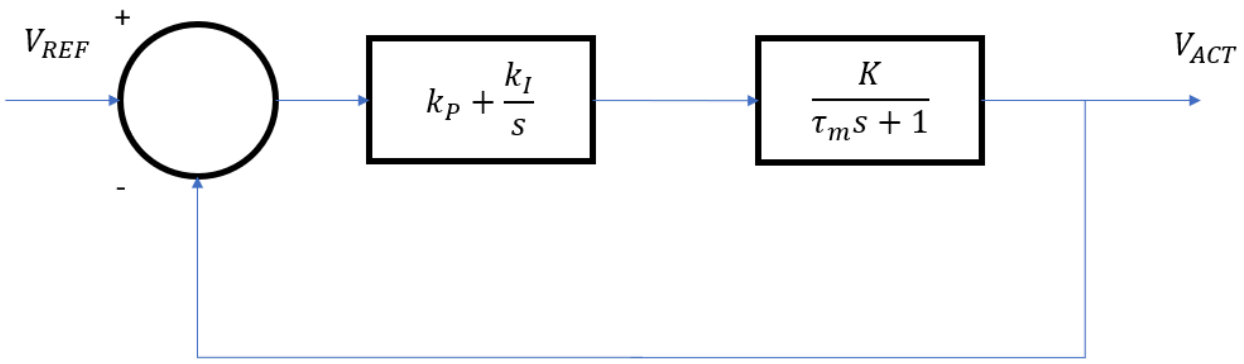

Figure 11. Closed Loop System Block Diagram

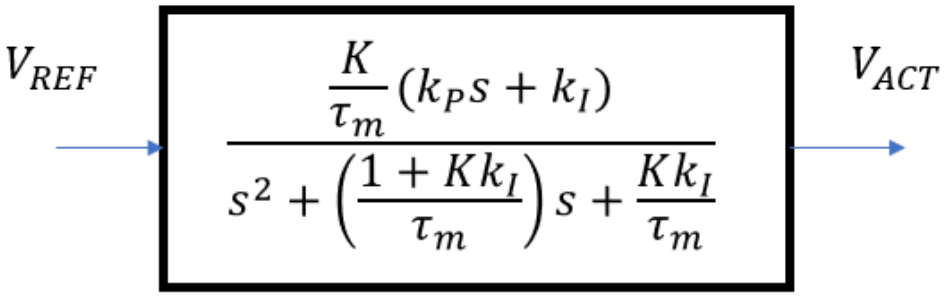

Figure 12. Simplified Closed Loop System Block Diagram

What we end up with is Equation 2.1.1, the canonical form of a second order system's transfer function.

$$\frac{V_{ACT}}{V_{REF}} = \frac{\frac{K}{\tau_m}(k_p s + k_I)}{s^2 + (\frac{1+Kk_p}{\tau_m})s + \frac{Kk_I}{\tau_m}} \tag{2.1.1}$$

For our analysis to work, k_I needs to be in the same units as $k_p s$, so an extra step we're required to take is to substitute Equation 2.1.2 into Equation 2.1.1 to get Equation 2.1.3.

$$k_I = \frac{k_I}{\Delta t} \tag{2.1.2}$$

$$\frac{V_{ACT}}{V_{REF}} = \frac{\frac{K}{\tau_m}(k_p s + \frac{k_I}{\Delta t})}{s^2 + (\frac{1+Kk_p}{\tau_m})s + \frac{Kk_I}{\tau_m \Delta t}} \tag{2.1.3}$$

Since we now have our transfer function, we can compare it to the canonical form of a second order system shown in Equation 2.1.4, producing Equations 2.1.5 and 2.1.6 for ω_n and ζ.

$$\frac{V_{OUT}}{V_{IN}} = \frac{C}{s^2+2\zeta\omega_n s+\omega_n^2} \quad (2.1.4)$$

$$\omega_n = \sqrt{\frac{Kk_I}{\tau_m \Delta t}} \quad (2.1.5)$$

$$\zeta = \frac{\frac{1+Kk_p}{\tau_m}}{2\omega_n} \quad (2.1.6)$$

Equations 2.1.5 and 2.1.6 can also be used to calculate k_p and k_I as seen in Equations 2.1.7 and 2.1.8.

$$k_I = \frac{\omega_n^2 \tau_m \Delta t}{K} \quad (2.1.7)$$

$$k_p = \frac{2\zeta\omega_n \tau_m - 1}{K} \quad (2.1.8)$$

Procedure

For calculating the ω_n and the ζ values we used the τ_m and the K values of the open loop test. We did this because we wanted to introduce less variables to compound error in our calculations. We did this by using the τ_m value given to use by the oscilloscope which is only possible in the open loop test. In the closed loop test we have to use equation 1.6.6 to find the true τ_m and this introduces more variables which can introduce more error as it has more assumed values than the τ_m of the open loop test.

We plugged in a value of 0.7 for k_p and 0.225 for K_i with a V_{REF} of 200 and ran the motor and read the transient response which is in Figure 14. To get k_p^* and k_i^* we used ω_n^* of 1.67*ω_n and a ζ of 0.95. After that we ran the motor again to read the transient response which is shown in Figure 15.

We then compared the transient responses for both runs to the ideal Simulink model as shown in Figures 14 and 15 using the block diagram provided to us in Figure 13. As seen on the right side of the block diagram, similar to scaling the voltage provided by DAC A, we also scale the simulated model exactly the same way. We multiply both by 13, add 2048, or 5V, then multiply by the reference 10V and divide by 4095 to account for the 12 bit system from Equation 2.2.1. To align the models, we also then shifted them to start at the same time.

$$V_{OUT} = \frac{n}{2^N} V_{OUT} \quad (2.2.1)$$

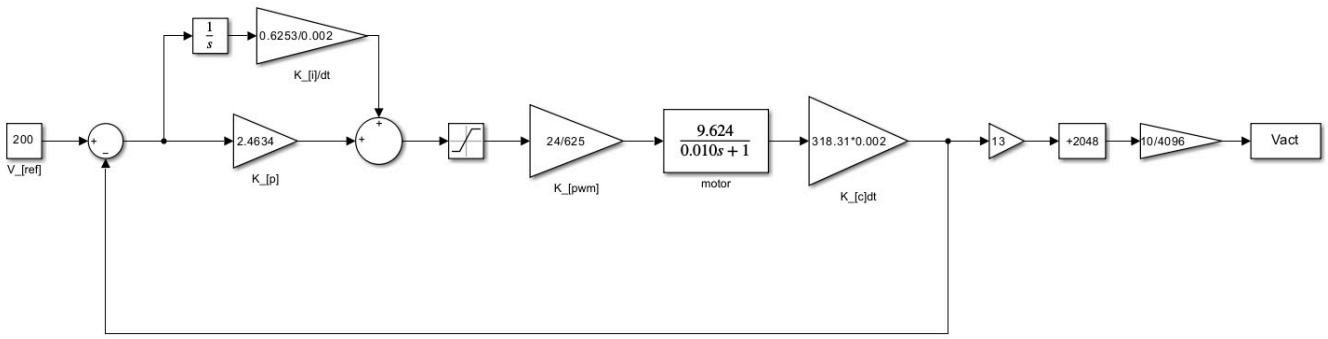

Figure 13. Simulink Closed Loop Block Diagram

Experimental Data

Table 9. Second Order Properties for K_p+K_I and K_p^*+K_I^*

Experimental								
V_{ref}	k_p*1024	k_p	k_I*1024	k_I	τ_m (ms)	K	ω_n	ζ
200.000	717.000	0.700	230.000	0.225	10.133	0.186	45.418	1.228
200.000	2522.499	2.463	640.256	0.625	10.133	0.186	75.712	0.950

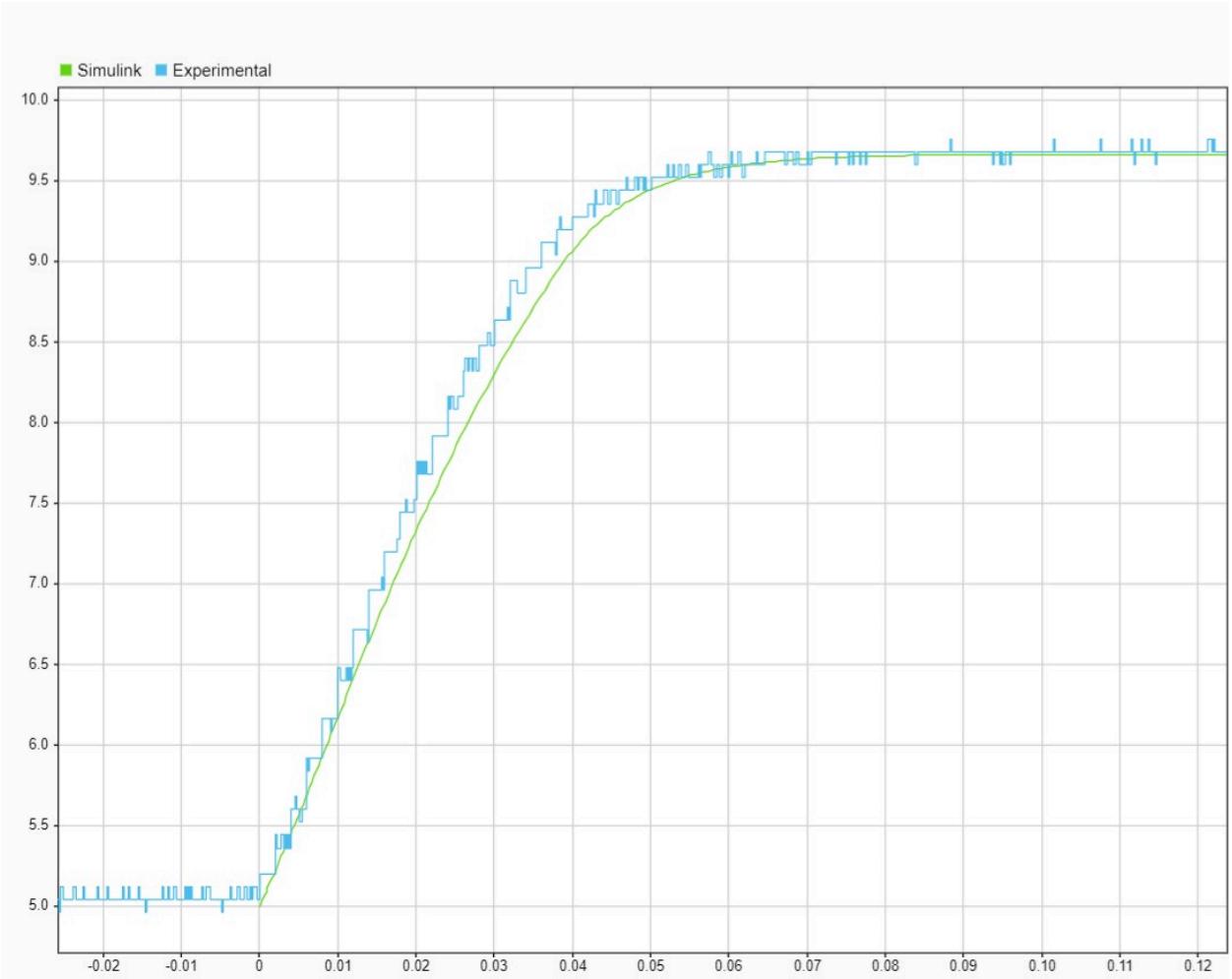

Figure 14. Experimental Vs Simulink Second Order Response for K_p=717 and K_I=230

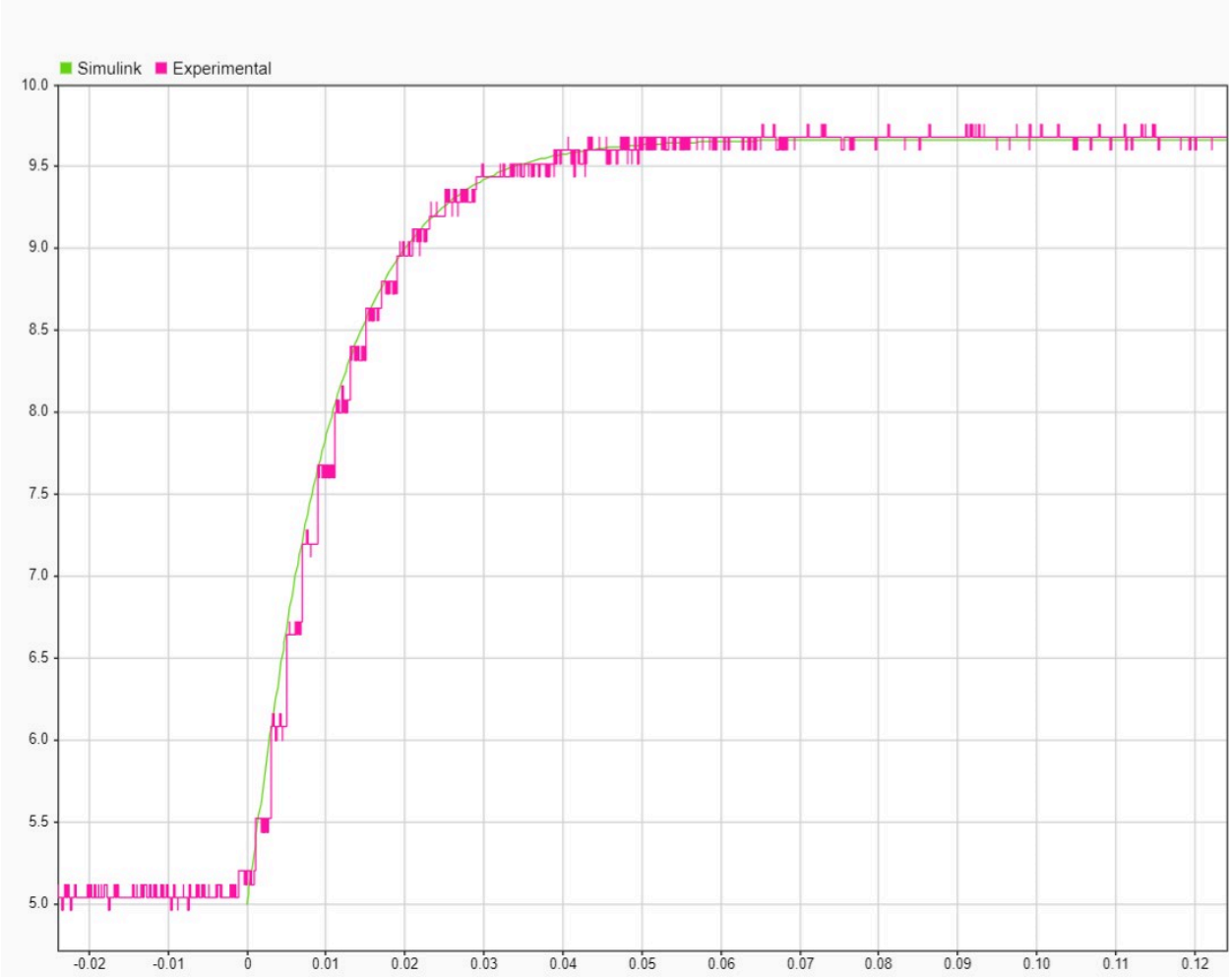

Figure 15. Experimental Vs Simulink Second Order Response for K_p^*=2522 and K_I^*=640

Results

From Equations 2.1.5 to 2.1.8, we were able to calculate our values for ω_n and ζ for our given k_p and k_I values and found them to be about 45 and 1.23, respectively. From this, we calculated our our k_p^* and k_I^* using our new ω_n^* and ζ^* values of 76 and 0.95, respectively, and found them to be 2.463 and 0.625, respectively. And after using these values to run in the real and simulated systems, we could compare the experimental data with the Simulink results seen in Figures 14 and 15. The two figures show fair correlation between the two, as we'd expect, with the experimental curve appearing much blockier and having slight variation. As we expected, the damping coefficient of a value of 1.23 created an overdamped system and comparatively, had a much slower response with a higher natural frequency than the second run we used.

Error Analysis

Looking at Figures 14 and 15 we can see that our simulated and experimental graphs line up well. Figure 14 does have a bit of a gap between the Simulink and the experimental model. This is likely because of the constants in the real system not being as accurate as we hoped when compared to an ideal simulated model. The experimental model likely has more error because the values used like voltage for instance might not be exact due to the hardware's inability to perform ideally which could skew things like our k_{PWM} value.

We also have errors occur due to the blockiness of the experimental data. This is because of the limited resolution that the DAC has along with the interrupt only occurring at set times and not continuously providing DAC values.

From a damping coefficient of 0.95, we also expected an underdamped system in Figure 15. However, this is not what we observed. Instead of overshooting and oscillating to its steady state value, we actually get a curve that looks overdamped. This is likely due to the effort reaching its maximum value and not allowing the system to gain enough speed to overshoot.

Conclusions

We determined the motor parameters τ_m of 10.133 and k_m of 7.596 for the open loop test and τ_m of 10.204 and k_m of 9.091 for the closed loop test, which we found to be fairly constant between the two tests. We also analyzed the effects of having k_i and k_p on the system using the transient responses shown on the oscilloscope and through their effects on ω_n and ζ.

Raw Code

```
;*******************************************************************************
;* Lab 5 main_ISR                                                              *
;*******************************************************************************
;* Summary:                                                                    *
;*  This code is designed for use with the 2016 hardware for ME305. This code  *
;* implements a proportional plus integral controller on the velocity of the ME 305 *
;* dc motor with encoder system.  A full user interface is required.           *
;*                                                                             *
;*                                                                             *
;* Author: William R. Murray, E. Espinoza-Wade and Charlie T. Refvem           *
;*  Cal Poly University                                                        *
;*  Fall 2016                                                                  *
;*                                                                             *
;* Revision History:                                                           *
;*  CTR 10/03/16                                                               *
;*   - Programmed the ISR that implements the controller, but with no user interface. *
;*  WRM 10/09/16                                                               *
;*   - Began to add user interface as time allowed.                            *
;*  WRM 11/12/16                                                               *
;*   - Initial user interface coded, and ready to merge with Charlie's controller.  *
;*  WRM 11/17/16                                                               *
;*   - User interface has passed moderate testing.                             *
;*  WRM 12/02/16                                                               *
;*   - Locked out changes in CL/|OL to prevent active data entry needing an <ENT>  *
;*       terminate when OL/|CL is pressed, and cleaned up unintentional consequences. *
;*  WRM 12/07/16                                                               *
;*   - Added a signed overflow [OVF] to ERR display.                           *
;*  WRM 11/16/17                                                               *
;*   - Changed default gains to reduced values to make headroom for boosting the  *
;*       natural frequency.  This is an issue due to the 2-quadrant limitations  *
;*       of the VNH5019 motor driver. Kp=1 and Ki=0.45.                        *
;*   - Blocked out enabling RUN while entering 1024*Kp or 1024*Ki.             *
;*  WRM 12/06/18                                                               *
;*   - Corrected sign issue in SAT_MULDIV                                      *
;*  WRM 01/06/19                                                               *
;*   - Adapted to run with Library V2.1 [changed PWM period to 625]            *
;*  WRM 03/10/19                                                               *
```

```
;*   - Changed the default gains for use with Library V2.1                    *
;*       [Kp=0.7; Ki=0.225], that is, [1024*Kp=717; 1024*Ki=230]               *
;*  WRM 05/14/22                                                              *
;*   - Tidied up comments and formatting                                      *
;*  WRM 05/14/22                                                              *
;*   - Stripped out TCOISR for testing Lab 5 library                          *
;*                                                                            *
;*  ToDo:                                                                     *
;*   - Could fool around with making EFF based on the motor current;          *
;*      otherwise, I'm happy ... until more bugs surface.                     *
;******************************************************************************

;/-----------------------------------------------------------------------------\
;| Include all associated files                                              |
;\-----------------------------------------------------------------------------/
; The following are external files to be included during assembly


;/-----------------------------------------------------------------------------\
;| External Definitions                                                      |
;\-----------------------------------------------------------------------------/
; All labels that are referenced by the linker need an external definition

          XDEF  main
          XDEF  Theta_OLD, RUN, CL, V_ref, KP, KI, UPDATE_FLG1

;/-----------------------------------------------------------------------------\
;| External References                                                       |
;\-----------------------------------------------------------------------------/
; All labels from other files must have an external reference

          XREF  ENABLE_MOTOR, DISABLE_MOTOR
          XREF  STARTUP_MOTOR, UPDATE_MOTOR, CURRENT_MOTOR
          XREF  STARTUP_PWM, STARTUP_ATD0, STARTUP_ATD1
          XREF  OUTDACA, OUTDACB
          XREF  STARTUP_ENCODER, READ_ENCODER
          XREF  DELAY_MILLI, DELAY_MICRO
          XREF  INITLCD, SETADDR, GETADDR, CURSOR_ON, CURSOR_OFF,
DISP_OFF
          XREF  OUTCHAR, OUTCHAR_AT, OUTSTRING, OUTSTRING_AT
```

```
            XREF  INITKEY, LKEY_FLG, GETCHAR
            XREF  LCDTEMPLATE, UPDATELCD_L1, UPDATELCD_L2
            XREF  LVREF_BUF, LVACT_BUF, LERR_BUF,LEFF_BUF, LKP_BUF,
LKI_BUF
            XREF  Entry, ISR_KEYPAD


            XREF  V_act_DISP, ERR_DISP, EFF_DISP, INTERVAL
            XREF  IFENTRY, Kscale
;/---------------------------------------------------------------------------\
;| Assembler Equates                                                         |
;\---------------------------------------------------------------------------/
; Constant values can be equated here


TFLG1         EQU   $004E
TC0           EQU   $0050

C0F           EQU   %00000001          ; timer channel 0 output compare bit
PORTT         EQU   $0240              ; PORTT pin 8 to be used for interrupt timing

LOWER_LIM     EQU   -625               ; number for max reverse duty cycle
UPPER_LIM     EQU   625                ; number for max forward duty cycle


;/---------------------------------------------------------------------------\
;| Variables in RAM                                                          |
;\---------------------------------------------------------------------------/
; The following variables are located in unpaged ram

DEFAULT_RAM:  SECTION

RUN:          DS.B  1                  ; Boolean indicating controller is running
CL:           DS.B  1                  ; Boolean for closed-loop active

V_ref:        DS.W  1                  ; reference velocity
V_act:        DS.W  1                  ; actual velocity
Theta_NEW:    DS.W  1                  ; new encoder position
Theta_OLD:    DS.W  1                  ; previous encoder reading
KP:           DS.W  1                  ; proportional gain
```

```
KI:          DS.W  1                ; integral gain
ERR:         DS.W  1                  ; error, where ERR = V_ref - V_act
EFF:         DS.W  1                ; effort
ESUM:        DS.W  1                  ; accumulated error (area under err vs. t curve)

UPDATE_COUNT: DS.B  1                 ; counter for display update
UPDATE_FLG1   DS.B  1                 ; Boolean for display update for line one

MOTOR_POWER:  DS.W  1                 ; motor actuation value
P_POWER:      DS.W  1                 ; P-term of motor actuation value
I_POWER:      DS.W  1                 ; I-term of motor actuation value

CURRENT:      DS.W  1                 ; motor current in mA
RESULT:       DS.W  1
RESULT2:      DS.W  1
RESULT_KI:    DS.W  1


VALUE_Y DS.W  1
ADD1    DS.W  1
ADD2    DS.W  1


;/-------------------------------------------------------------------------------\
;|  Main Program Code                                                            |
;\-------------------------------------------------------------------------------/
; Your code goes here

MyCode:       SECTION
main:
       jsr IFENTRY
spin:   bra   spin                  ; endless horizontal loop


;/-------------------------------------------------------------------------------\
;| Subroutines                                                                   |
;\-------------------------------------------------------------------------------/
; General purpose subroutines go here

TC0ISR:
```

```
    bset PORTT, $80              ; turn on PORTT pin 8 to begin ISR timing
    inc UPDATE_COUNT             ; unless UPDATE_COUNT = 0, skip saving
    bne measurements             ; display variables
    movw V_act, V_act_DISP       ; take a snapshot of variables to enable
    movw ERR, ERR_DISP           ; consistent display
    movw EFF, EFF_DISP
    movb #$01, UPDATE_FLG1       ; set UPDATE_FLG1 when appropriate
    lbra Next_TC0_Set

measurements:; Measurements block
; Read encoder value
    jsr READ_ENCODER             ; read encoder position
    std Theta_NEW                ; store it
; Compute 2-point difference to get speed
    subd Theta_OLD               ; compute displacement since last reading
    std V_act                    ; store displacement as actual speed
    ldy #13
    emuls
    addd #2048
    jsr OUTDACA                  ; sets DACA to V_act
    movw Theta_NEW, Theta_OLD    ; move current reading to previous reading
; Compute Error
    ldd V_ref
    subd V_act
    std ERR                      ; subtracts V_ref-V_act to get Error
; Compute Effort
    ldy ESUM
    jsr Sat_Dble_Byte_Add        ; saturated double byte addition of Error(D) and Old
ESUM(Y) to get New ESUM
    std ESUM                     ; stores New ESUM
    ldy KI
    emuls                        ; multiplies KI and New ESUM
    ldx #1024                    ; loads 1024 in hex
    edivs                        ; divides ESUM*KI by 1024
    sty RESULT_KI                ; push the result of upper branch to stack    EDIT
    ldd ERR
    ldy KP
    emuls                        ; multiplies KP and ERR
    ldx #1024                    ; loads 1024 in hex
    edivs                        ; divides ERR*KP by 1024
```

```
    sty RESULT2
; Saturate Effort
    ldd RESULT_KI
    jsr Sat_Dble_Byte_Add          ; saturated double byte addition of the two branches
to get effort
    std EFF
    bmi neg_check                  ; branches to neg_check if result is neg
    cpd #625
    bmi exit_sat_eff
    beq exit_sat_eff               ; branches to exit_sat_eff if result is less than or equal to
625
    ldd #625                       ; loads 625 into D if pos sat
    std EFF
    bra exit_sat_eff
neg_check:
    cpd #-625
    bmi neg_overflow               ; branches to neg_overflow if result is <-625
    bra exit_sat_eff
neg_overflow:
    ldd #-625                      ; loads -625 into D if neg sat
    std EFF                        ; stores D into EFF
exit_sat_eff:
    tst RUN
    beq skip_update
    jsr UPDATE_MOTOR
    ldd EFF
    ldy #100
    emuls
    ldx #625
    edivs
    sty EFF
    bra Next_TC0_Set
skip_update:
    ldd EFF
    ldy #100
    emuls
    ldx #625
    edivs
    sty EFF
    ldd #$00
```

```
      std ESUM
      jsr UPDATE_MOTOR

Next_TC0_Set: ; Sets the next TC0 value to interrupt at
            ldx TC0                                 ; Loads TC0 into accul X
            addx INTERVAL                       ; Adds the value of INTERVAL into x
            stx TC0                                 ; Stores x into TC0
            bset TFLG1, C0F                 ; Sets the 0 bit of TFLG1 to clear the interrupt
flag
            rti


Sat_Dble_Byte_Add:
   pshy               ; Pushes value in accul Y to the stack
   addd SP            ; Adds the values at SP (Y:Y+1) to D
   bvs case_overflow     ; Branches to case_overflow if Overflow flag is set
   bra exit_Sat_Add      ; Branches to exit_Sat_Add if there is no saturation
(pos+pos=pos,
                     ; neg+neg=neg, or there is pos+neg)
case_overflow:
   tsty               ; Tests Y to determine its sign
   bmi neg_neg_overflow  ; Branches to neg_neg_overflow if Y is negative
   bra pos_pos_overflow  ; Branches to pos_pos_overflow if Y is negative
   bra exit_Sat_Add
neg_neg_overflow:
   ldd #%1000000000000000; Loads -32768 in accul D for saturated negative negative
addition
   bra exit_Sat_Add
pos_pos_overflow:
   ldd #%0111111111111111; Loads 32767 in accul D for saturated positive positive
addition
   bra exit_Sat_Add
exit_Sat_Add:
   puly               ; Pulls original value in accul Y from the stack
   rts




;/---------------------------------------------------------------------------\
```

```
;| Vectors                                                                    |
;\---------------------------------------------------------------------------/

; Add interrupt and reset vectors here:
        ORG  $FFFE                  ; reset vector address
        DC.W Entry

        ORG  $FFEE          ; INTERRUPT vector address
        DC.W TC0ISR
```