Demonstrating introductory control systems concepts on inexpensive hardware

by

Shane R. Smith

B.S., Kansas State University, 2015

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical & Nuclear Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2017

Approved by:

Major Professor
Dale Schinstock

Copyright

Abstract

There is a trend in the control literature and in university control education research to develop inexpensive laboratory equipment for control based laboratories. But can using cheaper equipment obfuscate the concepts we are trying to demonstrate in the experiments?

To investigate this, lab concepts were examined using an inexpensive platform developed at Kansas State University, Eeva, and compared to the existing lab equipment used in the introductory controls course, the MotorLab. While many lab concepts were successfully demonstrated on the cheaper hardware, they were obscured by higher order effects such as speed filters, back EMF effects, and encoder resolution. The effective operating range of the hardware also suffered from lower saturation limits and higher friction values, making the design of experiments more difficult.

Care should be taken when designing inexpensive laboratory equipment to ensure that the lessons desired can still be demonstrated clearly to the students using the equipment.

Table of Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank Dr. Schinstock for all of his help in developing this thesis, as well as giving me the opportunity to pursue my master's here at Kansas State University. I'd also like to thank Dr. White for all his help and advice during my time here at K-State. The mentorship provided by Professor Spaulding was also invaluable during my career here at Kansas State University.

This thesis is dedicated to my wonderful wife Sarah, who has supported and enabled me throughout my time here at K-State; I couldn't have done it without you.

Nomenclature

C.G. – Center of Gravity

Eeva – a two-wheeled inverted pendulum robot developed by Dr. Schinstock at Kansas State University to help teach a Mechatronics course.

EMF – the electromotive force, a voltage created by a spinning motor that is proportional to its speed and opposes the supplied voltage.

IR – Infrared

Li-Po – Lithium Polymer

MotorLab – lab equipment created by Dr. Schinstock and Dr. White to help teach an introductory controls course at K-State

PI – Proportional Integral controller.

PID – Proportional Integral Derivative controller.

SWD – Serial Wire Debug, a standard interface for microcontrollers.

Chapter 1 - Introduction

Laboratories have been shown to help with introductory control systems education ([2], [6], [7], & [9]]). Working with physical systems can help students better understand mathematical principles. Labs also allow time for instructors to reinforce important concepts and correct any mistakes in understanding on the students' part. For control systems, it also shows important concepts such as system instability and bandwidth directly from the real system's characteristics.

However, enrollment and class sizes are increasing. This adds to the number of students in the laboratory, necessitating an increase in the lab equipment required to effectively teach courses.

These considerations make setting up inexpensive, personal lab equipment an attractive alternative. It allows students to work on the lab with more freedom in location and time. Personal inexpensive equipment also allows laboratory size to scale easier. Additional equipment can be purchased and installed to account for growing class sizes. There is a trend in control systems literature towards developing such equipment for use in university education ([2], [6], [7], [9], [11] & [16]).

An important question is whether using inexpensive equipment makes the control systems concepts demonstrated in labs harder for students to grasp due to some limiting aspects of the cheaper equipment. Frequently the lower cost equipment has lower saturation limits, poor sensor resolution, and more noise on all levels of the system. Nonlinear effects such as friction, gearbox backlash, etc. are magnified as well.

To investigate the possible issues of teaching controls laboratories with cheaper equipment, this thesis explores the use of such equipment in several different lab exercises that were developed using more expensive hardware. These labs are meant to demonstrate key concepts for an introductory course in control systems. Each of the labs is investigated on Eeva, a small robot developed for a Mechatronics course at Kansas State University. The results are compared to results using the MotorLab, the control systems lab hardware that has been in use at this university for over a decade.

Chapter 2 - Laboratory Hardware

Two sets of hardware are discussed in this thesis. The MotorLab has been used as the laboratory equipment for the introductory controls course at Kansas State University for fifteen years with some modifications. It will be the standard to which results will be compared for all experiments conducted in this thesis.

Eeva was developed as an inexpensive, student owned, teaching hardware for a Mechatronics course. Since it includes cheaper hardware, this thesis investigates how the sensors and hardware on Eeva perform when attempting experiments demonstrating introductory controls concepts.

MotorLab

The latest version of the MotorLab consists of a brushless motor, a motor amplifier, power supply, and an ST Discovery microcontroller board as shown in Figure 2.1. The motor specifications as provided by the motor manufacturer are shown in the modelling chapter. All components were designed and specified to provide the cleanest, easiest to follow control systems lab results by having much larger bandwidths and operating limits than the students use in the labs. This has allowed the MotorLab system to reliably service students for many years. A cost estimate is difficult to make for the MotorLab since it was fabricated using the facilities at Kansas State University. However, the materials alone (not accounting for manufacturing, design, or labor) cost about $700 per unit.

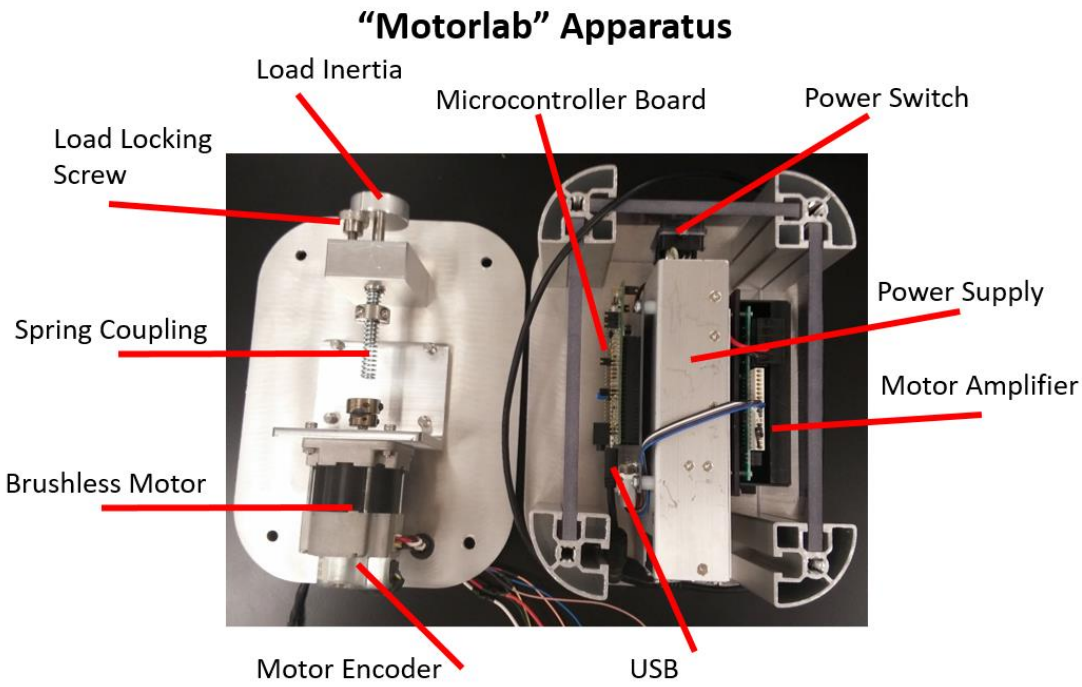


Figure 2.1: Components of the MotorLab

Eeva:

Eeva has two brushed motors with 30:1 gearboxes for increasing the torque supplied to the drive wheels. Power is supplied by a two cell Li-Po battery. All components (shown in Figure 2.2) were designed to be as inexpensive as possible while allowing a platform for Mechatronics students to program different scenarios involving several layers of controls on the motors. Since the introductory controls course is not a prerequisite for mechatronics, the course only covers simple tuning procedures for PID controllers and a very basic modelling problem for the IR detector array for a line following project. The hardware chosen was designed to perform the required functions adequately while being affordable for students to personally own the robot. Eeva has an estimated commercial cost of a little over $200, but the components on it could be made into a much cheaper device designed for controls experiments by removing un-

necessary components. This thesis uses Eeva as an example of inexpensive hardware for control systems laboratory experiments, even though that was not the original purpose for this design.

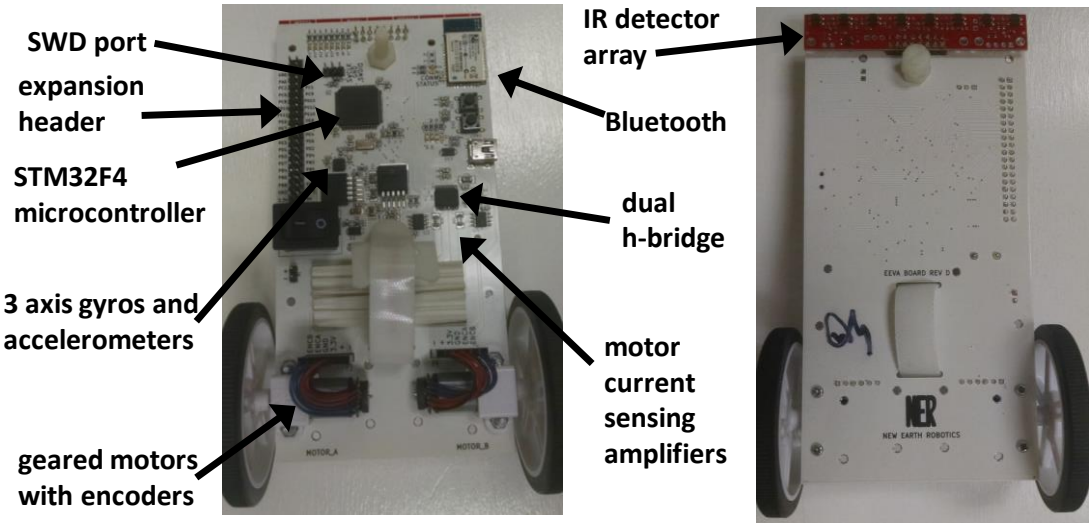


Figure 2.2: Components of Eeva

Chapter 3 - Modelling

This chapter describes the measurement/determination of parameters for the dynamic models for the MotorLab and Eeva as well as the derivation of the dynamic models used. First the parameters for the MotorLab are given in Table 3.1. Then the parameters for Eeva are given in Table 3.2 with details on the determination for each value. Finally, the dynamic models used in the experiments conducted for this thesis are presented along with general concepts used.

MotorLab System Parameters:

The important parameters for the dynamic model development are given in Table 3.1. The motor inertia and motor torque constant are provided by the manufacturer. This along with the inertia of the shaft collar provides the lumped inertia. The speed filter is implemented in computer code, and therefore known. The viscous friction coefficient is determined in the experiment described in Chapter 4. Finally, the spring is only used in the last lab covered in this thesis, that coefficient is found in the experiment described in Chapter 7.

Table 3.1: System Parameters for the MotorLab

	Value	Description
k_{tm}	0.05 N-m/A	Motor Torque Constant
J_m	1.29e^{-5} kg-m^2	Lumped Inertia
b_m	1e^{-5} N-m-s	Viscous Friction Coefficient
ω_{fm}	300 rad/s	Speed Filter Cutoff Frequency
k_{sm}	0.22 N-m/rad	Spring Constant (Found in Chapter 7)

The parameters needed to develop dynamic models for the MotorLab form a smaller set than those required for Eeva (compare Table 3.1 to Table 3.2). This is true because the motor amplifier implements closed loop current control for the motor. The bandwidth of this closed-loop system is high (on the order of 2400 rad/s), and much faster than any of the other dynamics

considered. Therefore, the input to the plant can be considered current (torque), eliminating the need to include components of the electrical system in the model.

Eeva System Parameters:

Table 3.2: System Parameters for Eeva

	Value	Description
R_e	6.8 Ohms	Motor Resistance
L_e	0.9 mH	Motor Inductance
k_{be}	0.0025 V-s/rad	Motor Back EMF Constant
k_{te}	0.0025 N-m/A	Motor Torque Constant
J_e	$3.23e^{-8}$ kg-m^2	Lumped Inertia
b_e	$2.6e^{-7}$ N-m-s	Viscous Friction Coefficient
ω_{fe}	126 rad/s	Speed Filter Cutoff Frequency
m_e	0.137 kg	Total Eeva Mass without wheels
l_e	0.0315 m	Distance from C.G. to Motor Shaft
J_{board}	$3.3e^{-4}$ kg-m^2	Board Inertia, Experimentally Found in Chapter 7

Eeva Motor Resistance:

The motor resistance was measured directly from the motor leads using a multi-meter. The measurement was repeated at several motor positions to account for variation. An average value of 6.8 Ω was chosen as the resistance.

Eeva Motor Inductance:

The inductance was calculated experimentally using a square wave of current while the wheels were held stationary (to eliminate back EMF effects). The output from the current measuring hardware on EEVA was captured on an oscilloscope (see Figure 3.1), showing the decay of current from an initial condition. The time constant of that decay is equal to the motor inductance divided by the motor resistance. Knowing the motor resistance and this time constant, we can calculate the motor inductance.

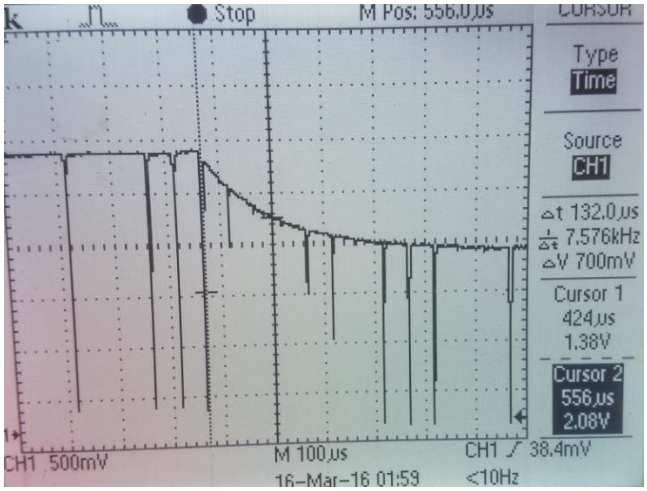


Figure 3.1: Oscilloscope Reading Used to Measure Eeva's Motor Inductance

From the screenshot of the oscilloscope, the time constant is 132 µs. With that and the motor resistance given above the inductance is: $L_e = R_e \cdot \tau = (6.8\,\Omega) \cdot (132e^{-6}\text{ sec}) = 8.976e^{-4}\ H$.

Motor Torque Constant / Back EMF Constant:

These constants are equivalent values with different units, and it is easier to measure the back EMF constant. Therefore, the back EMF coefficient was measured. With two motors coupled together using a small plastic hub across both drive shafts, one was driven at constant speeds, and the other unconnected motor had the voltage across its leads measured. Measurements were made at several speeds and then the slope of the resultant line gave the back EMF, K_b, coefficient as shown in Figure 3.2.

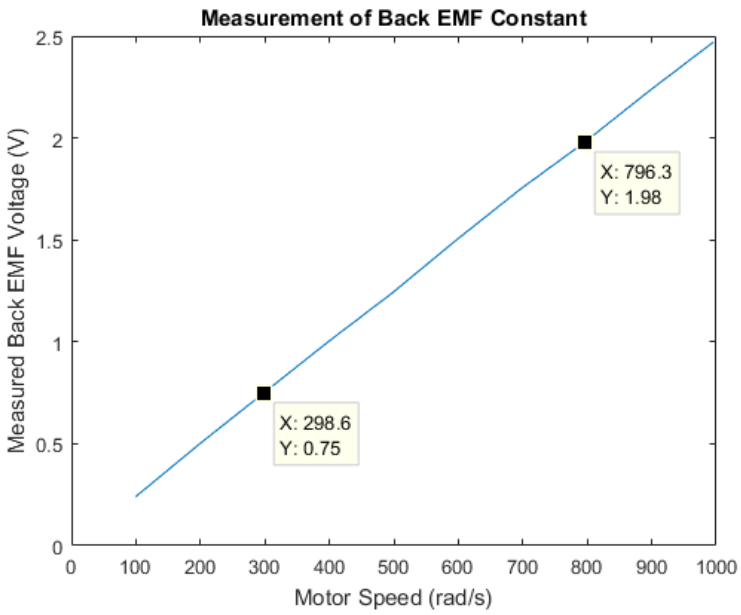


Figure 3.2: Eeva Back EMF Constant Measurement

$$K_{be} = \frac{(1.98 - 0.75)V}{(796.3 - 298.6)rad/s} = 0.0025\ \frac{V \cdot s}{rad}$$

Eeva Motor Inertia:

To calculate the moment of inertia, the motor was disassembled to measure the diameter of the armature as well as its mass. The armature was assumed to be a solid cylinder, using $J_e = \frac{1}{2}mr^2$ to approximate the mass moment of inertia. Here m = 2 g and r = 4.08 mm. This gives a moment of inertia of 1.66464 e^{-8} kg-m^2. The experiment described in Chapter 4 was used to obtain a total lumped inertia estimate of the motor and gearbox effects (not including the wheels), which is the number in Table 3.2.

Gear Ratio:

The gear ratio was provided by the manufacturer as 29.86, this can be confirmed by counting the gear teeth. Although it is provided here for completeness, all lab results presented here are given at the motor shaft, bypassing the gearbox and wheel completely.

Mass of Eeva:

For most labs Eeva is supported with its wheels off the ground, meaning the mass is not needed. However, if it is supported by the wheels to model a hanging pendulum, the mass will be part of the dynamics. Since the wheels are supported, the mass of the robot without the wheels was measured on a lab scale, giving a mass of 137 grams.

Center of Gravity for Eeva:

To estimate the location for the C.G. of Eeva, the robot was balanced on the point of pen along a line drawn on the center of the board, and then the distance between the balancing point and the center of the motor shaft was measured and found to be 31.5 mm.

MotorLab Model Development:

Current control is used on the MotorLab, which simplifies the modeling significantly. The first three speed control labs can be modeled by summing torques, taking the Laplace transform, and solving for a transfer function as follows:

$$J_m\dot{\omega}(t) = k_{tm}i(t) - b_m\omega(t)$$

$$J_m s\omega(s) = k_{tm}i(s) - b_m\omega(s)$$

$$\frac{\omega(s)}{i(s)} = \frac{k_{tm}}{J_m s + b_m}. \tag{1}$$

A speed filter is used on the MotorLab to measure speed. An analysis and discussion of the speed filters for both the MotorLab and Eeva is provided later in this chapter. The speed filter for the MotorLab is given below, although it isn't a factor in most of the labs discussed, as will be shown in each lab's respective section.

$$G_{fm}(s) = \frac{\omega_m(s)}{\omega(s)} = \frac{\omega_{fm}^2}{s^2 + 0.707\omega_{fm}s + \omega_{fm}^2} \tag{2}$$

An initial condition response is also investigated in the first lab. For this the initial condition is speed, then the motor amplifier is turned off, driving the motor torque to zero. A simple, time-domain solution for our linear modeled system can be arrived at as follows. First sum torques for our system:

$$\tau_m(t) = J_m\dot{\omega}(t) + b_m\omega(t).$$

Then we set the input torque to zero and take the Laplace transform, making sure to account for initial conditions:

$$0 = J_m s\omega(s) - J_m\omega_0 + b_m\omega(s).$$

Then we can collect terms and take the inverse Laplace transform to get a time domain solution:

$$\omega(t) = \mathcal{L}\left\{\frac{J_m\omega_0}{J_m s + b_m}\right\} = \omega_0 \cdot \mathcal{L}^{-1}\left\{\frac{1}{s+\frac{b_m}{J_m}}\right\}.$$

The inverse Laplace gives us equation 3, a simple exponential decay model for our system response to the initial condition:

$$\omega(t) = \omega_0 e^{-\frac{b_m}{J_m}\cdot t}. \tag{3}$$

For the resonance lab, the last lab discussed, a spring is included in the system. A picture of the model is given below in Figure 3.3.

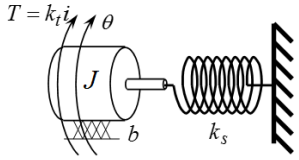


Figure 3.3: Electromechanical Model for Motor Position on the MotorLab

The position control model can be modelled by summing torques, taking the Laplace transform, and solving for a transfer function as follows:

$$J\ddot{\theta}(t) = k_t i(t) - b\dot{\theta}(t) - k_s\theta(t)$$

$$Js^2\theta(s) = k_t i(s) - bs\theta(s) - k_s\theta(s)$$

$$\frac{\theta(s)}{i(s)} = \frac{k_{tm}}{J_m s^2 + b_m s + k_{sm}}. \quad (4)$$

Eeva Model Development:

Several related dynamic models for Eeva are developed in this section, using the parameters presented in the previous section. These models are used in Chapters 4 through 7, which describe the experiments. Figure 3.4 is a schematic model of the motor speed model used for Eeva in the three experiments described in Chapters 4 through 6.

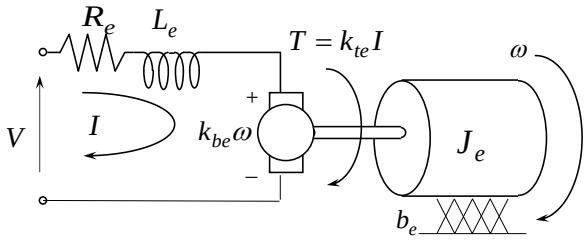


Figure 3.4: Electromechanical Model for Motor Speed on Eeva

The hardware used on Eeva did not allow for good current control, as a result the system must be modelled with Voltage as the input. This means that the back EMF effects need to be included in our model. The motor inductance is not needed because the electrical pole at $-\frac{R_e}{L_e}$ has a magnitude of 7600 rad/s; which is well over ten times larger than the bandwidth of any of our controllers or any of the other poles in the model, including the speed filter.

For the first three labs covered in this thesis the input is voltage and the output is the speed of the motor. The motor torque is $\frac{k_{te}}{R_e}(V - k_{be}\omega)$ where V is the voltage supplied to the motor. This takes into account the back EMF effects. A transfer function can be derived using this motor torque and by writing the sum of torques, taking the Laplace transform, and solving for output over input to get equation (1) below.

$$J_e\dot{\omega} = \frac{k_{te}}{R_e}(V - k_{be}\omega) - b_e\omega$$

$$J_e s\omega(s) = \frac{k_{te}}{R_e}V(s) - \frac{k_{te}k_{be}}{R_e}\omega(s) - b_e\omega(s)$$

$$G_{me}(s) = \frac{\omega(s)}{V(s)} = \frac{k_{te}}{R_e J_e s + k_{be}k_{te} + R_e b_e} \tag{5}$$

A speed filter on Eeva is represented by the following transfer function.

$$G_{fe}(s) = \frac{\omega_m(s)}{\omega(s)} = \frac{{\omega_{fe}}^2}{s^2 + 1.414\omega_{fe}s + {\omega_{fe}}^2} \tag{6}$$

It is the result of the method used to measure the speed of the motor using the encoder. A discrete time implementation of (6), which includes a derivative, is used to estimate the speed of the motor from the position measured by the encoder. This filter has the transfer function of:

$$\frac{\omega_m(s)}{\theta(s)} = \frac{\omega_{fe}^2 s}{s^2 + 1.414\omega_{fe} + \omega_{fe}^2}$$

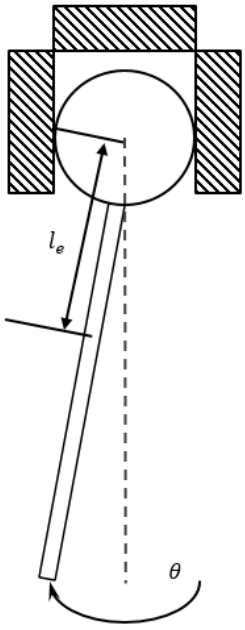


Figure 3.5 Eeva Pendulum Model

In the fourth lab discussed in this thesis we investigate an inverted pendulum setup for Eeva to find a resonant frequency as shown in Figure 3.5. Two motors are used to apply torque. The effect of gravity must be included in this model. It applies a torque opposing the motor torque equal to $m_e g l_e sin\theta$. If the angle of the robot remains small, a small angle approximation can be used to replace $sin\theta$ with θ. At an angle of $\pi/4$ this approximation has 11% error. With all this taken into account the sum of torques gives:

$$J_{board}\ddot{\theta}(t) = \frac{2k_{te}}{R_e}\left(V(t) - k_{be}\dot{\theta}(t)\right) - m_e g l_e \theta(t) - b_{board}\dot{\theta}(t).$$

Taking the Laplace transform results in

$$J_{board}s^2\theta(s) = \frac{2k_{te}}{R_e}V(s) - \frac{2k_{te}k_{be}}{R_e}s\theta(s) - m_e g l_e \theta(s) - b_{board}s\theta(s).$$

Note the difference between J_{board} (containing the lumped inertia of the motors, battery, components, and board) and J_e, which only contains the inertia of the motor. Gathering terms, solving for output over input, and putting the result in standard second order form gives the following transfer function.

$$\frac{\theta(s)}{V(s)} = \frac{\frac{2k_{te}}{R_e J_{board}}}{s^2 + \left(\frac{2k_{te}k_{be}}{R_e J_{board}} + \frac{b_{board}}{J_{board}}\right)s + \frac{m_e g l_e}{J_{board}}} \quad (7)$$

Speed Filters

On both sets of hardware, speed is found by taking any change in encoder count and dividing it by the change in time, a crude differentiation of the position measurement. Because the encoder readings are quantized, this approximation of speed gives large spikes at each change in encoder reading. This noisy signal is filtered using a low pass filter with a tunable cutoff frequency. On the MotorLab this cutoff frequency was chosen high enough to give a clean signal but low enough that students could observe its effects when increasing controller gains to

drive the system harder. On Eeva the cutoff frequency was chosen at the highest point that allowed clear measurement of speed. A graph showing the discrete implementation of a speed filter used on Eeva compared to a continuous model is given in Figure 3.6. This figure confirms that our continuous model approximates the discrete implementation used on Eeva, so we can vary the cutoff frequency in the continuous model to easily observe how increasing this cutoff frequency would affect the resulting speed measurements.

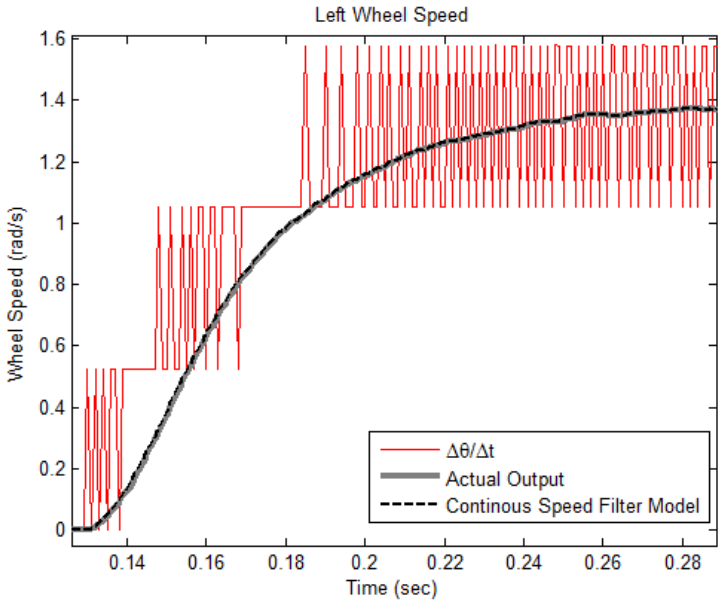


Figure 3.6: Speed Filter and Continuous Model

If the cutoff frequency is increased the speed filter dynamics will affect observed responses less in more of the labs we consider, but the resulting speed measurement will be noisier as well. In Figure 3.7 the cutoff frequencies are varied and the results are demonstrated. It is shown that the cutoff frequency could be increased a little for this case while still removing a lot of the noise, but large increases will not filter out the noise from the quantized encoders. The small increase in the cutoff frequency will not allow the speed filter dynamics to be ignored in the majority of the labs discussed in this thesis, since the dynamics of the speed filter are close to

the dynamics of motors until the speed filter cutoff frequency is increased by at least a factor of 3, which prevents it from filtering out the noise from the quantized encoders.

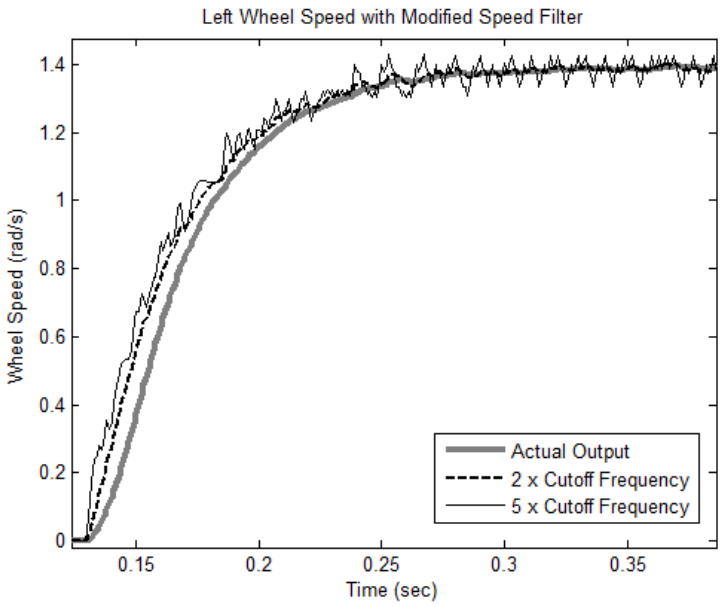


Figure 3.7: Speed Filter Performance with Varying Cutoff Frequencies

High Frequency Dynamics

Throughout this thesis the concept of high frequency dynamics is used. This is the concept that there are always un-modeled dynamics in our system that may cause it to become unstable or behave in a non-predicted manner when pushing the system to high performance. In the introductory controls systems course at Kansas State University this concept is used as analog to a system's bandwidth before the students are introduced to frequency response. It serves to illustrate some of the real-world reasons why controller gains cannot be increased limitlessly without leading to issues.

Chapter 4 - Experiment I – Friction Coefficient Estimate

This experiment is taken directly from the ME 570, Control of Mechanical Systems I, course in the ME curriculum at KSU. In this lab, students validate the concept of using a transfer function model to capture the behavior of a physical system and to make connections between model parameters and physical characteristics of the real system. Using the MotorLab, this is accomplished with a simple transfer function. The lab takes place early in the semester, so students have not yet dealt with complicated models or advanced control systems topics.

One of the core concepts of controls is that system behavior can often be predicted with simple models. Estimating a linear coefficient of friction demonstrates how a simple model can predict the response of more complicated systems while demonstrating how the model and real parameters interact. Students are also familiar with the nonlinear (static and viscous) nature of friction, making it a good introduction to modeling.

MotorLab:

The open loop speed model for the MotorLab is given in Figure 4.1 with equations (1) and (2) from chapter 3. The speed filter is provided here for completeness, although it will be shown that it does not have a large effect on the system's response.

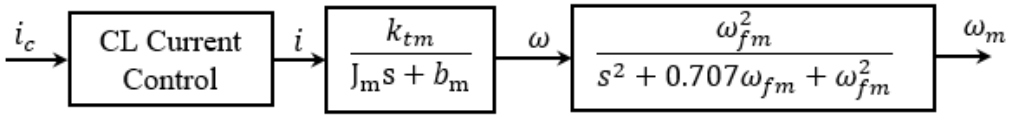


Figure 4.1: MotorLab Open Loop Speed Model

For the MotorLab, students first put the motor in open loop (current) control and take steady state speed values at different values of current. Then a plot of motor input torque (current) vs motor speed is created, allowing a friction coefficient to be estimated from a best-fit line, as shown in Figure 4.2, because the dc gain of the system from equation (1) is given by:

$$K_{dcm} = \frac{K_{tm}}{b_m}.$$

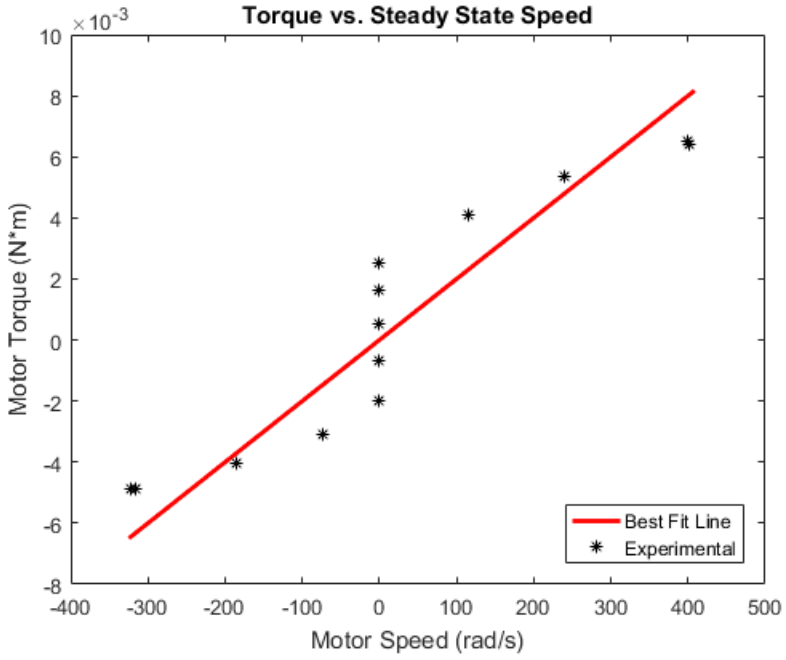


Figure 4.2: MotorLab Friction Coefficient Estimate

Once the students have estimated a coefficient for the linear model of friction, they use that to model an initial condition response. The motor current is set to 0.15 Amps and speed is allowed to settle to steady state; then the motor current is set to zero and data on the decay of speed is collected. Students compare the linear model for friction they created earlier in the lab with the actual response of the system, as shown in Figure 4.3.

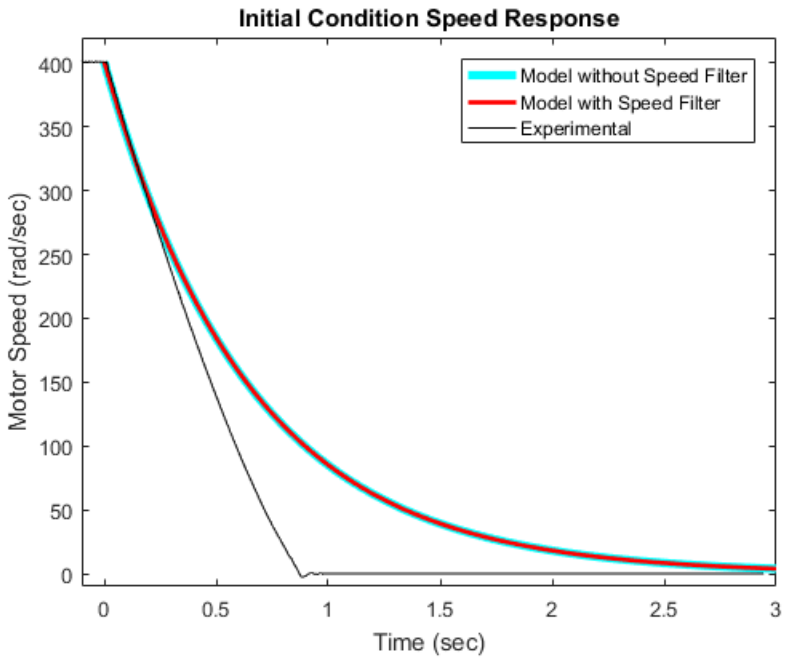


Figure 4.3: MotorLab Initial Condition Response

The response of the system shows the nonlinear effects of friction on the decay of the motor speed. A simple exponential decay solution to the linear model is graphed as well, showing that simple models can capture a lot of the response characteristics of the real system. Students observe that the greatest difference between their linear model and the actual response is at low speed, which makes intuitive sense since that is where the higher static friction coefficient starts to operate.

Eeva:

The open loop model for speed on Eeva is given in Figure 4.4. Input for this model is the voltage command and output is the measured speed. The equations for the speed filter and speed response model from Chapter 3 are also provided in this figure for clarity.

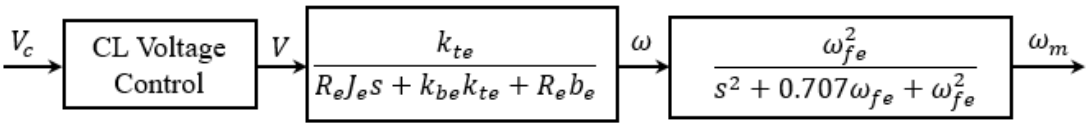


Figure 4.4: Eeva Open Loop Speed Model

On Eeva, the voltage input complicates this lab because students must consider the effects of back EMF. The constant they end up measuring includes those effects, rather than being just a measurement of the friction coefficient. This is seen in the dc gain of the system in equation (5), which is given by:

$$K_{dce} = \frac{k_{te}}{k_{be}k_{te} + R_e b_e}$$

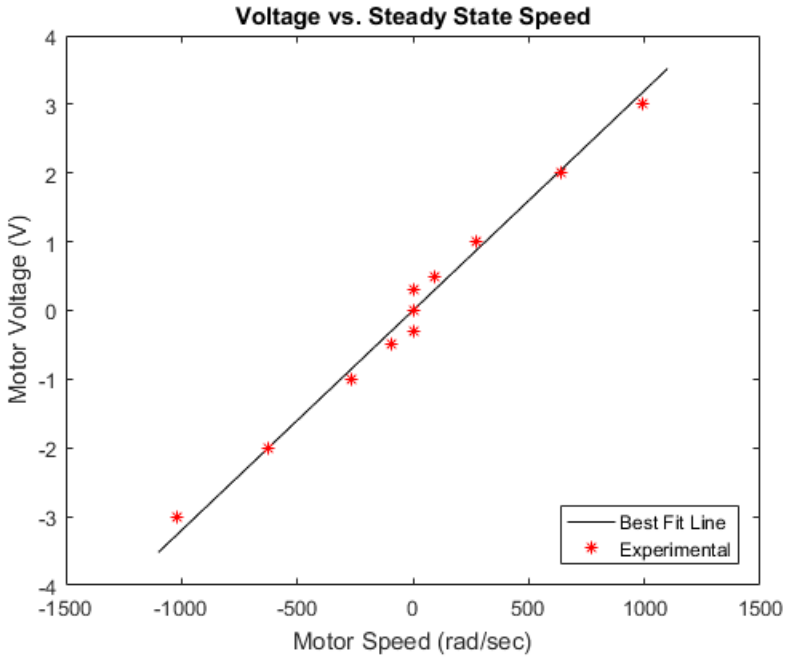


Figure 4.5: Eeva Friction Coefficient Estimate

Students can still calculate the friction coefficient, but it must be solved for using the dc gain equation given above and the values for back EMF, torque constant, and motor resistance as well. Once this value is calculated, the students can look at the initial condition response just as

done for the MotorLab. An initial voltage is set on the motor, steady state speed is reached, and then the motor voltage is set to zero and the speed decay is observed as shown in Figure 4.6.

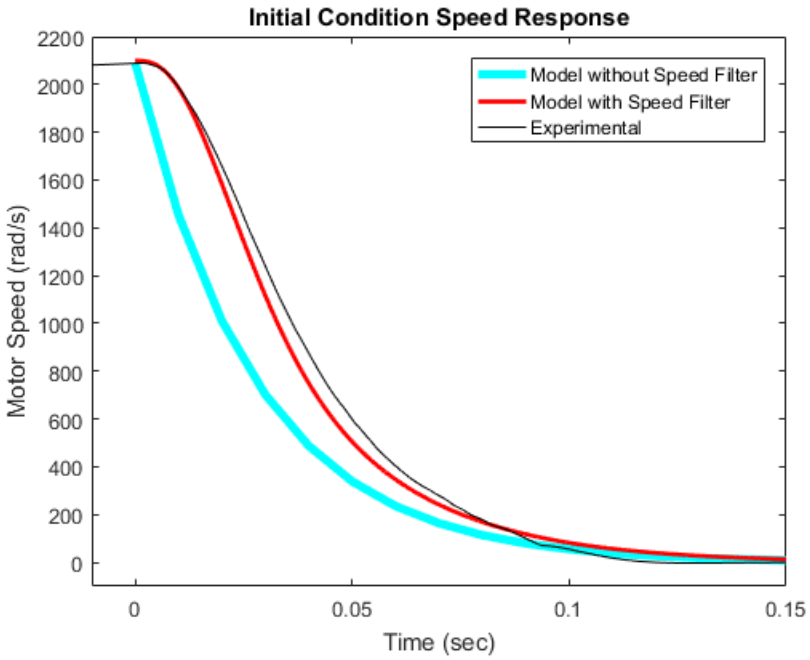


Figure 4.6: Eeva Initial Condition Response

As you can see, when the decay starts the simple model does not match the response as well as the MotorLab. This is a result of the speed filter used to estimate speed from a low-resolution encoder. When the speed filter is included in the model there is a good match between the experimental data and the model. However, this injects a second order system into the originally simple model, complicating the lessons being taught to the students. Although the data from the model and actual system, in Figure 4.6, do not match at low speed, this is due to the nonlinear friction not a higher order model. The low cutoff frequency for the speed filter on Eeva in required because of the low resolution encoder, which can be seen in the unfiltered data. The unfiltered, $\frac{\Delta\theta}{\Delta t}$, signal representing a crude speed measurement appears to match the simple (not including the speed filter) model solution as shown in Figure 4.7.

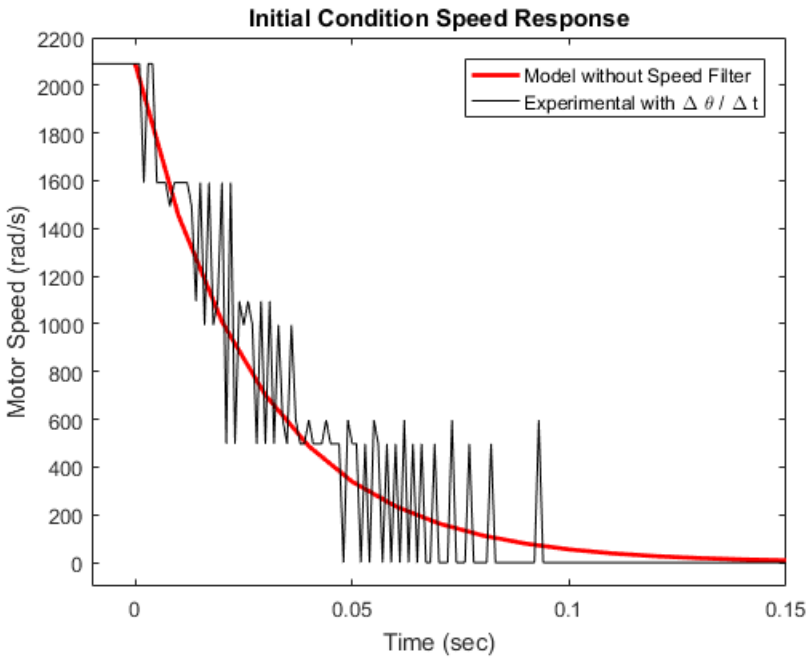


Figure 4.7: Eeva Unfiltered Initial Condition Response

Results:

It is possible to develop a model for the speed response on Eeva and use it to estimate the friction coefficient. However, the process is complicated by the back EMF effects and the need for a low-frequency filter on the speed measurement. These complications are likely to confuse the students and obscure the lessons from the lab. This is especially true early in the semester when the students are struggling with basic modelling concepts. It is difficult, if not impossible, to design a lab as simple as the one used for the MotorLab discussed at the beginning of this Chapter.

Chapter 5 - Experiment II – Second Order Responses and the Impact of Gain on Pole Locations

This lab was originally designed to use a proportional position controller for the MotorLab to demonstrate second order responses and the impact of changing gain. Part of the handout for this original lab, showing a position controller, is given in Figure 5.1. Students would observe that increasing the proportional gain would speed up the response and increase disturbance rejection, but the oscillations would grow. They would then be able to look at the closed loop model and observe that changing the proportional gain was increasing the complex component of the poles, which predicts the increased oscillations in the systems response. The oscillation frequency and decay rate of the oscillations match those predicted from the model very well.

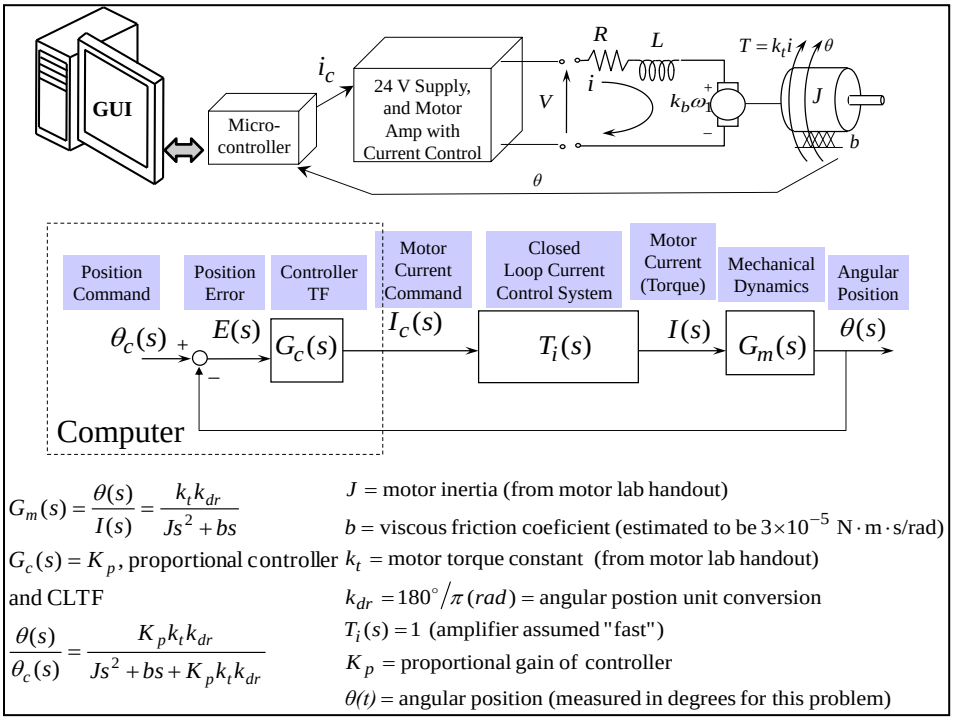


Figure 5.1: Original Second Order Lab Model

However, when this lab was attempted on Eeva, the static friction made it extremely difficult to demonstrate a second order response with the ability to measure oscillation frequency. At lower step commands the higher friction on the brushed motors damped any oscillations immediately, and at higher step commands the voltage saturated, changing the response.

In order to combat this, another second order system is considered. We went to pure integral control on speed to attempt to find something that would show standard second order oscillations and enough range on the input to allow several gains to be used without saturation. This solution was not perfect, the pure integral control has to build up to overcome static friction, leading to a time delay and an amplitude difference between the model and the real system.

MotorLab:

For the MotorLab the system is the same simple model from the first lab, only we are now controlling speed with an integral controller. The closed loop model with the transfer functions for the integral controller and plant model from equation (1) in chapter 3 is given in Figure 5.2.

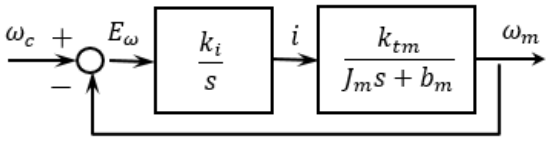


Figure 5.2: MotorLab Closed Loop Model

First students look at two different gains in the MotorLab for integral control on speed. The response from a lower gain is shown in Figure 5.3 along with the response predicted by the model. Even with the integral build up to overcome static friction at start of the response, the model still predicts the low gain response very accurately as shown in Figure 5.3. The

experimental model also shows easy to measure oscillations that students can relate to the model parameters. This response also allows a clear picture of how integral control builds up over time.

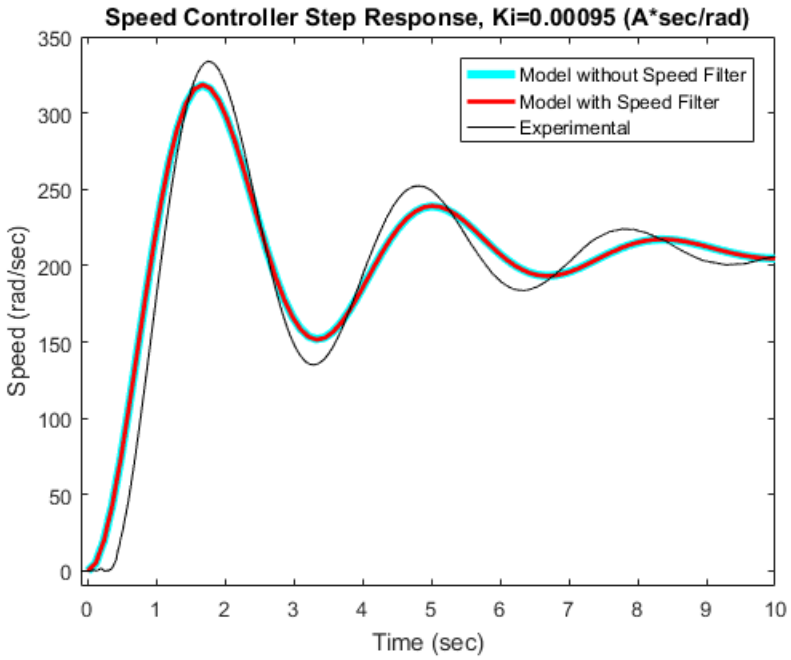


Figure 5.3: MotorLab Integral Control Lower Gain Speed Response

In the higher gain system on the MotorLab the integral build up is less obvious, although the phase and amplitude differences appear with increasing intensity as time goes on as shown in Figure 5.4. It is also clear that increasing the gain leads to more oscillations, which is predicted by the simple model the students work with. Of note also is that in both MotorLab systems, the dynamics of the speed filter do not affect the observed response, simplifying the model that the students need to work with to learn the concepts of the lab.

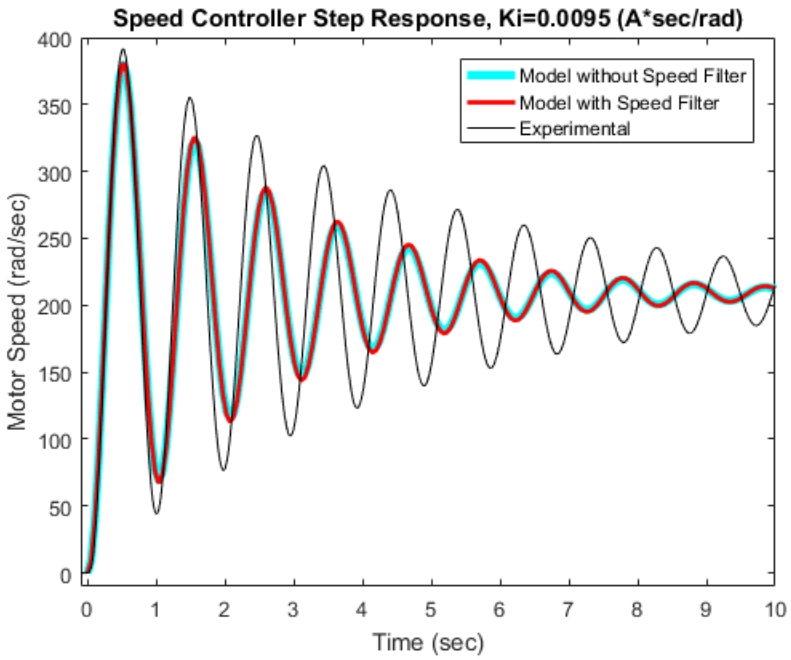


Figure 5.4: MotorLab Integral Control Higher Gain Speed Response

Eeva:

The closed loop model for speed control on Eeva is given in Figure 5.5. Pure integral control is used and the transfer functions for the plant model and speed filter from chapter 3 (Equations (5) and (6) respectively) are given again here for clarity.

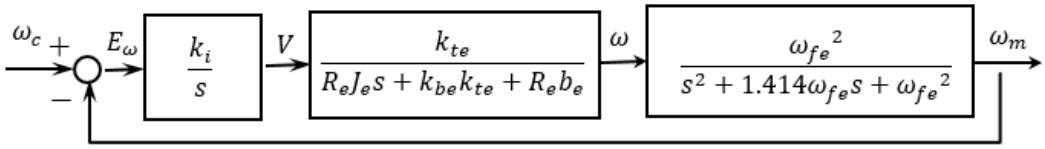


Figure 5.5: Eeva Closed Loop Speed Model

When working with Eeva the effects of the inexpensive hardware (noise, encoder resolution, etc.) are clear. The oscillations from the gains are evident even through the encoder noise. However, it is clear that including the speed filter is necessary to capture full effects we are observing. The lower gain in Figure 5.6 has a definite larger static friction than the MotorLab, making the relationship between the model and the experimental data harder for students to see.

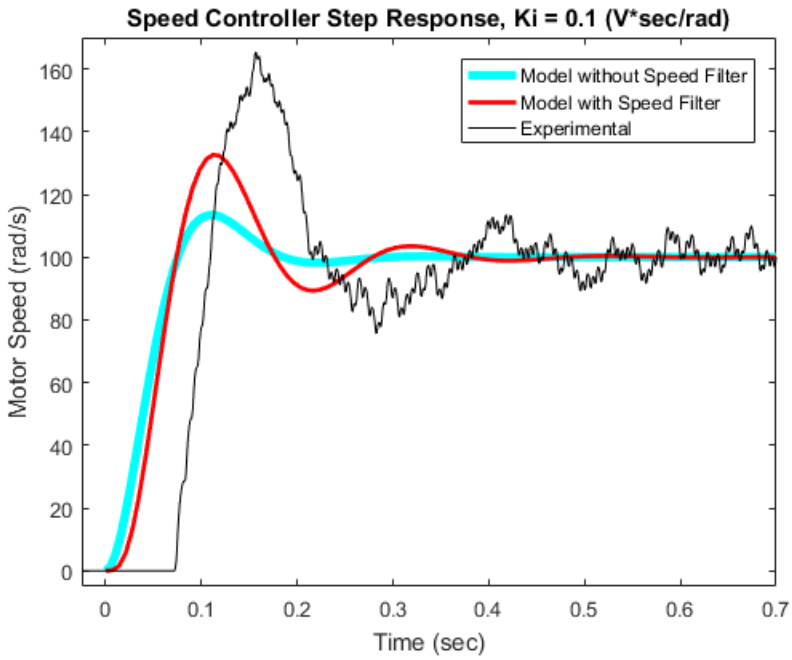


Figure 5.6: Eeva Integral Control Lower Gain Speed Response

For the higher gain response shown in Figure 5.7 the effect of the speed filter is even more obvious. The model without the speed filter greatly underestimates the overshoot and oscillations. For any reasonable match between experimental and theory the full fourth order model must be considered.

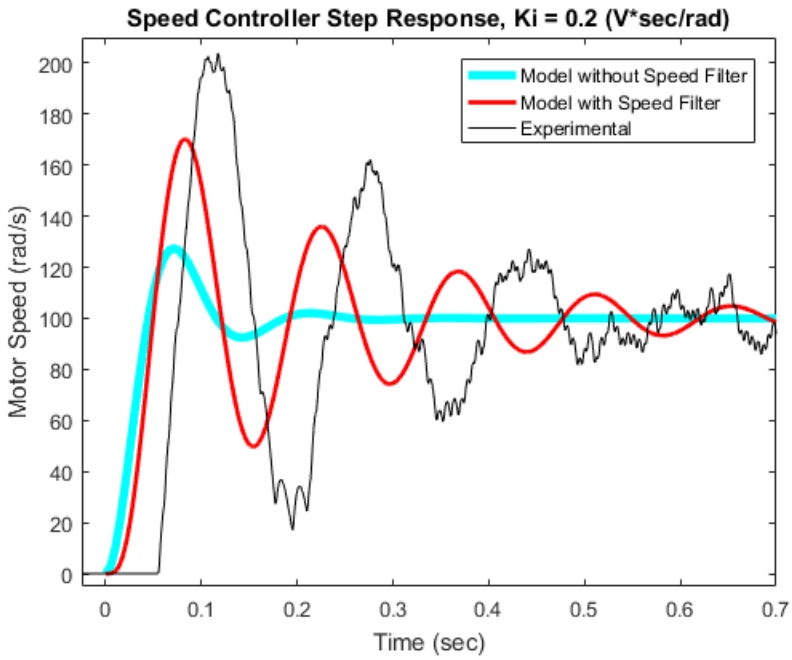


Figure 5.7: Eeva Integral Control Higher Gain Speed Response

Results:

The speed control lab on Eeva demonstrates the relationship between changing gain and system response. Students can also relate the real system performance to the pole locations. However, in order to get good agreement between the real system response to the model students have to include the speed filter in our model, making the model fourth order instead of second order. Integral build up to overcome static friction also causes a delay and amplitude difference that further distorts the relationship between the model and real response.

Chapter 6 - Experiment III – PI Controller and System Type

For this lab the goal is to demonstrate the importance of system type and controller choice on our system's response. The motor speed system has no free integrators, meaning it has a system type of zero. This means that there should be steady state error in the response to a step input. Proportional control can help speed up the response, but it will not increase the system type, meaning the steady state error will not go to zero. PI control adds a free integrator, increasing the system type to one, which will lead to zero steady state error. As shown in Table 6.1 we can expand the concept of system type to several other input types, which may more closely approximate the expected real world input.

Table 6.1: Steady State Error

	System Type = 0	System Type = 1	System Type = 2
Step Input	Finite	0	0
Ramp Input	Infinite	Finite	0
Parabolic Input	Infinite	Infinite	Finite

MotorLab:

The closed loop model for speed control on the MotorLab is given in Figure 6.1. It is the same system used in chapter 5, only the controller is now either proportional or proportional-integral, meaning $G_c = K_p$ or $G_c = \frac{K_p(s+z)}{s}$ with $z = K_i/K_p$. The transfer functions for the MotorLab speed model and speed filter shown in chapter 3 are presented again here for clarity.

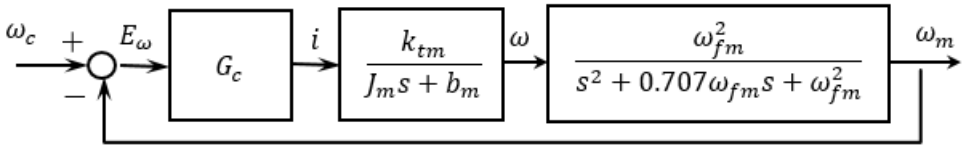


Figure 6.1: MotorLab Closed Loop Speed Model

Students take three sets of data, two pure proportional controllers and one PI controller. The gains for this lab are high enough where the speed filter has to be considered in the

dynamics of the model to accurately represent the real system behavior. However, at this point in the semester the students have progressed significantly compared to their knowledge during the previous two labs discussed. Therefore, it is appropriate to introduce the higher order models and to begin to consider the effects of the higher frequency dynamics. Both proportional gains have a small amount of steady state error, and increasing the gain only increases the oscillations. The higher proportional gain plot is given in Figure 6.2. Notice that the speed filter dynamics are required to model the system behavior accurately for the MotorLab for this gain.

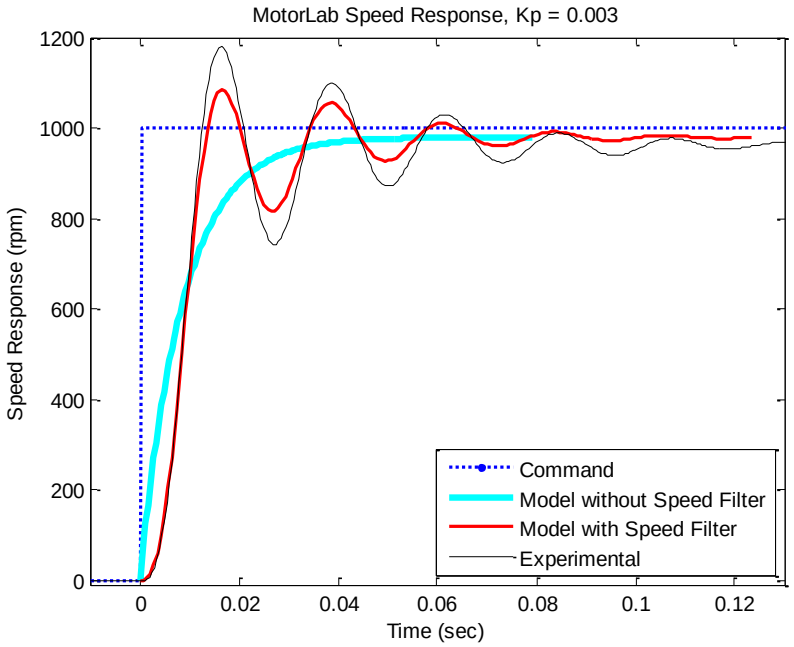


Figure 6.2: MotorLab Proportional Speed Controller

For the final step in the lab the students increase the system type by using PI control, which adds a free integrator. This improves the steady state tracking without driving the system close to instability. With the addition of integral control we also can use a lower proportional gain which can decrease the oscillations while still giving a similar settling time. The plot for the PI controller in the MotorLab is given in Figure 6.3.

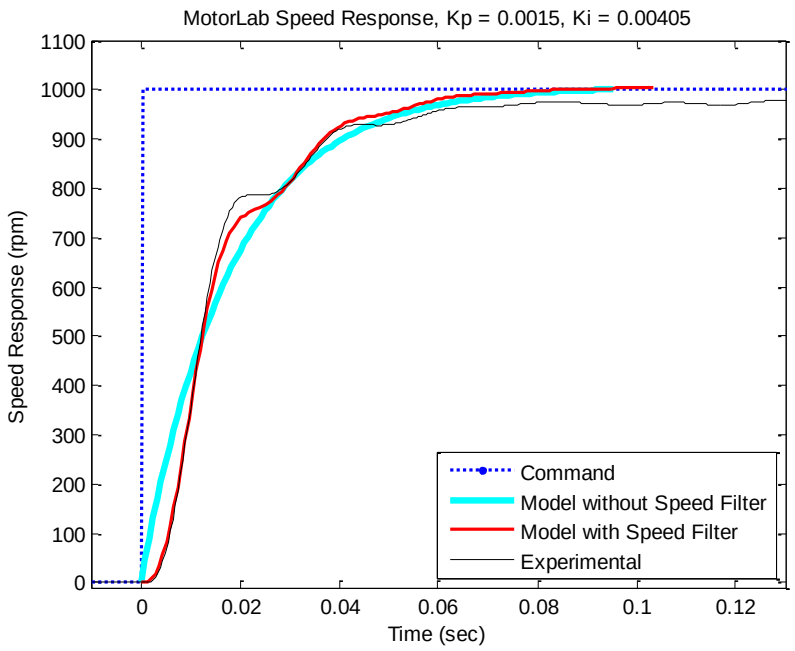


Figure 6.3: MotorLab Proportional Integral Speed Controller

Eeva:

The closed loop model for Eeva is given in Figure 6.4. It is the same system described in Chapter 5, only now both proportional and proportional-integral speed control are used. The transfer functions for the speed model and speed filter from chapter 3 are given here again for clarity.

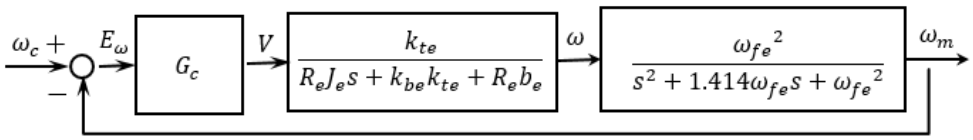


Figure 6.4: Eeva Closed Loop Speed Model

On Eeva we start the investigation with a pure proportional controller on speed. The speed input was set to about 500 rad/s to avoid any saturation issues after trying a few higher commands that did saturate. The goal is to see an observable steady state error and fair agreement between the model and the experimental data. For this lab it is not a problem for the students' comprehension to include the speed filter because they are learning about the effects of

the higher order dynamics. The response with a high proportional gain is shown in Figure 6.5. It demonstrates the steady state error we wanted to see as well as fairly good agreement between the model and the data. The steady state error is actually much more visible on Eeva than it was on the MotorLab, making the benefits of using integral control more obvious to students.

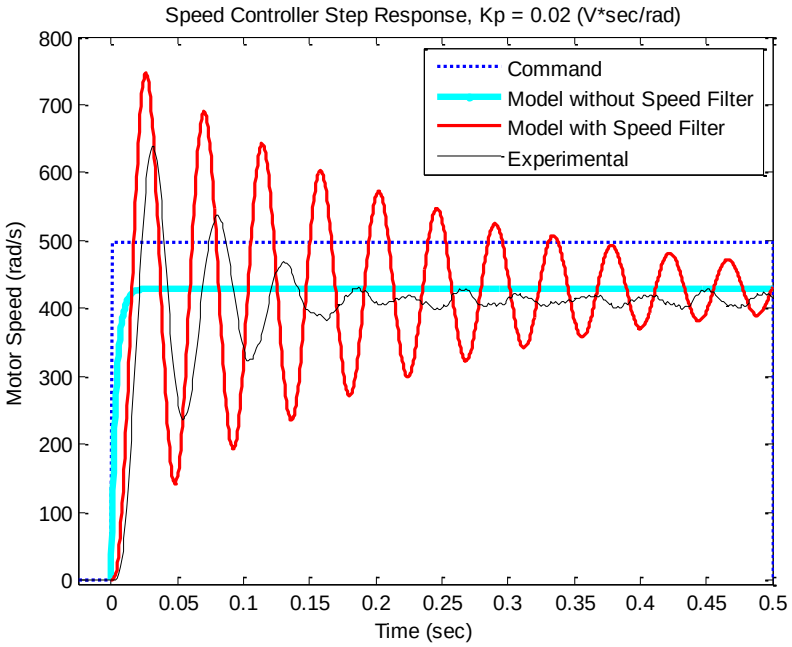


Figure 6.5: Eeva Proportional Speed Controller

Next is the PI controller where Eeva's response is shown in Figure 6.6. The steady state error clearly goes to zero and there is even better agreement between the model and the experimental data than there was with the pure proportional controller. There are also less oscillations, although the settling time is about the same. This plot should clearly demonstrate the value of increased system type and the effect of PI control on tracking to the students.

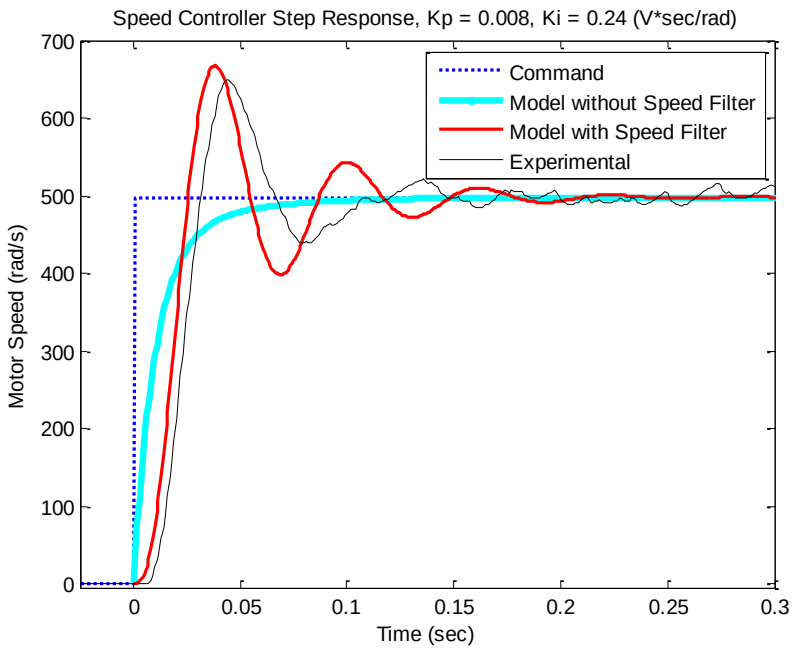


Figure 6.6: Eeva Proportional Integral Speed Controller

Results:

For the system type lab Eeva seems to demonstrate the desired concepts very comparably to the MotorLab. The steady state error on Eeva is greater than it is on the MotorLab, making the benefits of using integral control more obvious to the students. While the encoder resolution and inexpensive sensors do make more noise, the scale of this lab prevents that from obfuscating the desired concepts.

Chapter 7 - Experiment IV – Frequency Response and System Parameter Estimations

The goal of this lab is to introduce frequency response. A simple dynamic system with resonance is useful to introduce students to frequency response. This lab was designed to investigate resonance in frequency response as well as the process for using frequency response and a model to estimate system parameters. In the introductory control systems course at K-State this lab takes place as students are being introduced to frequency response. At this point in the semester they know how to calculate magnitude and phase from a transfer function and are learning to sketch asymptotic Bode plots.

MotorLab:

For the MotorLab a dynamic system is used where the motor is attached to a spring, as shown in Figure 7.1 which is taken from the handout for the laboratory. The transfer function given in this figure is the same as equation (4), with minor differences in the symbols used. One side of the spring is tied down, meaning the input torque of the motor will balance against the torsion spring's torque in steady state.

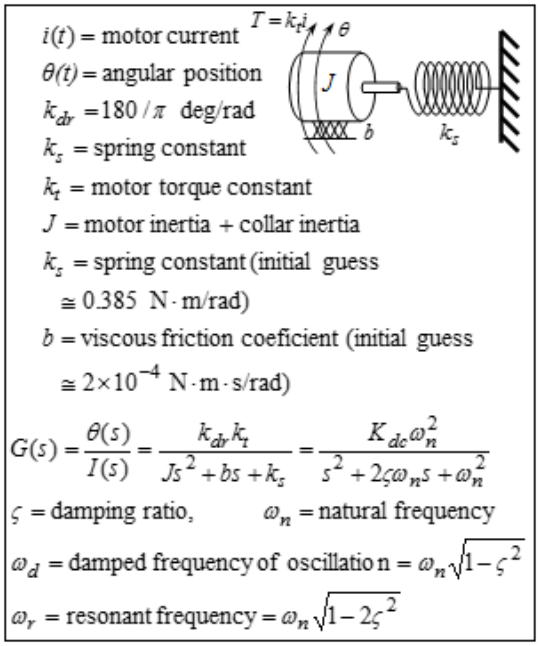


Figure 7.1: MotorLab Resonance Frequency Lab

The lab begins by searching for the resonant frequency of the system. Sine waves are commanded for the current while the sinusoid response of the motor angle is recorded. To find the resonant frequency, the students search for a response where the position lags the current by 90°. This is calculated using the zero crossing of the command, $t_{crossing}$, and the corresponding peak and valley of the response, t_{peak} and t_{valley} respectively. The commanded frequency and magnitude, f_c & mag_c, are also known, allowing calculation of the phase lag and magnitude ratio as shown in Figure 7.2. First the time lag is calculated from the time data from the cursors:

$$t_{lag} = \frac{abs(t_{peak} - t_{valley})}{2} - t_{crossing}.$$

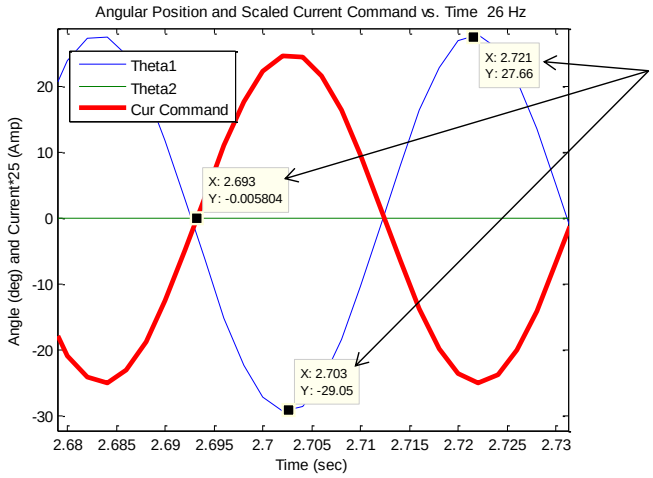
Then the phase lag can be calculated from the time lag and the input frequency:

$$phase\ lag = 360 \cdot t_{lag} \cdot f_c.$$

The magnitude ratio is calculated from amplitude data of the cursors and the input magnitude:

$$magnitude\ ratio = \frac{y_{max} - y_{min}}{2} \cdot \frac{1}{mag_c}.$$

All of these calculations are performed in a MATLAB function called ‘calc_mag_phase’ provided for students to use.



Use `mlolplots()` to plot the data.
Make three data cursors: one for a zero crossing of the current command, and two the peak and valley of the corresponding crossing of the output. Right click on one of the cursors and choose “Export Cursor Data to Workspace.” Then run, for example, `calc_mag_phase(cursors,26,2)`.
Here the cursor data was saved to a variable named ***cursors***, the input frequency was 26 Hz, and the input amplitude was 2 Amp. Also, in this example the phase lag was about 180 degrees.

Figure 7.2: Finding Resonance Frequency

When finding the resonance frequency (ω_n) of the system students are instructed to use a small current amplitude to avoid excessive displacements of the spring that could lead to cyclic fatigue failure. It is necessary to increase the amplitude of the command at higher frequencies to achieve enough displacement to measure accurately and consistently. A table showing a typical data set on the MotorLab is given in Table 7.1 while a subset of figures with cursors for that data set is provided in Figure 7.3. These are provided to compare the consistency and reliability of data collection with a similar process for Eeva given later on in this chapter.

Table 7.1: MotorLab Resonance Frequency Test Data

Freq. (Hz)	ω_n	$\omega_n/10$	$0.75 \cdot \omega_n$	$1.25 \cdot \omega_n$	$2 \cdot \omega_n$
Input Amp. (Amp)	0.25	1	1	1	2
Freq. Value (Hz)	20.98	2.1	15.74	26.23	41.96
Mag. Ratio (deg/Amp)	317.25	12.29	30.26	27.90	4.95
Mag. Ratio (dB)	50.03	21.79	29.61	28.91	13.89
Phase Shift (deg)	-93.53	-3.68	-9.53	-174.69	-181.39

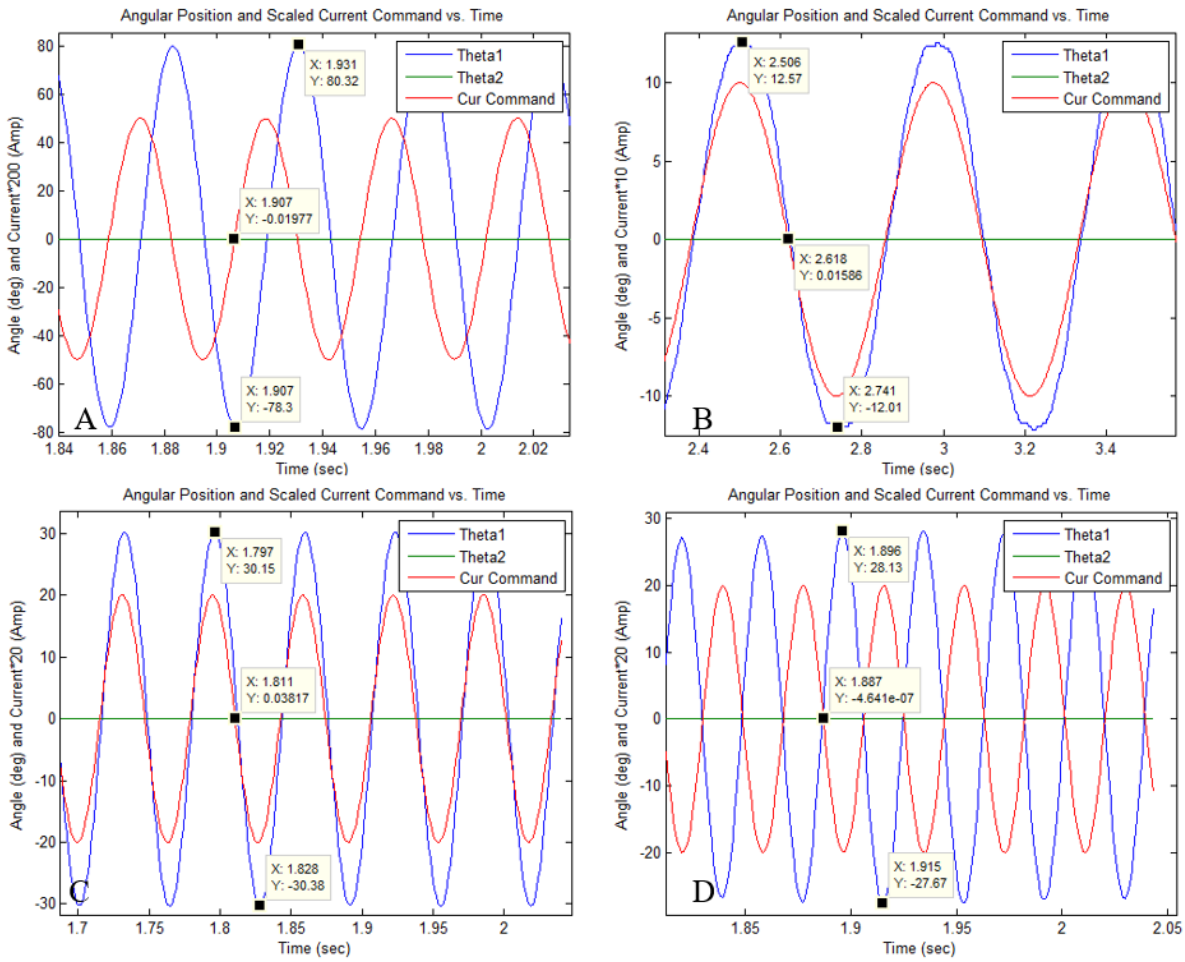


Figure 7.3: MotorLab Data for A: ω_n, B: $\omega_n/10$, C: $0.75 \cdot \omega_n$, & D: $1.25 \cdot \omega_n$

Once students have collected the data, they can use that data to improve their parameter estimates using the model transfer function given in equation (6) and relating it to the standard second order form for transfer functions given below.

$$\frac{k_{dc}\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

The resonant frequency is ω_n, k_{dc} is the magnitude ratio at one-tenth of the resonant frequency, and ζ is found from dividing the dc gain by one half of the magnitude ratio at the resonant frequency. Once these values are known, students can estimate the system parameters using the

following relationships between the standard second order form and the transfer function from equation (6).

$$k_s(spring\ constant) = J\omega_n^2$$

$$b(friction\ coefficient) = 2\zeta\omega_n J$$

$$k_t(motor\ torque\ constant) = k_{dc}J\omega_n^2 k_{dr}$$

With the new estimates for system parameters, students can investigate how well their new model matches the data taken. Not surprisingly, the model fits nicely since the coefficients have been calculated to fit the data. The initial model with estimates for the parameters is compared to the experimental data and the improved model in Figure 7.4.

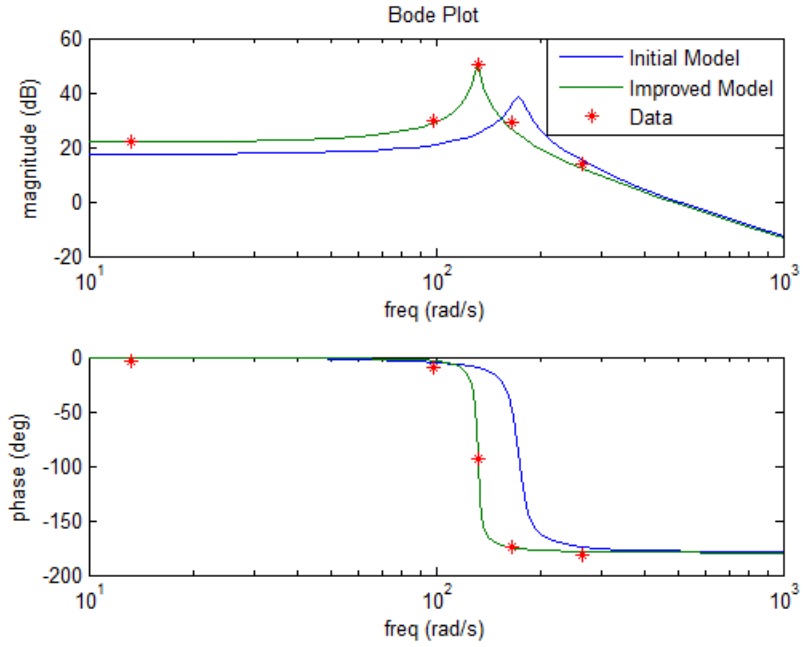


Figure 7.4: MotorLab Final Bode Plots

Eeva:

On Eeva there are no springs, so Eeva is configured as a hanging pendulum to produce a simple system with resonance. The model is more complex than the simple one for the MotorLab, but it is still second order. This model is shown in Figure 7.5 with the transfer functions from chapter 3 given for clarity.

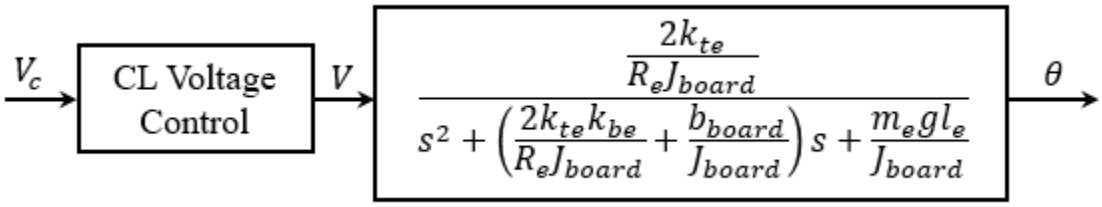


Figure 7.5: Eeva Open Loop Position Model

A set of vises is used to secure the wheels of Eeva to allow it to hang and swing freely with about 60 degrees of freedom before hitting the desk. In Figure 7.6 there is a picture of the setup with Eeva in place.

Figure 7.6: Eeva Inverted Pendulum Setup

There are a number of possible complications with this setup that could impact the agreement between the real data and the model. First, there is encoder quantization that only provides one degree of resolution for the tilt of the robot. Second, if displacements are increased to combat the low resolution encoders, the small angle approximation in the model will start to give larger errors. Third, the motors are attached to the board with screws on the outside, which clamps the motor down against the board. This causes a slight angle relative to the board, so as the board rotates the wheels bow in and out, causing some nonlinear resistance to the board rotating. Finally, there is the un-modeled nonlinear dynamics of the gearboxes.

This way of producing a second order underdamped system with resonance needs to consistently provide a useful lab to help demonstrate the control systems concepts we are looking to show students. Therefore, several sets of data were taken to investigate whether the results were consistent. In each case the input magnitudes were varied to see if the observed frequencies remained consistent. The average values used are provided in Table 7.2, while magnitude and phase plots for all four sets of data and the average are provided in Figure 7.7 and Figure 7.8 respectively.

Table 7.2: Eeva Average Resonance Frequency Test Data

Freq. (Hz)	ω_n	$\omega_n/10$	$0.75 \cdot \omega_n$	$1.25 \cdot \omega_n$	$2 \cdot \omega_n$
Input Amp. (V)	1.288	2.750	1.213	1.288	2.750
Freq. Value (Hz)	1.688	0.169	1.266	2.109	3.375
Mag. Ratio (deg/V)	27.723	10.617	29.023	17.133	4.673
Mag. Ratio (dB)	28.565	20.403	29.055	24.433	13.260
Phase Shift (deg)	-88.285	-31.035	-61.028	-120.003	-150.943

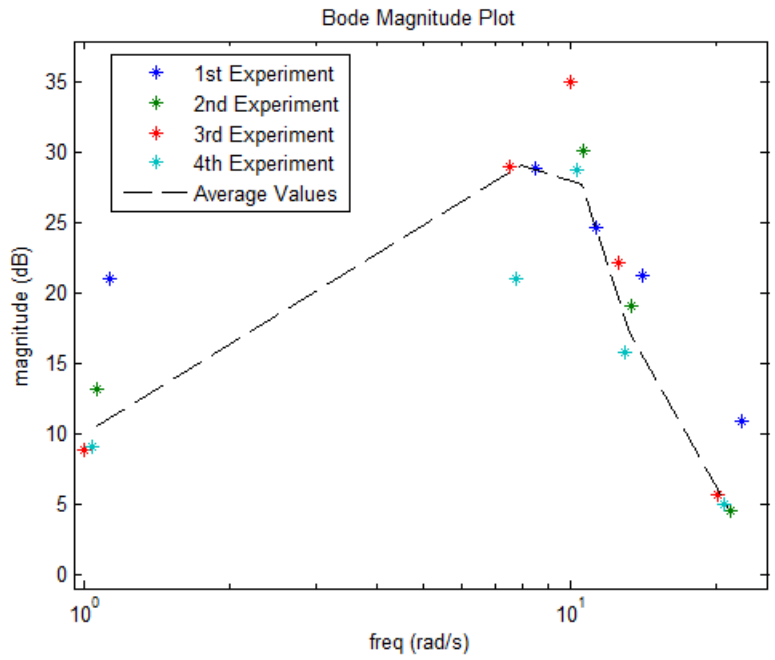


Figure 7.7: Eeva Resonance Frequency Magnitude Test Data

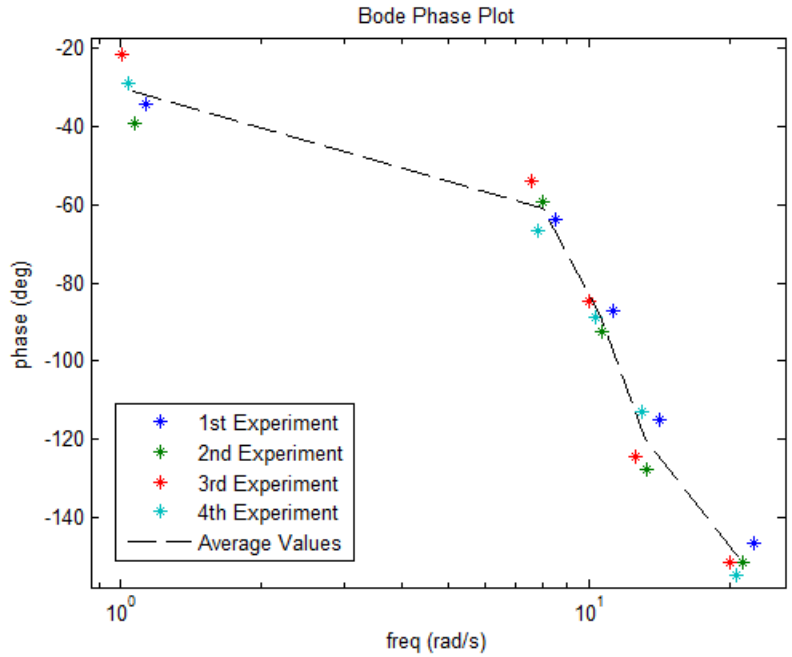


Figure 7.8: Eeva Resonance Frequency Phase Test Data

The magnitude data for Eeva has some fluctuations, but since the magnitudes were widely varied among the tests, some variation was expected. In the phase plot there appears to be good consistency among the results. Data collection was performed exactly as it was done on the MotorLab, with the same calculations to find the phase lag and magnitude ratio. In Figure 7.9 a sample of the plots that provided the values for these calculations are provided. It is worth noting that this process was much more difficult and unclear than it was on the MotorLab.

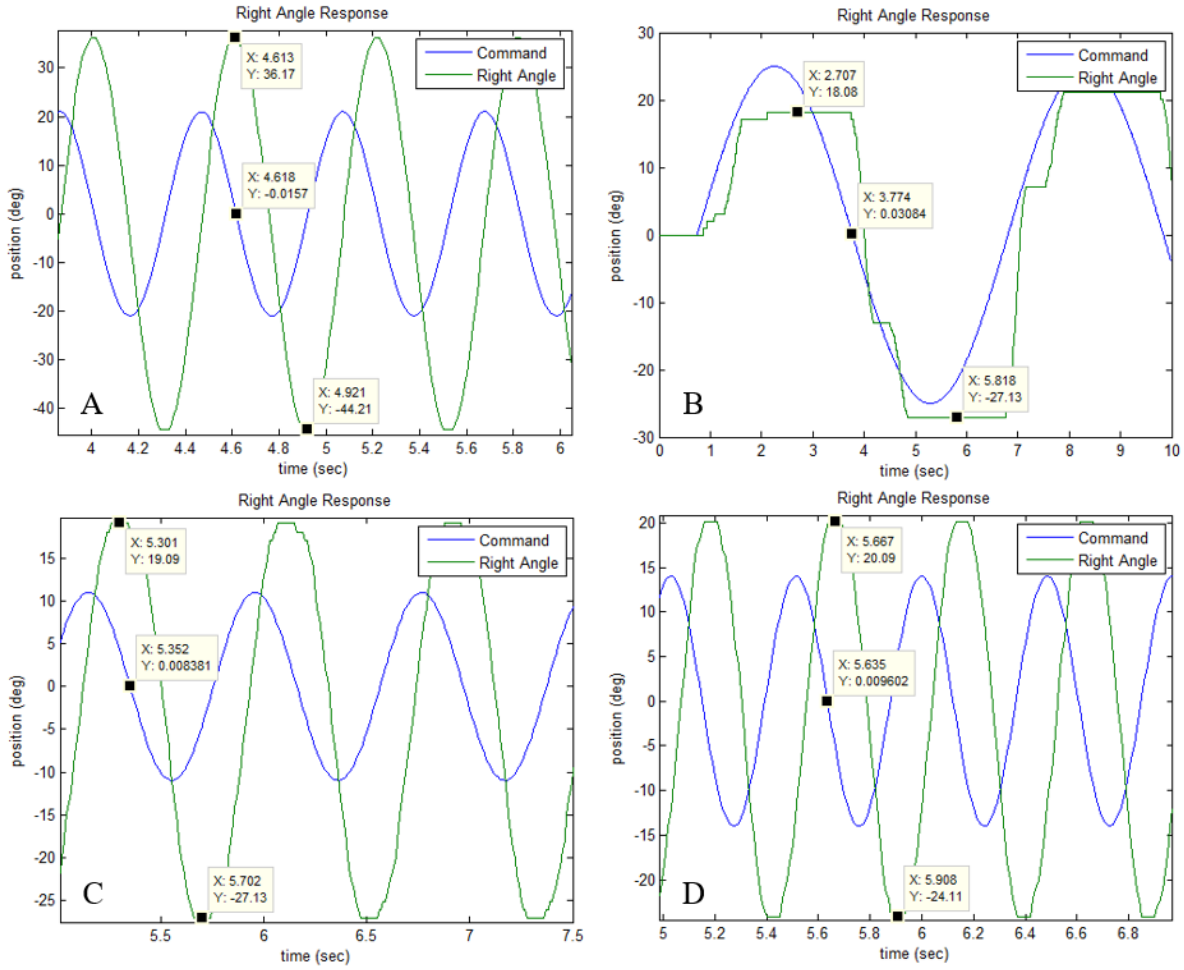


Figure 7.9: Eeva Data for A: ω_n, B: $\omega_n/10$, C: $0.75 \cdot \omega_n$, & D: $1.25 \cdot \omega_n$

Eeva has an encoder resolution of 12 counts per motor revolution. With the 30:1 gear ratio, that means the robot's tilt angle in the pendulum setup has a resolution of only one degree. This accounts for some of the jagged edges and plateaus observed in the data collected. At one tenth of the measured resonant frequency the observed motion of the robot was abrupt and staggered, while at higher frequencies it was observed to move in a smoother, more sinusoid fashion as expected. While the lab is continued just as with the MotorLab, it is clear that there may be issues with the dc gain calculated from one-tenth of the resonant frequency due to the jerky motions and unclear data from the graph.

From comparing the model given in equation (7) to the standard second order form for a transfer function the following estimates for system parameters can be made.

$$J_{board}(moment\ of\ inertia\ of\ board) = \frac{m_e g l_e}{\omega_n^2}$$

$$b_{board}(viscous\ friction\ coefficient) = 2\zeta\omega_n J_{board} - \frac{2k_{te}k_{be}}{R_e}$$

With these new estimates a comparison between the data and the model can be made. This experiment was used to measure the moment of inertia for the board as well as the coefficient of friction for the board's movement. Therefore, no initial estimated model is given for comparison with the fitted model's results in Figure 7.10.

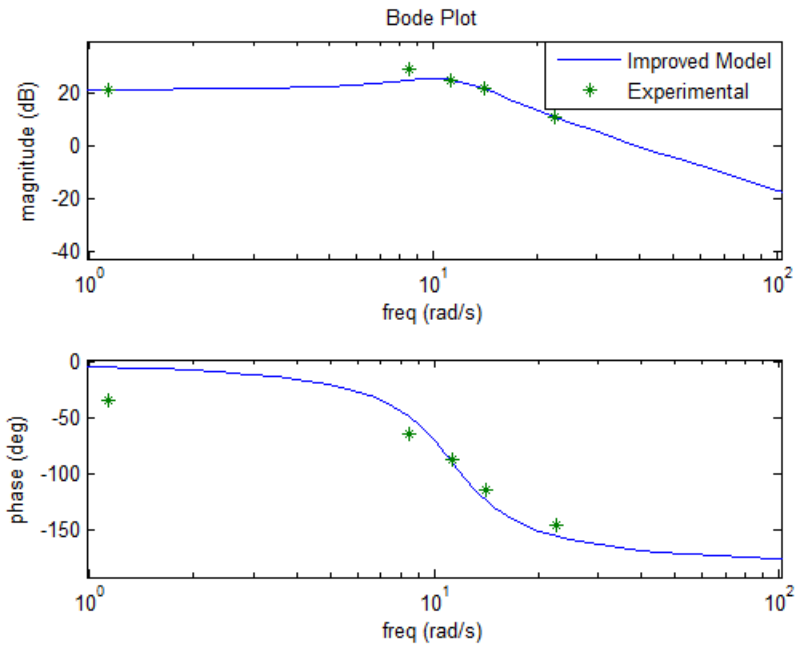


Figure 7.10: Eeva Final Bode Plots

The final bode plot shows good agreement between the model and data, especially given the uncertainty concerning the dc gain measured from the rough plot at one-tenth of the resonant frequency.

Results:

While a resonant frequency response was successfully generated on Eeva in the pendulum setup, the encoder resolution and unexpected jerky behavior at low frequencies clearly obfuscate the process for students. The model used, while only second order, has many more physical coefficients than the model of the MotorLab, complicating the calculations to demonstrate the control systems concepts in this lab.

Chapter 8 - Conclusions

Four labs were investigated on the inexpensive hardware on Eeva and compared with the results on the MotorLab used for the past fifteen years as the hardware for the introductory control systems course at Kansas State University. Voltage control led to more complicated models since back EMF effects had to be included. Additionally, the low resolution encoders also complicated the models since the dynamics of the speed filter had to modeled as well when doing speed control as shown in Chapters 4 and 5.

In the system type lab discussed in Chapter 6 the inexpensive hardware on Eeva performed extremely well compared to the MotorLab, since the greater friction and losses in the system made the benefits of PI control more obvious. The imperfect dynamics on Eeva also allow for opportunities to discuss real world effects such as quantization, saturation, and sensor noise.

In the resonant frequency lab discussed in Chapter 7 Eeva's hardware did not compare favorably to the MotorLab. Inconsistencies in data collection and a more complicated model served to obfuscate the introduction to frequency response.

When designing inexpensive hardware to teach a control systems course it is important to balance equipment cost with the performance necessary to demonstrate the desired concepts clearly and consistently to students. Better hardware also allows higher operating limits, making the laboratory design much easier on the instructor.

References

[1] D. H. Alcantara, R. Morales-Menendez and R. A. R. Mendoza, "Teaching semi-active suspension control using an experimental platform," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 7334-7339. doi: 10.1109/ACC.2016.7526830

[2] C. J. Bay and B. P. Rasmussen, "Exploring controls education: A re-configurable ball and plate platform kit," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 6652-6657. doi: 10.1109/ACC.2016.7526718

[3] C. C. Bissell, "Control education: time for radical change?," in IEEE Control Systems, vol. 19, no. 5, pp. 44-49, Oct 1999. doi: 10.1109/37.793440

[4] T. Emami and J. Benin, "Computer support for teaching the Routh-Hurwitz criterion," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 1329-1334. doi: 10.1109/ACC.2016.7525102

[5] B. Ferri and A. Ferri, "A controls approach to improve classroom learning using cognitive learning theory and course analytics," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 7321-7327. doi: 10.1109/ACC.2016.7526828

[6] R. Krauss and J. Croxell, "A low-cost microcontroller-in-the-loop platform for controls education," 2012 American Control Conference (ACC), Montreal, QC, 2012, pp. 4478-4483. doi: 10.1109/ACC.2012.6314807

[7] R. Krauss, "Combining Raspberry Pi and Arduino to form a low-cost, real-time autonomous vehicle platform," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 6628-6633. doi: 10.1109/ACC.2016.7526714

[8] A. S. Miller, W. Singhose and U. Glauser, "Integrating PLC theory and programming into advanced controls courses," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 7302-7307. doi: 10.1109/ACC.2016.7526825

[9] R. M. Reck, R. S. Sreenivas and M. C. Loui, "Assessing an affordable and portable laboratory kit in an undergraduate control systems course," Frontiers in Education Conference (FIE), 2015. 32614 2015. IEEE, El Paso, TX, 2015, pp. 1-4. doi: 10.1109/FIE.2015.7344319

[10] R. M. Reck, "Defining common aspects of undergraduate instructional laboratories for control systems," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 6646-6651. doi: 10.1109/ACC.2016.7526717

[11] R. M. Reck and R. S. Sreenivas, "Developing a new affordable DC motor laboratory kit for an existing undergraduate controls course," 2015 American Control Conference (ACC), Chicago, IL, 2015, pp. 2801-2806. doi: 10.1109/ACC.2015.7171159

[12] S. D. Ruben, "Respect the implementation: Using NI myRIO in undergraduate control education," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 7315-7320. doi: 10.1109/ACC.2016.7526827

[13] D. Schinstock, S. Schinstock and W. N. White, "Micro-controller based update of inexpensive undergraduate control systems laboratory hardware," 2015 American Control Conference (ACC), Chicago, IL, 2015, pp. 2807-2812. doi: 10.1109/ACC.2015.7171160

[14] N. L. Toner and G. B. King, "Restructuring an undergraduate mechatronic systems curriculum around the flipped classroom, projects, LabVIEW, and the myRIO," 2016 American Control Conference (ACC), Boston, MA, 2016, pp. 7308-7314. doi: 10.1109/ACC.2016.7526826

[15] M. W. Spong, "Control education crossing department boundaries," American Control Conference, 1999. Proceedings of the 1999, San Diego, CA, 1999, pp. 992-996 vol.2. doi: 10.1109/ACC.1999.783189

[16] Wright, Ann M.; Wright, Andrew B. An inexpensive dynamic system for teaching measurement and controls. 2013, 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013

Appendix A - Original MotorLab Assignments

Experiment 1:

In this lab you are to experimentally determine an approximation for the viscous friction coefficient for the motor of the Motorlab. Then you are to use this coefficient to predict the response of the dynamic system to an initial condition (initial velocity) and compare this to the actual I.C. response. If the spring coupling is removed from the Motorlab apparatus, then we are left with the dynamic system described by the equations and schematic model to the right. This is the dynamic system studied in this laboratory. Also in this lab you will be continue to learn to use MATLAB.

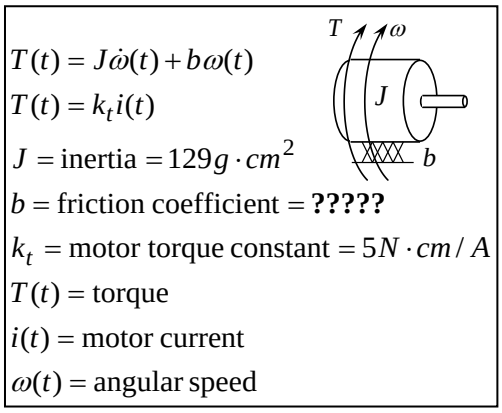


Figure A.1: Lab 1 System

ONE REPORT is due from each group, but you all are responsible for understanding what is in the report and how it was generated.

You are to set the "Controller Mode" in the motorlab GUI to "Open Loop" to acquire experimental data for this lab. This program does not implement closed-loop control of the Motorlab mechanical hardware. It allows you to manipulate the input to the mechanical system, the motor current (torque), and acquire data.

You are also to complete the MATLAB m-file that will generate the plots required for this lab. You are given most of the m-file code on the following page. You may copy this code out of this document and past it into an m-file to modify it. You should use the help in MATLAB and your instructor to continue to learn the language.

Estimating the Viscous Friction Coefficient

Looking at the differential equation in the model it can be seen that if a constant torque (current) is input then in steady state, where $\dot{\omega}(t) = 0, T(t) = b\omega(t)$. Therefore, we should be able to estimate the viscous friction coefficient by obtaining steady state velocity and current data.

Change the motor current command using the jog buttons then save the data to the workspace after sufficient time for the buffer to fill. Use the following two commands in the command window to get the average current and speed:

```
mean(data(:,8))
mean(data(:,5))
```

Fill in the table below. In this lab you will probably discover that friction is often a hard thing to model and that the linear, viscous friction, model is not completely accurate in some cases. We are attempting to find an "engineering estimate" that might be used in closed loop control where completely accurate models are not necessary.

Table A.1: Speed vs Current Data

Current Command (A)	-0.14	-0.12	-0.10	-0.08	-0.06	-0.03	0
Avg. Current (A)							
Avg. Speed (rpm)							
Current Command (A)	0.03	0.06	0.08	0.10	0.12	0.14	--
Avg. Current (A)							
Avg. Speed (rpm)							

Using the data from the table above obtain a plot of torque vs. angular velocity. On this same plot, draw the "best fit" line through the data by playing with the slope of the line described by $T = b_{estimate}\omega$. Generate this plot by completing the top of the m-file provided and running it with successive guesses at $b_{estimate}$.

Comparing Theoretical and Experimental IC Responses

Now that you have an estimate of b YOU ARE TO USE IT TO SOLVE THE DESCRIBING DIFFERENTIAL EQUATION given that $T(t) = 0$ and given that the initial condition (IC) is $\omega(t = 0) = \omega_0$. Do this by hand, paying close attention to units. You are also to obtain data for the IC response from the actual system, and then compare this with the theoretical responses on the same plot. To obtain the response to an IC use a current of 0.15 A to generate an initial velocity. Using a sample frequency of 500 Hz do the following:

1. Jog the current to the desired level, or type it into the command window, and allow the velocity to settle, and wait at least four seconds for the data buffer to fill.
2. Hit the "Turn Off Motor Amp" button to turn off the current – count to 3 seconds – then immediately hit the "Save Data Buffer to Workspace" button. This should give data with about one second of initial velocity and 3 seconds of IC response.
3. In MATLAB and view the data with "mlolplots()". In figure (2) of MATLAB you should see the IC response. Zoom in and use the cursor to find your actual initial velocity and a time at which the torque was set to zero.
4. Complete the bottom part of the m-file provided so that it generates the experimental response and the theoretical response on the same graph.

Things to Turn In

- You should have two different plots: 1) torque vs. angular velocity, 2) IC response
- You should have code for the m-file completed.
- Hand development of the solution to the differential equation with the I.C.
- Fill in the blanks below. (This must be turned into your instructor before they leave the lab or other arrangements made with them. Attach another copy, which may be different/corrected, to your report).

Fill In The Blanks: All answers should have units where appropriate. This is your chance to learn. So think about your answers, find as many connections as you can, and try to extrapolate. **You should copy this out of the given word file and fill in the blanks with BOLD face type and underlined.**

Lab #2 QUESTIONS Names

In the first plot we can see the relationship between motor speed and the friction torque. With a constant motor current, the motor ?????? is constant, and the speed settles to a fairly constant value where the input torque balances with the ?????? torque. Our typical model of friction for control design is ??????, with the friction torque being proportional to velocity. However, the data points in the plot show that the actual friction has a ?????? component along with the linear component, resulting in a zero velocity with small values of constant torque. Our estimate of the viscous (linear) coefficient of friction roughly capturing both of these effects is ??????(units).

In the second plot we see the actual initial condition response along with one from the model, which was found from the solution of the ?????? equation with an initial condition. The time constant the linear model can be found with the mass moment of inertia and our estimate of the friction coefficient. It has a numerical value of ??????(units). At one time constant the linear is model is at exactly ??????(rpm) (ignoring roundoff), which is 37% of the initial value. The actual data from system is at a value of ??????(rpm) at the time constant. The major difference between the linear model and the actual system is at ?????? speed where the model significantly underestimates the friction torque. It can be seen that as the motor slows down the velocity decay is much faster than predicted by the ??????.

Starting m-file code:

```
% lab 2 starting file
i=[-0.14 -0.12 -0.10 -0.08 -0.06 -0.03 0 0.03 0.06 0.08 0.10 0.12 0.14]; % YOU SHOULD CHANGE THIS
                                                                          % TO THE AVERAGE VALUES
rpm=[????]  %vector of velocity data

kt = ????;
T=kt*i;          % convert current data to torque data
w=(????)*rpm      % convert rpm to rad/s

west=(2*pi/60)*[-3500 3500];
best = ????;     % play with this to get the straight line to approximate the data
Test=best*west;

plot(w,T,west,Test);
ylabel('Friction Torque (????)');
xlabel('Angular Velocity (rad/s)');
title('Input Torque vs. Angular Velocity with Estimated Straight Line Fit');

% icresponse part of the file requires "data" to be present in the workspace
% uncomment the lines below to complete the ic response plot

% dataTime=data(:,1);  %extract the first column of the data matrix
% dataRPM=data(:,5);
%
% to=????;  %time at which torque was shut off in original data
% dataTime=dataTime-to;  %shift the time vector of the data to zero at IC
%
% J=1.29e-5;
% tau=J/best;  %using your units for J and best is tau in seconds? check it
% Wo=????;  %Initial velocity (rpm)
%
% theoryTime=0:0.01:3;   %WHAT DOES THIS DO? TRY IT IN THE COMMAND WINDOW. ALSO TYPE HELP COLON
% theoryRPM=Wo*exp(-theoryTime/tau);
%
% figure(2);
% plot(dataTime,????,????,theoryRPM);
% ylabel('Angular Velocty (????)');
% xlabel('Time (sec)');
% title('Actual and Theoretical Response to and IC of ????? rpm');
```

Experiment 2:

In this lab you are to experiment with the position control system of the "Motorlab" apparatus. Also, you are to use a model of the closed-loop position control system to predict the response. You will compare the theoretical step response with the actual response obtained experimentally from the Motorlab. You will compare the responses for three different proportional controller gains. You should also make connections between pole locations and characteristics of the response such as the frequency of oscillation and the decay rate of the oscillations.

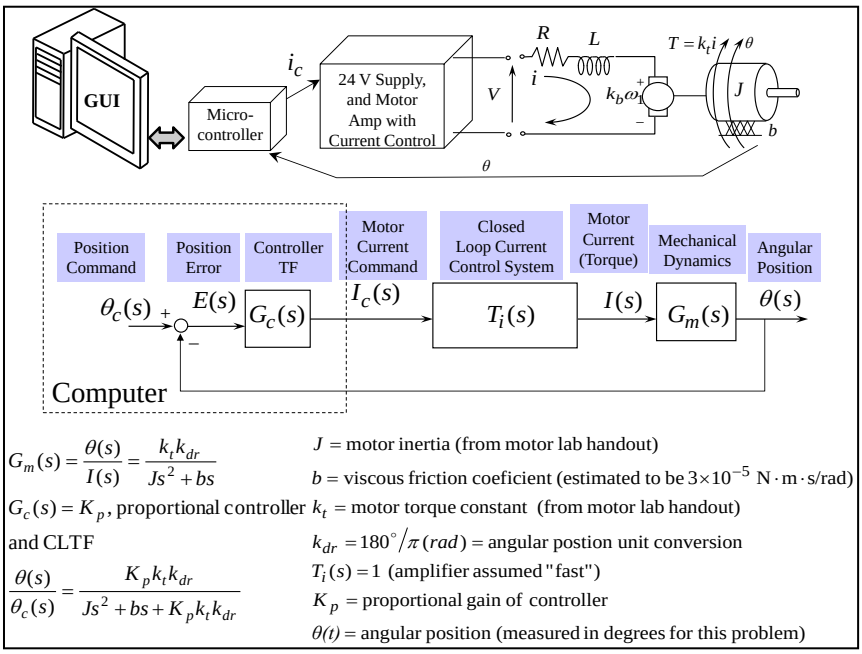


Figure A.2: Lab 2 System

Work To Be Done Prior To Lab:

a) Assuming the transfer function of the closed loop current control system, $T_i(s)$, is one obtain a symbolic representation of the CLTF $\theta(s)/\theta_c(s)$.
b) From a) write an equation for the closed-loop poles of the system.
c) From b) determine an equation for K_p where the response of the system becomes oscillatory (i.e. where the poles become complex rather than real).
d) Plug in the numbers and determine the value of K_p for part c).
e) Plug the numbers and the following three gains into your equation for part b) to find the oscillation frequency, and time constant for the decay rate of the oscillations, for each gain. $K_p = 0.01, 0.001, 0.0001$ $(units?)$

Obtaining Data From The MotorLab:

In this lab you are using a position control system. Therefore, you should run the Motorlab control program in position control mode. For this part of the lab you need to collect experimental data for the step response of the closed-loop system for the three different proportional gains given above. You will have to change the gains and you will have to play with the sample frequency and wave frequency to obtain appropriate data that shows the entire step response. Use the following wave magnitudes for the responses and save the data into the

matrix name given. Hint: you should have at least 3 seconds of data on the positive portion of the square wave.

PAY ATTENTION TO THE ORDER!

Table A.2: Lab 2 Data Collection Instructions

Gain, K_p (units?)	Magnitude of Square Wave (degrees)	Name MATLAB workspace matrix for data.
0.01	200	data3
0.001	1000	data2
0.0001	10000	data1

Immediately after importing the data to the workspace, plot the data using the "mlposplots(data*i*)" command inside of the MATLAB command window. Check the appropriateness of your sample frequency and wave frequency. Also, use the data cursors to measure the period of oscillation for the table below.

Obtaining the required plots and data

By completing the given m-file code you should generate the required plots for this lab. You should also fill in the table below. Some of the data for this table is generated in the m-file. Other data can be found with the data cursors available in the plots generated with "mlposplots.m".

Table A.3: Lab 2 Data Collection

Gain, K_p (units?)	Theoretical CLTF poles, $-\zeta\omega_n \pm j\omega_d$ (rad/s)	Theoretical Period of Oscillations, $2\pi/\omega_d$ (seconds)	Measured Period of Oscillations, T (seconds)	Theoretical Time Constant of Envelope, $1/\zeta\omega_n$ (seconds)	Measured Time Constant of Envelope, τ (estimate one for all three gains) (seconds)
0.01					
0.001					
0.0001					

Things to turn in:

- You should have three different plots (with axis labels including units, titles, and legends): 1) simulated unit step response for all three gains, 2) experimental normalized step responses for all three gains, and 3) simulated and experimental response for $K_p = 0.01$.
- The completed table.
- Hand development of parts a) thru e).
- The answers to the fill in the blanks below (bold and underlined).

Fill in the blanks (Turn in by end of lab):

1. In the theoretical model, as the proportional gain is increased beyond the value where the closed loop response becomes oscillatory, the damped frequency of oscillation _____ and the time constant for the envelope of the oscillations _____. This captures the behavior of the actual system pretty well, although the envelope does change a little. This might be explained by the nonlinear friction and saturations.

2. As we increase the proportional controller gain beyond 0.001 some aspects of the controller get better while others get much worse. If we try to turn the shaft with our fingers the higher gain system deflects much _____ than the lower gain (try it). This indicates _____ disturbance rejection. However, the damping of oscillations in the step response becomes much _____. This indicates the system is nearly unstable. This is one reason we often add "dynamics" to the controller rather than just the proportional gain which has no integrals or _____.

3. If we keep turning the proportional gain up the system actually becomes _____ (try it). The theoretical model we used doesn't predict this. There are always more dynamics out there at higher frequency that we haven't modeled (we'll look at some in the next lab). For example, by assuming the current controller in the amplifier had a TF of 1, we assumed that it responds _____ fast.

4. Using mlposplots to plot the data in the "data1" matrix we see in the fourth plot, which compares the _____ with the _____ command, that early in the response the current does not actually track the commanded current. As we simulated in the previous lab real systems sometimes have saturations that can affect the response. Looking at the other plots we can see in plot number _____ that the _____ seems to saturate during this period, as can be seen by it reaching a high value and staying constant at that value for a short period. We asked our instructor (do this □) and they explained that this is actually due to the limited voltage of the power supply and the _____ constant of the motor. The motor actually generates a voltage as it spins that is proportional to the _____.

Starting m-file code:

```
% lab5.m file
% Requires that the square-wave-response data files
% have been imported into data1, data2, and data3.

kt = ???;  % N-m/A
J=???;     % kg-m^2 or N-m-s^2/rad
b= ???;    % N-m-s/rad
kdr=???;   % deg/rad

Gm=tf(???);

kp=0.0001;
Gol=kp*Gm;
T1=feedback(Gol,1);
[th1,t1]=step(T1);
[p1,z1]=pzmap(T1)

kp=0.001;
Gol=kp*Gm;
T2=feedback(Gol,1);
[th2,t2]=step(T2);
[p2,z2]=pzmap(T2)

kp=0.01;
Gol=kp*Gm;
T3=feedback(Gol,1);
[th3,t3]=step(T3);
[p3,z3]=pzmap(T3)

dt1=data1(:,1);   %extract the time column of the data matrix
dth1=data1(:,3);  %extract the first angle column of the data matrix
dth1=dth1/10000;  %scale the response to a unit step response

dt2=data2(:,1);   %extract the time column of the data matrix
dth2=data2(:,3);  %extract the first angle column of the data matrix
dth2=dth2/2000;   %scale the response to a unit step response

dt3=data3(:,1);   %extract the time column of the data matrix
dth3=data3(:,3);  %extract the first angle column of the data matrix
dth3=dth3/200;    %scale the response to a unit step response

figure(1);              %Theoretical for all three gains
plot(???)

figure(2)               %Experimental for all three gains
plot(???)

figure(3)               %Experimental and Theoretical for Kp=0.01
plot(???)
```

Experiment 3:

In this lab you are to experiment with the velocity control system of the "Motorlab" apparatus. You are to compare proportional control to PI control, understand the concept of "system type", and relate the step responses to the poles of the closed loop transfer functions.

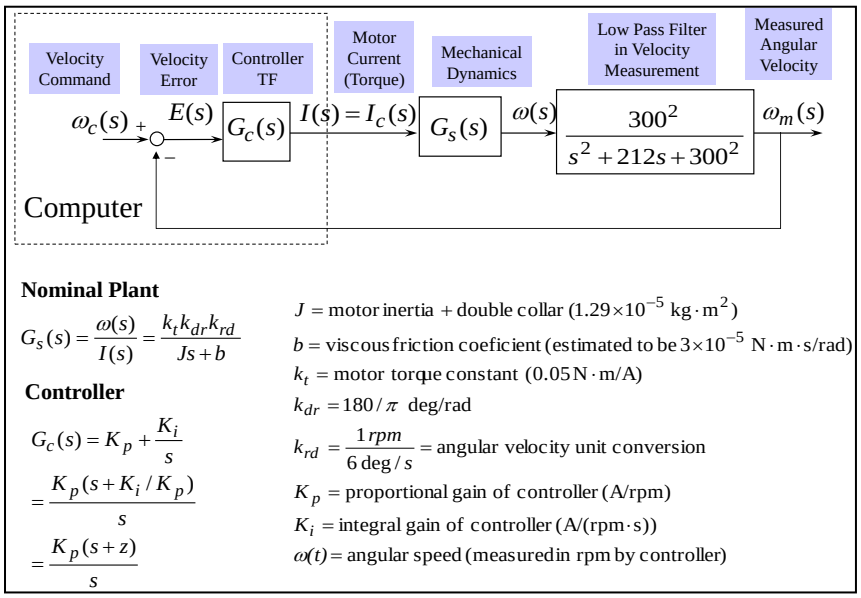


Figure A.3: Lab 3 System

"System Type" Background (See pdf)

"System type" for a unity-feedback closed loop system is defined as the number of free integrators in the open loop transfer function. It can be related to steady state errors for different commands (e.g. steps, ramps, parabolas) to the closed loop system.

Velocity Measurement in The Motorlab

To measure velocity in the Motorlab system a filter is used on the position output from the encoder. This filter takes a derivative, and also uses a second order low pass filter to smooth the discrete pulses coming from the encoder, which would cause larger spikes in the derivative. With a cutoff frequency of 300 rad/s, this filter is the higher frequency dynamics that limit the size of the proportional gain.

Obtaining Data:

You should obtain the step response from the motor lab for three separate controllers: two proportional controllers and one PI controller.

Table A.4: Lab 3 Data Collection Instructions

Gain, K_p (units?)	Gain, K_i (units?)	Magnitude of Step (rpm)	Matrix name when saved into the MATLAB workspace
0.0015	0	1000	data1
0.003	0	1000	data2
0.0015	0.00405	1000	data3

Things to Turn In:

- A single plot showing the experimental step responses obtained for the three sets of controller gains.
- The completed table below
- Include the narrative below with the blanks filled in with bold underlined answers.

Table A.5: Lab 3 Data Collection

Controller Gains		Model				Experimental data
Gain, K_p (*units?*)	Gain, K_i (*units?*)	DC Gain, K_{DC} rpm/rpm)	Step Response SS Speed (rpm)	CLTF poles (rad/s)	CLTF zeros (rad/s)	Step Response SS Speed (rpm)
0.0015	0				none	
0.003	0				none	
0.0015						

Narrative:

In the table and the plots we find that the experimental responses are very, very similar to the response from the models. So we will use the models for detailed discussion.

With a step input of 1000 RPM, the steady state speeds for the two systems with the proportional controllers are ???? RPM and ????? RPM. Using the friction coefficient we can calculate that the input torques required to balance with the friction torque at these two speeds are ????? N-m and ????? N-m, respectively. Using the torque constant we can calculate that these two torques correspond to motor currents of ???? Amps and ???? Amps. Now, we can look at this from another direction. The output of a proportional controller is the gain multiplied by the error. We find that the steady state outputs of the two controllers should be ????(Amps/RPM)*40 (RPM) = ????Amps, and ????(Amps/RPM)*????(RPM) = ???? Amps. Therefore, we see that the outputs of the controllers in steady state are balancing with the ???? torques (currents). And, since a steady state torque (current) is required to maintain speed in this system, there must be a steady state error if we only use ????? control.

When the integral gain is included, we see that the steady state speed is ???? RPM, because the DC gain of the CLTF is ????. The integral part of the controller continues to grow, by integrating the error, until the steady state error ???? ????? ????. With only proportional control we can only decrease the steady state error by ???? the gain, which we see causes the system to become more oscillatory and less ????. However, we can drive the steady state error to zero with the smaller proportional gain when we include the integral control action, and still maintain a relatively stable closed loop system.

The behavior discussed above can be abstracted to other systems and to other inputs to a closed loop control system. We can use “system type” in this abstraction. We see that the DC gain of

the closed loop system is ???? (i.e. the steady state error for a constant input is ????) if the open loop TF has a ???? ????. In this lab the open loop transfer function has ???? free integrators with proportional control, and is therefore type ????. It has ???? free integrator with the PI controller, and is therefore type ????. However, if we put a ramp command into a closed loop system that is type one, it would have a steady state error. A type two system would track a ramp command with zero steady state error. In general we can increase the ability of the closed loop system to track more quickly changing commands by increasing the system type (i.e. by using ???? control).

On a different subject, we can relate the transient part (not steady state) of step responses of the three systems to closed loop poles and zeros. The first system is dominated by the real pole at ???? rad/s, with the underdamped poles causing ???? superimposed on top of the first order response. The real pole gives a time constant of ???? seconds, which can be seen in both the step response of the model and the actual system. The second system has a set of complex poles and a real pole, neither of which are ????. The step response looks like a second order response except the first couple of oscillations are not quite symmetric about the steady state value. The first order pole causes them to shade ????. The third system is very similar to the first except it has very near ????-???? ???? at -2.7 rad/s.

Starting m-file code:

```
% lab8 m-file
% Requires that the square-wave-response data files
% have been imported into data1, data2, and data3.

kt = 0.05;    % N-m/A
J=1.29e-5;    % kg-m^2 or N-m-s^2/rad
b= 3e-5;      % N-m-s/rad
kdr=180/pi;   % deg/rad
krd=1/6;      % rpm/(deg/s)

Gs=tf(kt*kdr*krd,[J b]);            % mechanical dyn' with speed output
wn=300; zeta = 0.707/2;
Gvf=tf(wn^2,[1 2*wn*zeta wn^2]);    % low pass part of the velocity filter
Gp=Gs*Gvf;

Gc=0.0015;
T1=feedback(Gc*Gp,1);
[p1,z1]=pzmap(T1)
SSspeed = dcgain(T1)*1000

Gc=0.003;
T2=feedback(Gc*Gp,1);
[p2,z2]=pzmap(T2)
SSspeed = dcgain(T2)*1000

Gc = tf(0.0015*[1 2.7],[1 0]);
T3=feedback(Gc*Gp,1);
[p3,z3]=pzmap(T3)
SSspeed = dcgain(T3)*1000

input=data1(:,2);
```

```
t1=data1(:,1);
rpm1=data1(:,5);  %extract the first speed column of the data matrix

t2=data2(:,1);
rpm2=data2(:,5);  %extract the first speed column of the data matrix

t3=data3(:,1);
rpm3=data3(:,5);  %extract the first speed column of the data matrix


figure(1)                          %Experimental for all three gains
plot(t1,input,t1,rpm1,'b',t2,rpm2,'g',t3,rpm3,'r')
legend('StepInput','kp=0.0015','kp=0.003','kp=0.0015,ki=0.0041')
xlabel('Time (s)')
ylabel('Speed Response (rpm)')
title('Step Response of Speed Control with Three Sets of Gains')

[vm1,tm1]=step(1000*T1);
figure(2); plot(t1,rpm1,tm1,vm1)
title('Step responses from actual system and model for kp=0.0015')

[vm2,tm2]=step(1000*T2);
figure(3); plot(t2,rpm2,tm2,vm2)
title('Step responses from actual system and model for kp=0.003')

[vm3,tm3]=step(1000*T3);
figure(4); plot(t3,rpm3,tm3,vm3)
title('Step responses from actual system and model for PI control')
```

Experiment 4:

In this lab you are to experimentally determine five data points for the frequency response of the motor-and-spring configuration of the Motorlab. Then you are to estimate all parameters of the model except J and k_{dr}. We assume we know these two parameters accurately.

You are to use sine waves ("Run Wave Autosave ") for the input current on the Motorlab, since the input to the TF of interest is current. You should use a magnitude of 0.25 Amp for sine wave frequencies near the natural frequency (~resonance). This will hopefully prevent fatigue failures of the spring. Be careful near the resonance. It is easy to break the spring with the resonance. For the other input frequencies you are given and input amplitude to use.

To begin the lab, you should experiment to find the actual natural frequency. You should find natural frequency by finding a frequency where the phase lag is very near 90 degrees. If you find a phase lag between 80 and 100 degrees that is sufficient to estimate the natural frequency given that the phase transition is very sharp for this lightly damped system. But try to do your best. Once you have found the natural frequency, then you should fill in the data table. Note that the frequencies you use for data collection are dependent on the natural frequency you find. You may round these other frequencies to the nearest Hz.

$i(t)$ = motor current

$\theta(t)$ = angular position

$k_{dr} = 180/\pi$ deg/rad

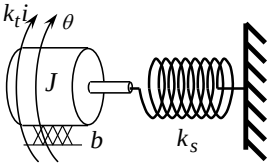
k_s = spring constant

k_t = motor torque constant

J = motor inertia + collar inertia

k_s = spring constant (initial guess $\cong 0.385$ N·m/rad)

b = viscous friction coeficient (initial guess $\cong 2\times10^{-4}$ N·m·s/rad)



$$G(s) = \frac{\theta(s)}{I(s)} = \frac{k_{dr}k_t}{Js^2 + bs + k_s} = \frac{K_{dc}\omega_n^2}{s^2 + 2\varsigma\omega_n s + \omega_n^2}$$

ς = damping ratio, ω_n = natural frequency

ω_d = damped frequency of oscillation = $\omega_n\sqrt{1-\varsigma^2}$

ω_r = resonant frequency = $\omega_n\sqrt{1-2\varsigma^2}$

Figure A.4: Lab 4 System

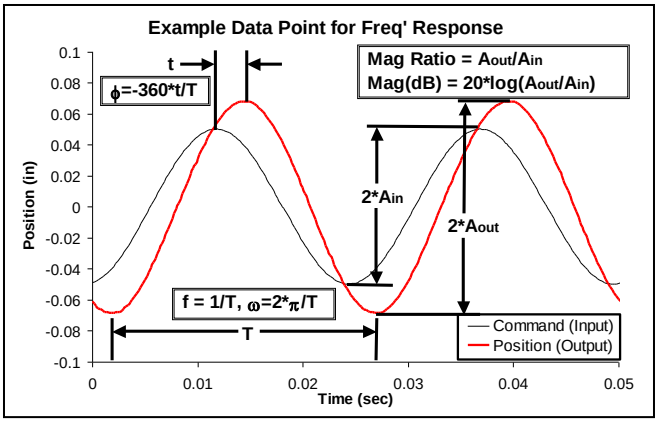


Figure A.5: Calculating Mag. and Phase

Plotting The Responses to Input Sine Waves

You should use the mlolplots(data,Iscale) function. You may have to include the Iscale argument for the current to be visible on the same plot as the position.

Some Related MATLAB Functions

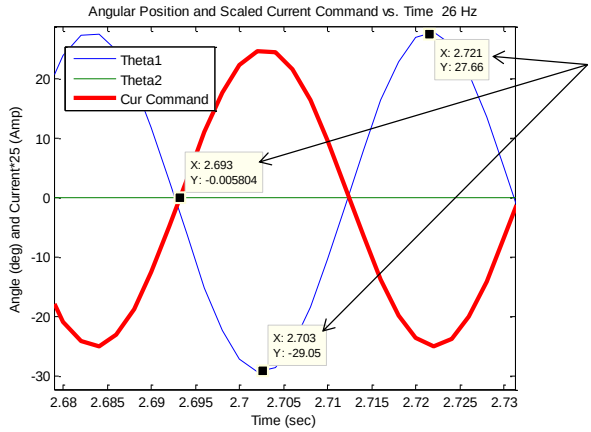
Helpful MATLAB functions:

log10() – log base 10

bode() – generates the bode (freq' response) plot of a tf – note you can change the freq' units to Hz by right clicking on the figure and choosing 'properties'

[m,p,w]=bode() – generates the data for a freq' response plot of a tf – note the mag (m) is a ratio not dB
loglog() – plotting routine for a log-log scale
semilogx() – plotting routine for a log scale on the x-axis

Taking data for the table and searching for the natural frequency.



Use `mlolplots()` to plot the data. Make three data cursors: one for a zero crossing of the current command, and two the peak and valley of the corresponding crossing of the output. Right click on one of the cursors and choose "Export Cursor Data to Workspace." Then run, for example, `calc_mag_phase(cursors,26,2)`. Here the cursor data was saved to a variable named ***cursors***, the input frequency was 26 Hz, and the input amplitude was 2 Amp. Also, in this example the phase lag was about 180 degrees.

Figure A.6: Using calc_mag_phase function

The sample frequency should be chosen so it small enough that there is sufficient time for the response to settle in to steady state oscillations but large enough so that it is at least 10 times larger than the input sine wave frequency. The data should be taken from the steady state oscillations.

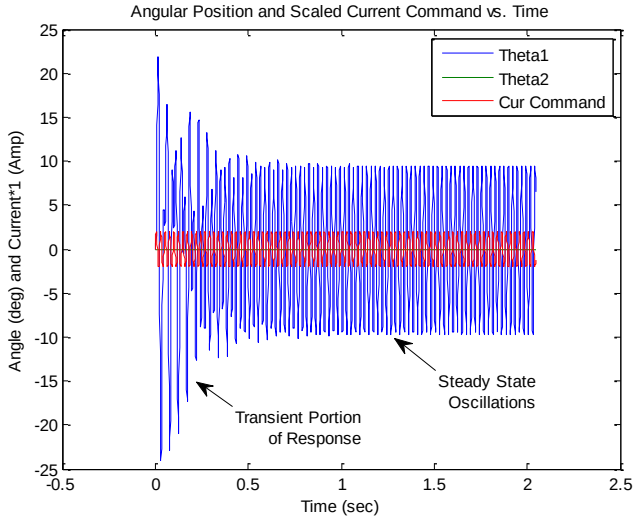


Figure A.7: Transient and Steady State Data

Table A.6: Lab 4 Data Collection

Freq' (Hz)	ω_n	$\omega_n/10$	$0.75 \cdot \omega_n$	$1.25 \cdot \omega_n$	$2 \cdot \omega_n$
Input Amplitude (Amp)	0.25	1	1	1	2
Freq' Value (Hz)					
Mag' Ratio (deg/Amp)					
Mag' Ratio (dB)					
Phase Shift (deg)					

Improve your theoretical model

Use data from the table to find all the coefficients for the standard 2nd order form. The magnitude ratio at one tenth of the natural frequency should give you the DC gain. The damping ratio can be found by symbolically calculating the magnitude of the standard 2nd order form at the natural frequency and using then using the actual magnitude at the natural frequency from the data.

Then you should equate the two forms of the model to determine the physical parameters (k_t, k_s, b) of the model.

Things to Turn In

- The Data Table and new estimates for . (a copy turned in to your instructor before you leave).
- Five experimental plots (from mlolplots() like on the previous page) of the input and output showing the data cursors used for the "calc_mag_phase()" function.
- A final Bode plot showing the initial model, the improved model, and the magnitude and phase data.
- One set of hand-written calculations that duplicate the work done "calc_mag_phase()." This should be for the natural frequency and use the data shown in the data cursors.
- A hand development of the magnitude of the standard 2nd order form at the natural frequency.
- Your completed lab 10 m-file.

```
% lab10.m file

% initial model
kt = 0.05;              % N-m/A
kdr = ???;              % deg/rad
J=???;                  % kg-m^2 or N-m-s^2/rad
b=???;                  % N-m-s/rad
ks = ???;               % N-m/rad

G=tf(????);                % model from initial estimates
figure(1); bode(G)         % generate initial estimate of bode plot
[m,p,w] = bode(G);         % get magnitude, phase, and freq data
m=squeeze(m);              % make m two dimensional
m=20*log10(m);             % convert to dB
p=squeeze(p);

fdata=[2 16 21.15 26 42];
wdata=fdata*2*pi;
magdata=[21.9 28.8 48.6 29.1 13.6];
phdata=[-5 -8 -90 -176 -187];

figure(2);
subplot(2,1,1); semilogx(w,m,wdata,magdata,'*')
title('Bode Plot'); ylabel('magnitude (dB)'); xlabel('freq (rad/s)')
subplot(2,1,2); semilogx(w,p,wdata,phdata,'*')
ylabel('phase (deg)'); xlabel('freq (rad/s)')

%%%%%%%%  system id - come up with improved model
wn = 21.15*2*pi;    % natural freq from data (rad/s)
Kdc = 12.5;         % dc gain from data
Mwn = 267;          % magnitude ratio at wn from data
zeta = Kdc/Mwn/2;   % calculate damping ratio using Mwn and Kdc
Gnew = ????;

[mnew,pnew,wnew] = bode(Gnew); % get magnitude, phase, and freq data
mnew=squeeze(mnew);
mnew=20*log10(mnew);
pnew=squeeze(pnew);

%plot two models and data together
figure(3);
subplot(2,1,1);
?????

ksnew = ???      % N-m/rad
bnew = ???       % N-m-s/rad
ktnew = ???      % N-m/A
```

```
% calc_mag_phase function file - should not need modification

function calc_mag_phase(cursors,freq,inputmag)

  ymin = 1e200; ymax = -1e200; crossing=0;
  for i=1:3
      xval = cursors(1,i).Position(1);
      yval = cursors(1,i).Position(2);
      if ((yval > -0.5)&&(yval < 0.5))
          tcrossing = xval;
          crossing = i;
      end
      if (yval > ymax)
          peak = i;
          ymax = yval;
          tpeak = xval;
      end
      if (yval < ymin)
          valley = i;
          ymin = yval;
          tvalley = xval;
      end
  end
  dtPeakValley = abs(tpeak-tvalley);
  outfreq=1/2/dtPeakValley;
  if ((crossing==0)||(valley==peak))
    display('Innacurate cursors or freq!')
    return
  end
  if ((abs(outfreq-freq)/freq > 0.05)||(crossing==valley)||(crossing==peak))
      display('Innacurate cursors or freq!')
      return
  end

  timelag=mean([tpeak tvalley])-tcrossing;
  phaselag = 360*timelag*freq
  magratio=(ymax-ymin)/2/inputmag
  dB=20*log10(magratio)

end
```

Appendix B - MotorLab Specifications

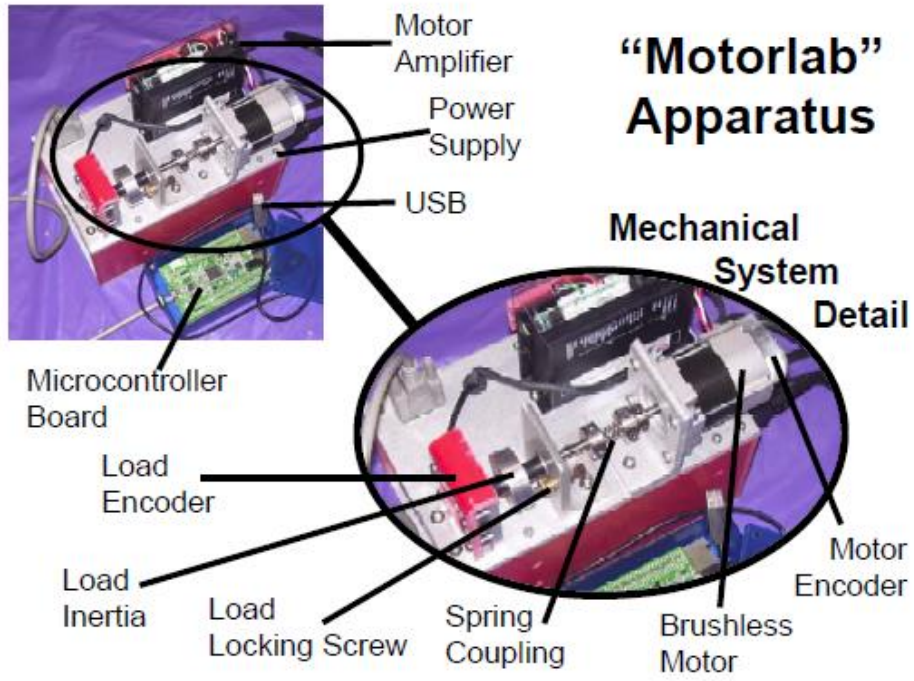


Figure B.1: MotorLab Components

System Description

Below is a schematic representation of the Motorlab system in a closed-loop position or speed control configuration. There are two position sensors on the apparatus, a motor encoder and a load encoder. The speeds of the two inertias are measured by numerically differentiating the position signals in the computer controlling the system (microcontroller). The motor amplifier has a control loop that measures and controls the electric current in the motor windings. This results in what is commonly known as a "torque controlled" motor, since the magnetic torque is proportional to the current in the windings. The microcontroller is interfaced to the motor amplifier through a +/-10V analog signal. By varying the magnitude of this voltage the microcontroller can change the current in the motor. This voltage, which is proportional to the controlled current, serves as a current command (desired current) for the current control loop in the amplifier. An additional sensor, not shown below, is the current sensor in the amplifier used to implement the current control. The signal from this sensor is also read by the microcontroller, using an analog to digital converter. Although this signal is not used in the control loops on the microcontroller, it is recorded for data analysis.

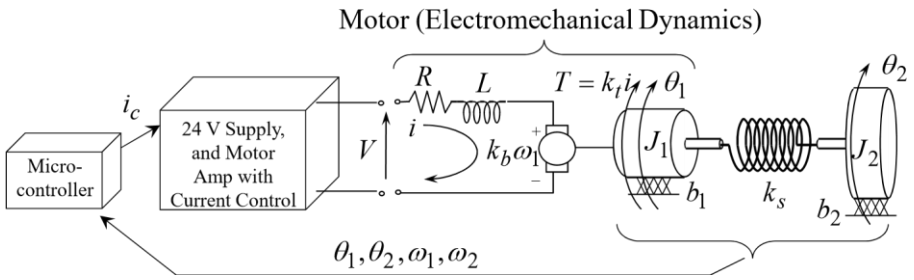


Figure B.2: MotorLab Model

Several different configurations of the system can be utilized in experiments. Either sensor, the motor or load encoder, can be used for the feedback of the control loop. The selection is made in the software interface. The motor encoder is known as a "collocated" sensor since it is co-located with the input to the mechanical system, the motor torque. The load sensor is separated from the input to the system by a spring and is therefore known as a "non-collocated" sensor. In addition to varying which sensor is used, the mechanical system can be changed with the lock down screw and the spring coupling. Also, a choice can be made between velocity control or position control by selecting the appropriate control program. Any of the following mechanical models may be realized using the Motorlab hardware and software.

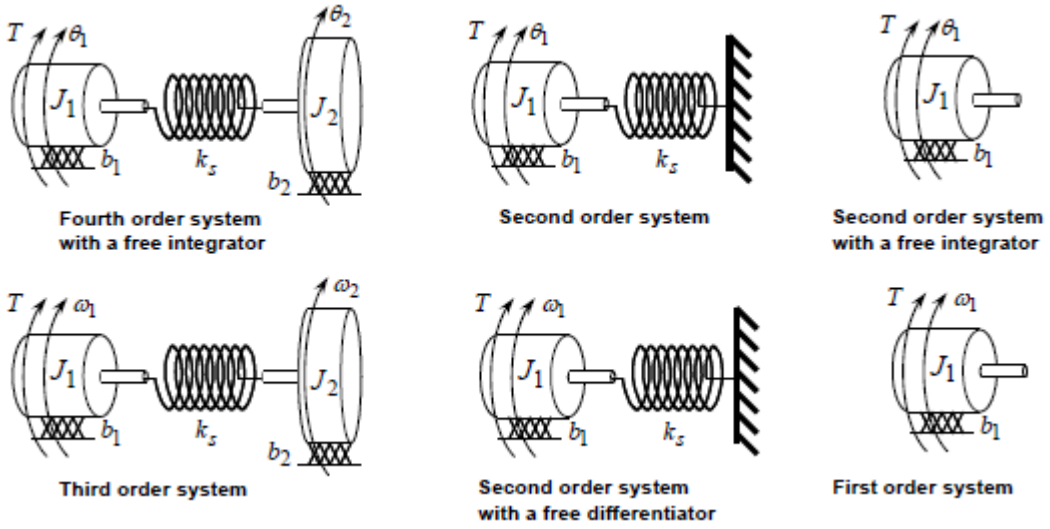


Figure B.3: MotorLab System Configurations

Software

The software for the system can be found in the "c:\Motorlab" directory on the laboratory machines. All the needed Matlab functions can be found there. The software that is on the microcontroller is included in this directory in the motorlabRepo.zip file. This program is burned into the flash memory of the microcontroller and runs on power up. The software that runs on the PC is a GUI written in Matlab ("motorlabGUI.m"). There are additional m-files in the "Motorlab" directory that can be used to plot data from the system.

User Interface

To run the Motorlab GUI you must open Matlab and add the "c:\Motorlab" directory to the Matlab path or set this directory as the current directory. Normally you will add it to the path and set the current directory to the location where you are storing your files. The microcontroller should be plugged into USB. In the Matlab command window type "motorlabGUI." The opening dialog (below) asks you to select the communication port for the microcontroller. If more than one port is listed you should be able to detect which is the Motorlab by unplugging the USB or powering it down and then clicking the "Refresh List" button. The GUI should open after selecting the com port.

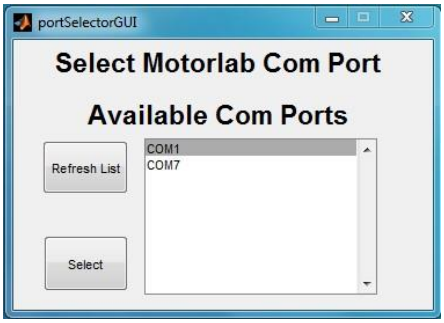


Figure B.4: Connection Dialog

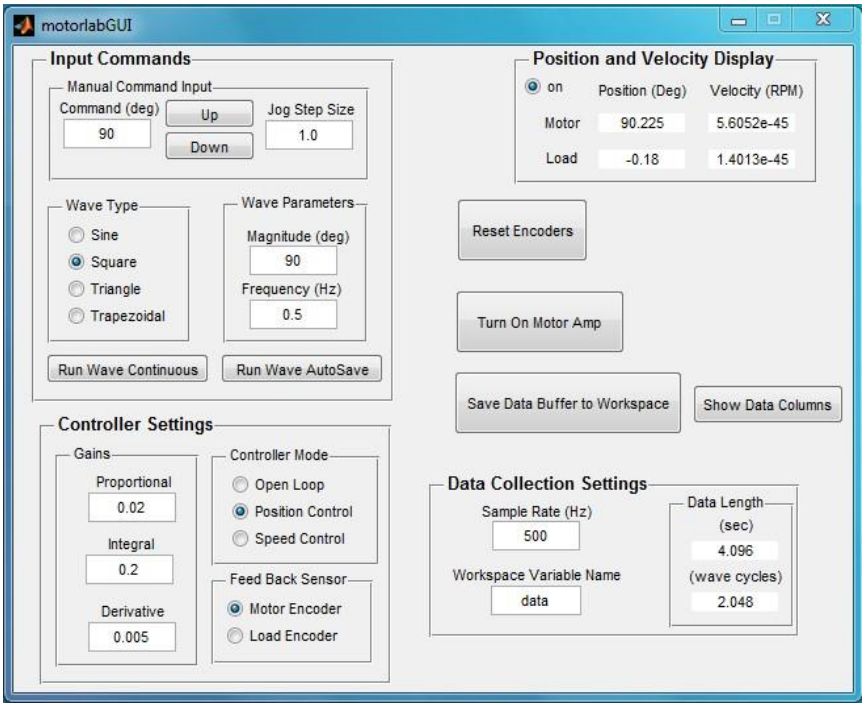


Figure B.5: MotorLab GUI

Data Acquisition

The microcontroller stores data in a circular buffer that is 2048 data samples in length with 9 variables in each sample. After 2048 sample periods the buffer begins to be overwritten with the more recent data. At any time the buffer contains the most recent 2048 samples. Pressing the "Save Data Buffer to Workspace" button will write this data to a 2048x9 matrix in the Matlab workspace. Pressing the "Run Wave AutoSave" button starts the wave type selected and then writes the data to the Matlab workspace once the buffer has filled with new data. The time length of the data depends on the sample rate. If for example the sample rate is set to 500 Hz, then the last 4.096 seconds (2048/500) of data will be saved in the buffer.

The data matrix saved in the Matlab workspace contains 9 variables (columns). The ninth column is reserved. The other eight are listed below. Note that the variable in the second column changes. It depends on the "Controller Mode" chosen at the time of the data storage.

Table B.1: Data Matrix Columns

Column	1	2	3	4	5	6	7	8
Description	Time	Command	Motor Encoder	Load Encoder	Motor Speed	Load Speed	Current Command	Motor Current
Variable	t (sec)	θ_c (deg), ω_c (rpm), i_c (Amp)	θ_1 (deg)	θ_2 (deg)	ω_1 (rpm)	ω_2 (rpm)	i_c (Amp)	i (Amp)

M-files for plotting

There are m-files provided in the "c:\Motorlab" directory that can be used to plot the data from the Motorlab. Although you will frequently want the access the data with your own m-files, these files are useful for quickly viewing the data after acquiring it. There is one file for each of the "Controller Mode" settings.

File: mlolplots.m function: mlolplots(data,Iscale); Uses data generated by the Motorlab in open loop control. If an "Iscale" argument is supplied then the commanded current values are scaled by the Iscale value in the plots. example: mlolplots(data); Does not scale the current command. example: mlolplots(data,Iscale); Multiplies commanded current values by Iscale.

File: mlposplots.m function: mlposplots(data); Uses data generated by the Motorlab position control mode. example: mlposplots(data);

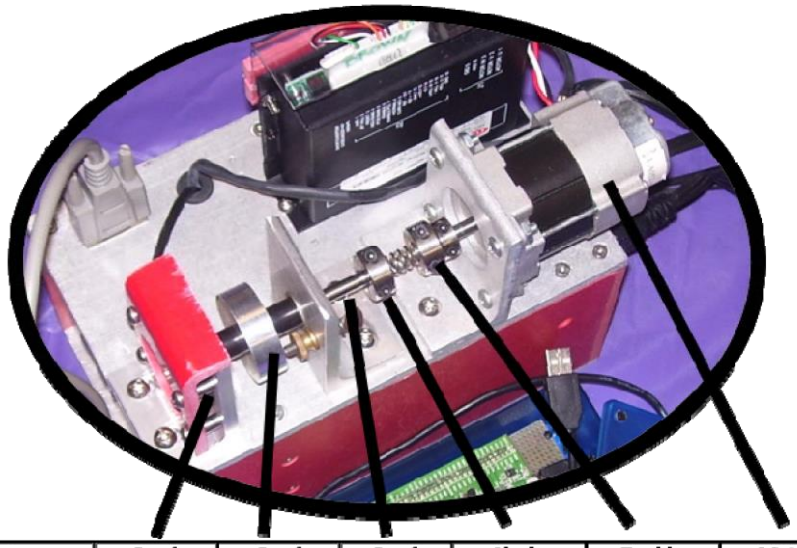
File: mlspeedplots.m function: mlspeedplots(data); Uses data generated by the Motorlab velocity control mode. example: mlvelplots(data);

File: trapprof.m function: [x,v,t] =trapprof(DX,Vmax,Amax,DT) Trapezoidal-velocity motion profile generation Outputs: x=position vector, v=trapezoidal velocity vector, t=time vector
Inputs: DX=distance to move, Vmax=maximum velocity, Amax=maximum acceleration, DT=time step for outputs example: [x,v,t] =trapprof(DX,Vmax,Amax,DT)

Hardware Specifications

Important Scaling Considerations

- Motor Amplifier Scaling = 1 Amp/Volt. Therefore, one Volt output from the microcontroller corresponds to a one Amp command to the current control loop in the motor amplifier. The plotting routines provided take this scaling into consideration.
- Position is measured in degrees and velocity is measured in RPM. The output of the control algorithm in the microcontroller is measured in Volts. Therefore, for example, the units of the proportional and derivate gains in the position controller would be Volts/deg and Volts*sec/deg, respectively. When multiplied by the amplifier scaling (1 Amp/Volt) these gains become Amp/deg and Amp*sec/deg. The units of the proportional gain in the velocity controller would be Volts/RPM (or Amp/RPM if amplifier scaling is included).



Object	Load Encoder	Load Wheel	Load Shaft	Single Shaft Collar	Double Shaft Collar	Motor Rotor and Motor Encoder
Inertia(g-cm^2)	0.8	82	1.4	15	19	110

Figure B.6: MotorLab Inertias

A Few Other Details

- Max Data Acquisition Sample Rate = 10 kHz (the control update rate of the microcontroller software)
- Motor Encoder Resolution = 360 deg/1600 counts = 0.225 deg/count
- Load Encoder Resolution = 360 deg/2000 counts = 0.18 deg/count
- Max motor velocity with the 24 Volt power supply is about 4000 rpm

Speed Measurement

The two speeds measured by the Motorlab system are found using a discrete time approximation (i.e. computer code) of a derivative with a low pass filter. The continuous time transfer function for this filter is given below. It uses the encoder position measurement for input. Note the free *s* in the numerator performs the differentiation and the filter with a cutoff frequency of 300 rad/s helps to filter spikes in the speed measurement caused by differentiating the discrete steps inherent in an encoder position measurement.

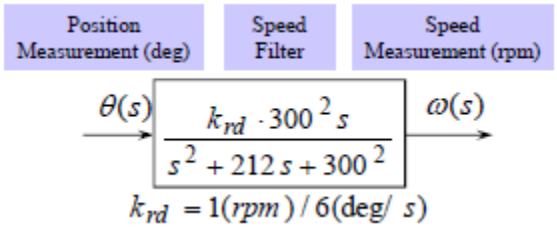


Figure B.7: MotorLab Speed Measurement

Specs from Motor Manufacturer's Data Sheet

Table B.2: Motor Specifications

LA052-040E Motor Dynamic Specs From Shinano Kenshi		
	UNITS	Value
RATED POWER	W	40
RATED VOLTAGE	VDC	24
RATED SPEED	rpm	3,000
RATED TORQUE	N-cm	12.7
	kgf-cm	1.3
RATED CURRENT	A	2.5
TORQUE CONSTANT	N-cm/A	5.0
	kgf-cm/A	0.51
BACK EMF CONSTANT	V/krpm	5.2
PHASE RESISTANCE	Ohm	1.18
PHASE INDUCTANCE	mH	4.4
INSTANTANEOUS PEAK TORQUE	N-cm	38.2
MAX SPEED	rpm	5,000
ROTOR INERTIA	g-cm^2	110
POWER RATE	kW/s	1.48
MECHANICAL TIME CONSTANT	ms	5.2
ELECTRICAL TIME CONSTANT	ms	3.7
MASS	kg	0.6

Current Control Loop Model

The motor amplifier has a current control loop. As configured in the Motorlab apparatus this loop has a bandwidth of approximately 400 Hz. Using data acquired from step and sinusoidal responses the following two closed loop transfer functions have been identified as approximate models for the closed-loop current control dynamics.

$$T_i = \frac{\omega_n^{\,2}(s+z)}{z(s^2+2\varsigma\omega_n s+\omega_n^{\,2})} \quad \text{or} \quad \begin{cases} T_{idelay} = \dfrac{\omega_n^{\,2}(s+z)}{z(s^2+2\varsigma\omega_n s+\omega_n^{\,2})}\, e^{-t_d s} \\ T_{ipade} = \dfrac{\omega_n^{\,2}(s+z)}{z(s^2+2\varsigma\omega_n s+\omega_n^{\,2})} \cdot \dfrac{s^2-6s/t_d+12/t_d^{\,2}}{s^2+6s/t_d+12/t_d^{\,2}} \end{cases}$$

where :

$z = 170 \cdot 2\pi$ *(rad/sec)*

$\omega_n = 230 \cdot 2\pi$ *(rad/sec)*

$\varsigma = 0.8$

$t_d = 0.0002$ *(sec)*

Figure B.8: Current Control Loop Model

Two of the models above contain a time delay while the other does not. One model with the time delay uses the exponential (exact) representation with the delay, while the other uses a second order Pade' approximation of the delay. In the following two figures the responses of these two models are compared with actual data acquired from one of the Motorlab systems. Both the step response and the frequency response models are shown.

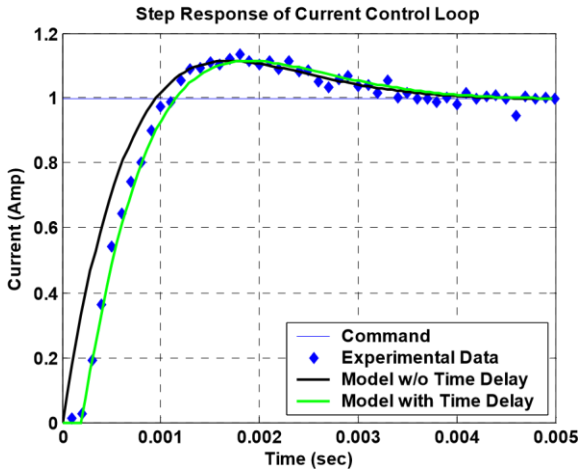


Figure B.9: Step Response of Current Control Loop

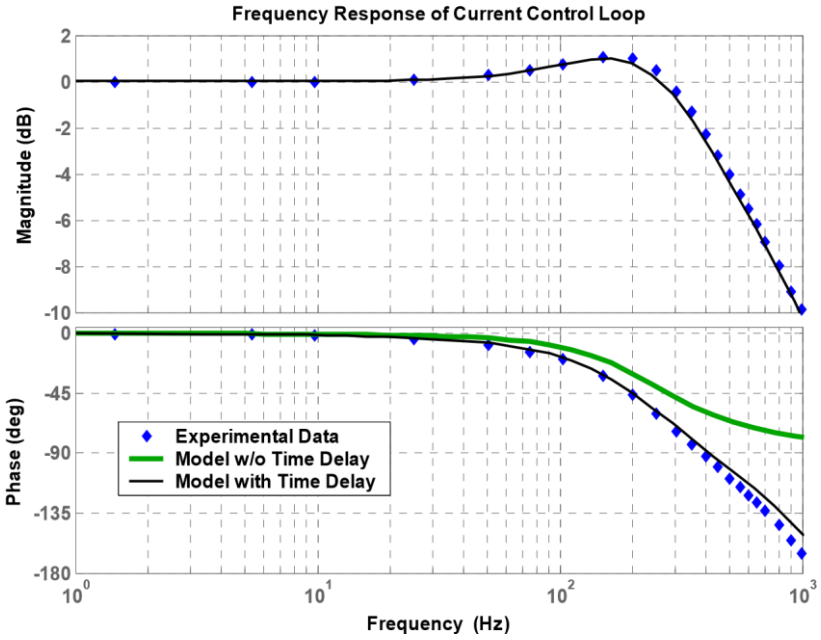


Figure B.10: Frequency Response of Current Control Loop

Appendix C - Eeva Lab Code

Experiment 1:

```
%% Lab 2 -- IC Response and Experimental determination of a system constant
% Fall 2016 - Shane Smith

% System Constants
wheel_radius = 0.03;     % meters
J = 3.23e-8;             % motor inertia (kg*m^2)
R = 6.8;                 % motor resistance (Ohms)
kt = 0.0025;             % motor torque constant (N*m/Amp)
gear_ratio = 29.86;

% Voltages to take speeds at
V = [-3 -2 -1 -0.5 -0.3 0 0.3 0.5 1 2 3];

% Measured Left Speed at corresponding voltage (m/s)
L_speed = [-1.0046 -0.613 -0.2579 0 0 0 0 0 0.2876 0.6394 0.9891];

% Measured Right Speed at corresponding voltage (m/s)
R_speed = [-1.0225 -0.6286 -0.2708 -0.0914 0 0 0 0.0969 0.2777 0.6441
0.9972];

% Convert speeds to rad/sec at the motor
L_rad = (L_speed/wheel_radius)*gear_ratio;
R_rad = (R_speed/wheel_radius)*gear_ratio;

% Estimate best fit line to approximate
west = [-1100 1100];
best = 3.2e-3;      % = (B+kb*kt)/kt, play with this to approximate data
Vest = best*west;

% Plot Velocity vs. Voltage with best fit line
figure(1);
plot(west,Vest,R_rad,V,'*')
xlabel('Motor Speed (rad/sec)'); ylabel('Motor Voltage (V)')
title('Voltage vs. Steady State Speed')
legend('Best Fit Line','Experimental','Location','Southeast')

% Calculate theoretical initial condition response from simple model
R_init = 2.1;   % m/s at wheel
R_init = (R_init/wheel_radius)*gear_ratio;   % rad/s at motor
tau = (R*J)/(best*kt);   % calculating time constant
th_time = 0:0.01:0.75;
R_th_RPM = R_init*exp(-th_time/tau);

% speed filter model
wn = 20*2*pi;
Gfilt = tf(wn^2,[1 1.414*wn wn^2]);
Gm = tf(kt,[R*J best*kt]);
G=Gm*Gfilt;
time = 0:0.0001:0.2;
```

```
u=time*0;
y = lsim(ss(G),u,time,[0 0 2100]/171.3);

% Getting speed measurement without filter
dxdt = (d(2:2000,7)-d(1:1999,7))/.001;  % Wheel position divided by elapsed
time
dxdt(2000)=dxdt(1999);  % making dxdt array same length as data
dxdt = dxdt/wheel_radius*gear_ratio;    % converting to motor speed in rad/s

% Plot experimental data vs theoretical data
to = 0.626;     % sec
data_time = d(:,1); data_time=data_time-to; % Grab data time and adjust it
data_R = d(:,9);   % Grab left and right wheel speeds in m/s
data_R = data_R/wheel_radius*gear_ratio;   % convert to motor speed in rad/s

figure(2);
plot(th_time,R_th_RPM,time,y,data_time,data_R)
xlabel('Time (sec)'); ylabel('Motor Speed (rad/s)')
title('Initial Condition Speed Response');
legend('Model without Speed Filter','Model with Speed Filter','Experimental')
axis([-0.01 0.15 -100 2200])

figure(3);
plot(th_time,R_th_RPM,data_time,dxdt)
xlabel('Time (sec)'); ylabel('Motor Speed (rad/s)')
title('Initial Condition Speed Response');
legend('Model without Speed Filter','Experimental \Delta x / \Delta t')
axis([-0.01 0.15 -100 2200])
```

Experiment 2:

```
%% Lab 5 -- Change pole locations, simpler models don't predict stability
% Fall 2016 - Shane Smith

%% System Constants
wheel_radius = 0.03;    % meters
J =  3.23e-8;    % kg*m^2
R = 6.8;         % Ohm
kt = 0.0025;     % N*m/Amp
b_kb = 3.2e-3;   % (b+kb*kt)/kt

Gm = tf(kt,[R*J b_kb*kt]);
wn = 20*2*pi;
Gfilt = tf(wn^2,[1 1.414*wn wn^2]);

%% Build Theoretical Models
ki = (100/29.86)*0.03; Gc = tf(ki,[1 0]);
Gol = Gm*Gc; T1=feedback(Gol,1);
[th1,t1]=step(100*T1,0.7); [p1,z1] = pzmap(T1);

ki = (100/29.86)*0.03; Gc = tf(ki,[1 0]);
Gol = Gm*Gc; T1filt=feedback(Gol*Gfilt,1);
[th1filt,t1filt]=step(100*T1filt,0.7); [p1filt,z1filt] = pzmap(T1filt);

ki = (200/29.86)*0.03; Gc = tf(ki,[1 0]);
Gol2 = Gm*Gc; T2=feedback(Gol2,1);
[th2,t2]=step(100*T2,0.7); [p2,z2] = pzmap(T2);

ki = (200/29.86)*0.03; Gc = tf(ki,[1 0]);
Gol2 = Gm*Gc; T2filt=feedback(Gol2*Gfilt,1);
[th2filt,t2filt]=step(100*T2filt,0.7); [p2filt,z2filt] = pzmap(T2filt);

%% Grab values from experimental data
time = d(:,1); time = time-0.251;
left_speed = d(:,8); % left wheel speed (m/s)
right_speed = d(:,9);% right wheel speed (m/s)
left_motor_speed = (left_speed/0.03)*29.86; % convert to motor speed in rad/s
right_motor_speed = (right_speed/0.03)*29.86; % ""

%% Plot Results
figure(1);
plot(t1,th1,t1filt,th1filt,time,left_motor_speed)
legend('Model without Speed Filter','Model with Speed Filter','Experimental')
xlabel('Time (sec)'); ylabel('Motor Speed (rad/s)'); axis([-0.025 0.7 -5
170])
title('Speed Controller Step Response, Ki = 0.1 (V*sec/rad)')

figure(2);
plot(t2,th2,t2filt,th2filt,time,right_motor_speed)
legend('Model without Speed Filter','Model with Speed Filter','Experimental')
xlabel('Time (sec)'); ylabel('Motor Speed (rad/s)'); axis([-0.025 0.7 -5
210])
title('Speed Controller Step Response, Ki = 0.2 (V*sec/rad)')
```

Experiment 3:

```
%% Lab 8 -- PI controller improves tracking, system type
% Fall 2016 - Shane Smith

%% System Constants
wheel_radius = 0.03;   % meters
J = 3.2328e-08; % kg*m^2
R = 6.8;        % Ohm
kt = 0.0025;    % N*m/Amp
kt = 0.0025;    % N*m/Amp
b_kb = 3.2e-3;  % (b+kb*kt)/kt

Gm = tf(kt,[R*J b_kb*kt]);

wn = 20*2*pi;
Gfilt = tf(wn^2,[1 1.414*wn wn^2]);

%% Build Theoretical models
kp1 = (8/29.86)*0.03;
kp2 = (20/29.86)*0.03;
ki1 = (240/29.86)*0.03;
wc = 497.7; % rad/s

% First Proportional Controller w/out speed filter
Gol1 = kp1*Gm; T1 = feedback(Gol1,1);
[w1, t1] = step(wc*T1, 0.5);

% First Proportional Controller with speed filter
Gol1filt = kp1*Gm*Gfilt; T1filt = feedback(Gol1filt,1);
[w1filt, t1filt] = step(wc*T1filt, 0.5);

% Second Proportional Controller w/out speed filter
Gol2 = kp2*Gm; T2 = feedback(Gol2,1);
[w2, t2] = step(wc*T2, 0.5);

% Second Proportional Controller with speed filter
Gol2filt = kp2*Gm*Gfilt; T2filt = feedback(Gol2filt,1);
[w2filt, t2filt] = step(wc*T2filt, 0.5);

% PI controller w/out speed filter
z = ki1/kp1;
Gc = tf(kp1*[1 z],[1 0]); Gol3 = Gc*Gm; T3 = feedback(Gol3,1);
[w3, t3] = step(wc*T3, 0.5);

% PI controller with speed filter
z = ki1/kp1;
Gol3filt = Gc*Gm*Gfilt; T3filt = feedback(Gol3filt,1);
[w3filt, t3filt] = step(wc*T3filt, 0.5);

%% Grab values from experimental data
time = d(:,1); time = time-0.124;
wave = d(:,3);    % commanded value (m/s)
left_speed = d(:,8); % left wheel speed (m/s)
```

```
right_speed = d(:,9);% right wheel speed (m/s)
left_motor_speed = (left_speed/0.03)*29.86; % convert to motor speed in rad/s
right_motor_speed = (right_speed/0.03)*29.86; % ""
wave_rad = (wave/0.03)*29.86;

%% Plot Results
figure(1);
plot(time,wave_rad,':',t2,w2,'--',t2filt,w2filt,time,left_motor_speed)
legend('Command','Model without Speed Filter','Model with Speed
Filter','Experimental')
xlabel('Time (sec)'); ylabel('Motor Speed (rad/s)'); axis([-0.025 0.5 -5
800])
title('Speed Controller Step Response, Kp = 0.02 (V*sec/rad)')

figure(2);
plot(time,wave_rad,':',t3,w3,'--',t3filt,w3filt,time,right_motor_speed)
legend('Command','Model without Speed Filter','Model with Speed
Filter','Experimental')
xlabel('Time (sec)'); ylabel('Motor Speed (rad/s)'); axis([-0.025 0.3 -5
700])
title('Speed Controller Step Response, Kp = 0.008, Ki = 0.24 (V*sec/rad)')
```

Experiment 4:

```
%% Lab 10 -- Finding resonance frequency
% This lab looks at frequency response and resonance behavior

%% Initial Guess at Values
kt = 0.0025;     % N-m/A
kb = 0.0025;     % V-s/rad
R = 6.8;         % Ohms
b = 2.6e-7;      % (board friction) N-m-s
J = 3.3e-8;      % (board inertia, not motor) kg-m^2
len = 0.0315;    % m, length in meters from motor shaft to c.g.
mass = 0.137;    % kg, mass of robot (w/o wheels)
gr = 29.86;      % gear ratio
g = 9.81;        % m/s^2

%% Initial Theoretical Model
Gp = tf([2*kt/gr],[J (2*kt*kb/R + b) mass*g*len]);
figure(1); bode(Gp)
[m,p,w]=bode(Gp);   % get magnitude, phase, and freq data
m = squeeze(m);     % make m two dimensional
m = 20*log10(m);    % Convert to dB
p = squeeze(p);
%% Experimental Model
fdata = [0.18 1.35 1.8 2.25 3.6];
wdata = fdata*2*pi;
magdata = [21.06 28.83 24.65 21.25 10.92];
phdata = [-34.34 -63.87 -87.29 -114.95 -146.31];

figure(2);
subplot(2,1,1); semilogx(wdata,magdata,'*')
title('Bode Plot'); ylabel('magnitude (dB)'); xlabel('freq (rad/s)')
legend('Model','Experimental')
subplot(2,1,2); semilogx(wdata,phdata,'*')
ylabel('phase (deg)'); xlabel('freq (rad/s)')

%% System ID - Come up with improved Model
wn = 1.8*2*pi;
Kdc = 11.3;
Mwn = 17.08;
zeta = Kdc/Mwn/2;

Gnew = tf([Kdc*wn^2],[1 2*zeta*wn wn^2]);
[mnew, pnew, wnew] = bode(Gnew);
mnew = squeeze(mnew);
mnew = 20*log10(mnew);
pnew = squeeze(pnew);

% Plot two models and data together
figure(3);
subplot(2,1,1); semilogx(wnew,mnew,wdata,magdata,'*')
title('Bode Plot'); ylabel('magnitude (dB)'); xlabel('freq (rad/s)')
legend('Improved Model','Experimental')
subplot(2,1,2); semilogx(wnew,pnew,wdata,phdata,'*')
ylabel('phase (deg)'); xlabel('freq (rad/s)')
```

```
Jnew = (mass*g*len)/(wn^2)
ktnew = R*Jnew*Kdc*(wn^2)*0.5
bnew = (2*zeta*wn*Jnew)-(2*kt*kb)/R


% Eeva Resonance Consistency Test
f1data = [0.18 1.35 1.8 2.25 3.6];
w1data = f1data*2*pi;
mag1data = [21.06 28.83 24.65 21.25 10.92];
ph1data = [-34.34 -63.87 -87.29 -114.95 -146.31];

f2data = [0.17 1.275 1.7 2.125 3.4];
w2data = f2data*2*pi;
mag2data = [13.23 38.51 30.14 19.09 4.52];
ph2data = [-39.17 -59.44 -92.57 -127.84 -151.28];

f3data = [0.16 1.2 1.6 2 3.2];
w3data = f3data*2*pi;
mag3data = [8.89 28.94 34.96 22.1 5.63];
ph3data = [-21.6 -54.18 -84.71 -124.32 -151.3];

f4data = [0.165 1.2375 1.65 2.0625 3.3];
w4data = f4data*2*pi;
mag4data = [9.04 21.01 28.71 15.79 5.02];
ph4data = [-29.03 -66.62 -88.57 -112.9 -154.88];

favgdata = [0.16875 1.265625 1.6875 2.109375 3.375];
wavgdata = favgdata*2*pi;
magavgdata = [10.61718 29.0225 27.7225 17.1325 4.6725];
phavgdata = [-31.035 -61.0275 -88.285 -120.003 -150.943];

figure(1);
semilogx(w1data,mag1data,'*',w2data,mag2data,'*',w3data,mag3data,'*',w4data,m
ag4data,'*',wavgdata,magavgdata,'k--')
title('Bode Magnitude Plot'); ylabel('magnitude (dB)'); xlabel('freq
(rad/s)')
legend('1st Experiment','2nd Experiment','3rd Experiment','4th
Experiment','Average Values')

figure(2);
semilogx(w1data,ph1data,'*',w2data,ph2data,'*',w3data,ph3data,'*',w4data,ph4d
ata,'*',wavgdata,phavgdata,'k--')
title('Bode Phase Plot'); ylabel('phase (deg)'); xlabel('freq (rad/s)')
legend('1st Experiment','2nd Experiment','3rd Experiment','4th
Experiment','Average Values')
```