Notebook 2: Pendulum System

Written by:
Riley Kenyon & Gabe Rodriguez
ECEN 5038: Control Systems Lab

Feb. 21, 2020

Contents

1 Problem Statement

The rotary servo base unit offered by Quanser is a fundamental component to subsequent rotary systems used in this lab. The motor is controlled by a DAC and power amplifier to provide the voltage range (-10V, 10V). There are two sensors on the motor module: a tachometer measures angular velocity and an encoder measures absolute angular position. The resolution of the encoder is 4096 pulses per revolution which provides a discernible difference in position of 0.088°. The tachometer is filtered through the power amplifier and is measured to a 1:1 ratio of voltage to angular velocity in radians per second. Both sensor signals are connected to an ADC and are imported into MATLAB/Simulink using the Quarc software offered by Quanser. Figure 1 references the typical setup of the system.

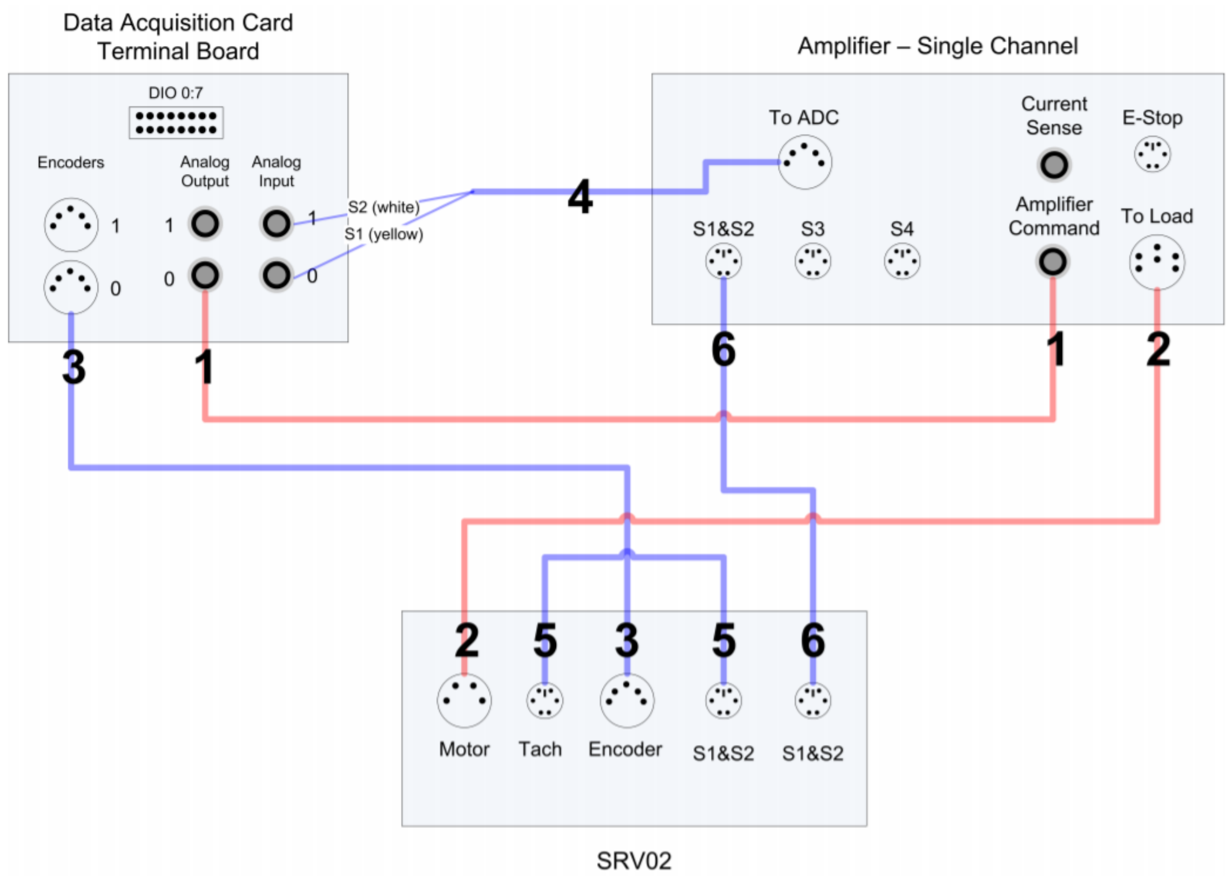


Figure 1: Connecting the SRV02 to a single channel amplifier and two channel DAQ (Image courtesy of the SRV02 User Manual). For the purposes of the lab, the tachometer output is connected directly to the S1&S2 port of the amplifier.

In this notebook, the objective is to identify a pendulum system experimentally and its inherent disturbance as well as to observe the effects of different compensation techniques to achieve minimal steady-state error, rise time, and overshoot. The methods to identify parameters are time- and frequency-based system identification. The notebook ultimately concludes by comparing the various controller designs used to achieve optimal response parameters.

2 Theory

2.1 Theoretical Model

A motor can be modeled as a coiled wire that produces a back voltage from an inertia-resisting motion. In terms of electrical components, this includes a resistance and inductance, modeled by an RL circuit. The mechanical parameters of a pendulum setup include a moment of inertia (J), total pendulum mass (m), centroid length (L), and damping (μ). General relations are found in Eqs. (1) & (2). The coupling that occurs between the components is defined using coefficients k_m and k_τ. The back voltage is defined in Eq. (4) and the torque-current relation is defined by Eq. (3).

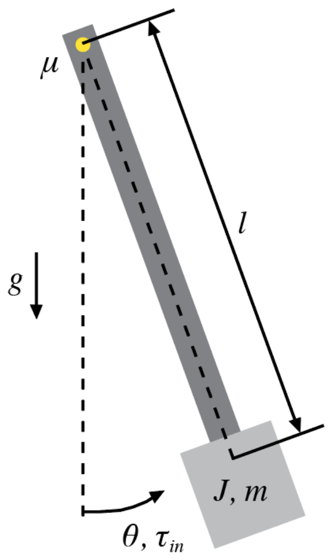


Figure 2: The pendulum setup observed in this lab notebook. The effects of gravity introduce a disturbance to the system that is not present in the previous lab notebook.

$$J\ddot{\theta} = \tau - \mu\dot{\theta} - \frac{1}{2}mglsin\theta \tag{1}$$

$$v_{in} = Ri + L\frac{di}{dt} + v_m \tag{2}$$

$$\tau = k_\tau i \tag{3}$$

$$v_m = k_m\dot{\theta} \tag{4}$$

When combining Eqs. (1) – (4) and assuming negligible motor inductance, the resultant equation is a non-linear differential equation, as shown in Eq. (5).

$$\ddot{\theta} + \frac{\mu R + k_\tau k_m}{JR}\dot{\theta} + \frac{1}{2}\frac{mgl}{J}sin\theta = \frac{k_\tau}{JR}v_{in} \tag{5}$$

For simplicity, the respective coefficients of $\dot{\theta}$, θ, and v_{in} are lumped and labeled as b, a, and c; Eq. (6) is used for any proceeding theoretical equation developments within this lab report.

$$\ddot{\theta} + b\dot{\theta} + asin\theta = cv_{in} \tag{6}$$

Since a nonlinear system cannot be used to generate a transfer function, a Taylor series linear approximation is used to determine initial parameters of the pendulum system, as shown in Eq. (7). In this formula, $\bar{x}$ is the point about which the linear approximation is estimated. To ensure linearity, only the first two terms of the Taylor series are used, as shown in Eq. (8).

$$f(x) \approx f(\bar{x}) + \nabla f(\bar{x})(x - \bar{x}) \tag{7}$$

$$f(x) \approx f(\bar{x}) + f'(\bar{x})(x - \bar{x}) \tag{8}$$

When applying this approximation to $sin\theta$, the resultant linearization of Eq. (6) is obtained, as shown in Eq. (9).

$$\ddot{\theta} + b\dot{\theta} + a(sin\bar{\theta} + cos\bar{\theta}(\theta - \bar{\theta})) = cv_{in} \tag{9}$$

The model's plant input variable (u), which sums the reference input voltage with the constants obtained from the linear representation of $asin\theta$, can then be shown in Eqs. (10).

$$u = v_{in} + \frac{a}{c}(cos\bar{\theta}(\bar{\theta}) - sin\bar{\theta}) \tag{10}$$

$$\ddot{\theta} + b\dot{\theta} + acos\bar{\theta}(\theta) = cu \tag{11}$$

Assuming initial conditions are set to zero, both Eqs. (9) & (11) simplifies to Eq. (12); doing so would produce less than 16% error for $|\theta| < 1$ *rad*. For theoretical simulations, this is considered acceptable.

$$\ddot{\theta} + b\dot{\theta} + a\theta = cv_{in} = cu \tag{12}$$

2.1.1 State Space

Another method of system representation is through state space. The state variable x is defined as angular position (θ) and angular velocity ($\dot{\theta}$). The following expressions utilize the differential equations describing the motor for representing the position-voltage relation in state space.

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{13}$$

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} \tag{14}$$

Eq. (11) can thus be constructed in state space form, Eq. (17), using the following relations.

$$\dot{x_1} = x_2 \tag{15}$$

$$\dot{x_2} \approx -acos\bar{\theta}x_1 - bx_2 + cu \tag{16}$$

$$\dot{x} \approx \begin{bmatrix} 0 & 1 \\ -acos\bar{\theta} & -b \end{bmatrix} x + \begin{bmatrix} 0 \\ c \end{bmatrix} u \tag{17}$$

2.1.2 Transfer Function

Using the Laplace transform of Eq. (11), the voltage-angle relationship of the system simplifies to an approximate second-order transfer function, shown in Eq. (18). Assuming the linear approximation is about $\theta = 0$ rad, Eq. (19) can be used to approximate the system.

$$G(s) = \frac{\Theta(s)}{U(s)} \approx \frac{c}{s^2 + bs + acos\bar{\theta}} \tag{18}$$

$$\frac{\Theta}{U} = \frac{\Theta}{V_{in}} \approx \frac{c}{s^2 + bs + a} \tag{19}$$

2.2 System Identification Methods

2.2.1 Time Domain

A step response is a time domain method for system identification, specifically for low-order systems. For second-order systems, several parameters can be obtained from the step response: rise time (t_r), settling time (t_s), overshoot (OS), natural frequency (w_n), damping ratio (ζ) and the steady-state value (y_{ss}). For this lab, the three parameters used to evaluate the system response are steady-state value, rise time, and overshoot. The steady-state value served as resulting parameters from the other two parameters. Rise time can be related to the natural frequency of the system such that $t_r \approx 1.8/w_n$, and overshoot relates to the damping ratio such that $OS = exp(-\pi\zeta/\sqrt{1-\zeta^2})$.

Based on the results from the previous Lab Notebook, the frequency domain presents more reliable results than the time domain; the Bode plots within the frequency domain provides a closer look at the experimental magnitudes across different frequency, which facilitates more precise match-ups across the theoretical and experimental data. Thus, the frequency domain is utilized for system identification purposes; however, the time domain is used to compare the rise times, overshoots, and steady-state error with different values of the controller inputs.

2.2.2 Frequency Domain

The alternative system identification method, better suited for complex systems, is through the frequency domain. Assuming the system is a linear time-invariant (LTI) system, the frequency of an input signal is equivalent to the frequency of the output from the system. Additionally, by using the superposition principle associated with linear systems, these frequencies can be analyzed individually and summed together. In essence, this method can be utilized by creating a multi-sine input signal and determining the output frequencies and magnitudes.

Two parameters determine the resolution and maximum frequency that can be identified from an experiment: the sampling period (dt) and experiment time (T). Sampling period determines the Nyquist frequency, or the maximum frequency content that can be identified from a signal. This limitation is due to aliasing; the signal frequency can only be reconstructed if the rate of sampling is twice the maximum frequency of the signal. If sampled less than the Nyquist frequency, the signal appears to have lower frequency content. On the opposite end, the experiment time T determines the frequency resolution that can be resolved during reconstruction.

If the experiment is conducted correctly according to sampling and experiment time, the data collected can be converted to the frequency domain via a Fourier transform. The transfer function of the system can then be determined by taking the ratio of magnitudes for the input and output signals. The system must reach steady-state for this method to work, otherwise the ratio will contain excessive noise due to the transient response of the system. This is accomplished through only recording data for the last half of an experiment. To maintain the same T, the overall length of the multi-sine input must be doubled. After post-processing the ratio of magnitudes, the resultant plot is the Bode magnitude of the system.

2.3 Controller Design

The efforts proceeding the pendulum's system identification involve the use of feedback control, shown in Figure 3, to better meet the desired rise time, overshoot, and steady-state value. To ascertain optimal results, several controller designs (C) are tested to identify various advantages and disadvantages for the system. The controllers considered include the following: PI controller, PD controller with gravity compensation, and PID controller.

Figure 3 illustrates the general feedback loop, applicable to the observed feedback pendulum system. The reference input, V_{in}, is the desired input of the system, which is a default 1 V for most controller experiments of the lab. The system output, $\Theta(s)$, represents to the angle of the pendulum as defined in Figure 2. The disturbance is equal to $d = \frac{a}{c} sin\theta$, which is the result of gravity.

2.3.1 PI Controller

To obtain optimal rise times, settling times, and overshoot for the identified second-order system, the ideal objective is to move the system's dominant pole further into the negative real axis from a Root Locus perspective. Theoretically, this can be achieved using a P controller or a PD controller.

The use of a P controller ($C(s) = K_P$) facilitates an increased natural frequency (w_n), thereby moving the dominant pole further in to the negative real axis. However, the ability for a P controller to move the dominant pole of a second-order pendulum system is limited by the mean of the two poles, after which overshoot grows substantially with increasing K_P. Another known disadvantage of a P controller is that steady-state error can never equal 0 using a finite K_P gain, as shown in Figure 5(a). For these reasons, this controller type is not tested experimentally.

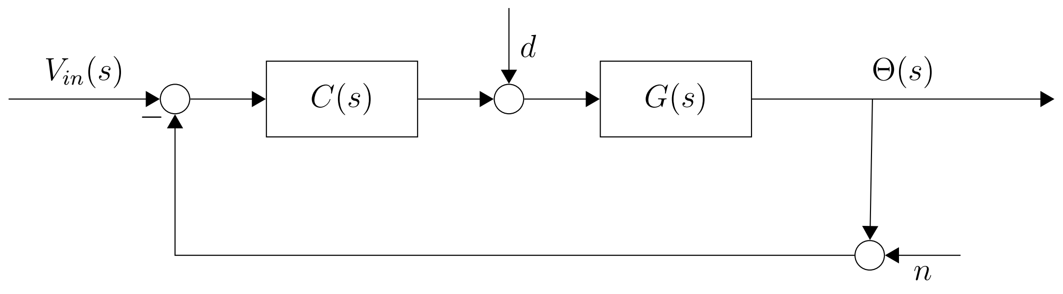


Figure 3: General closed-loop feedback system. In this instance, $R(s)$ represents the input voltage, which is set to 1 V for comparative studies. $C(s)$ is the controller design of the feedback system, explained further in proceeding subsections. $G(s)$ is the pendulum's transfer function with input $U(s)$, and $\Theta(s)$ is the output angle. **Note:** V_{in} represents the reference input voltage, analogous to reference angle, and not the input voltage of the pendulum setup.

A PD controller ($C(s) = K_D s + K_P$) can be implemented theoretically to obtain the desired results from a Root Locus plot. By introducing a zero to the system that is approximately equal to the dominant pole of the pendulum system, the dominant pole essentially cancels out leaving only the non-dominant pole of the pendulum system. This pole then can move further into the negative real axis with increasing gain, as shown in Figure 5(b). While the results of a PD controller are ideal, the physical implementation is non-causal, so it cannot be implemented physically without sensor data. Optimal K_D are additionally difficult to decide, because one cannot determine whether the plant's input voltage reaches saturation voltage.

The subsequent consideration is to design a causal controller that achieve desired effects similar to the PD, by adjusting the PD controller to a lead-lag controller, shown in Eq. (20). The lead controller achieves similar functionality to PD control- a zero is placed close to the plant's dominant pole for cancellation. The lead controller's pole, on the other hand, is inserted to ensure causality and is made large enough to provide maximum movement of the pendulum's pole without reaching the system's saturation voltage. This portion of the controller also increases the phase margin if stability at a certain frequency is needed. The introduction of this pole does, however, limits the maximum displacement of the pendulum's pole. The lag controller is implemented in order to reduce the theoretical steady-state. If the pole of the lag controller is set equal to an integrator—thereby representing a PI controller—the steady error can be minimized. The lag's zero and the controller's gain, K can then be adjusted for fine-tuning the step response. The simulated root locus of a lead-lag controller are shown in Figure 5(c).

$$C(s) = K\left(\frac{s + z_{lead}}{s + p_{lead}}\right)\left(\frac{s + z_{lag}}{s + p_{lag}}\right) \tag{20}$$

The previous three controller objectives and shortcomings validate the use and testing of a PI controller for experimental purposes, shown in Eq. (21), with a form similar to that of the lag controller in Eq. (20). The pure integrator theoretically eliminates the system's

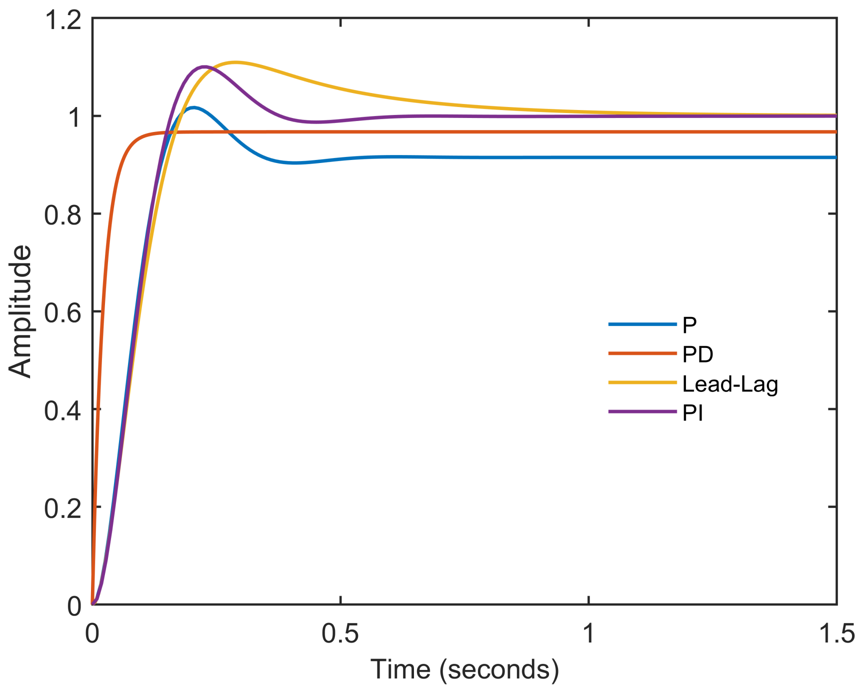


Figure 4: Theoretical step responses for the various controller settings. The controller chosen to be implemented on the physical system is the PI controller due to steady-state value and better settling time compared to the lead-lag.

steady state error for a step response, while the zero serves to cancel out the pendulum dominant pole. When using MATLAB tool, *rltool*, to generate this controller design, the system's step response exhibits quicker rise and settling times compared to the lead-lag controller, as shown in Figure 4. Thus, the PI controller is the first decided controller to test experimentally.

$$C(s) = K_P + \frac{K_I}{s} \tag{21}$$

2.3.2 PD Controller with Gravity Compensation

One of the benefits of PD controller is that it does not have an integrator and does not produce the windup phenomenon if the controller becomes saturated, however it does not have the benefit of producing zero steady-state error to a step input. A gravity compensator is a non-linear component of controller that introduces a theoretical input required to keep the system, in the case of this lab a mass on the end of a beam, at the desired steady state. This process is evaluated for each time step, providing an input equivalent to the energy needed to maintain the state. In the case of the pendulum, this is the sinusoidal term shown in Eq. (6). The controller output, as a differential equation, is shown in Eq. (22). In this instance, the $asin\theta$ serves to cancel out the effects of gravity. The implementation of this adjusted PD controller is feasible in this scenario since the Quanser machine comes with a tachometer and since the system is linearized.

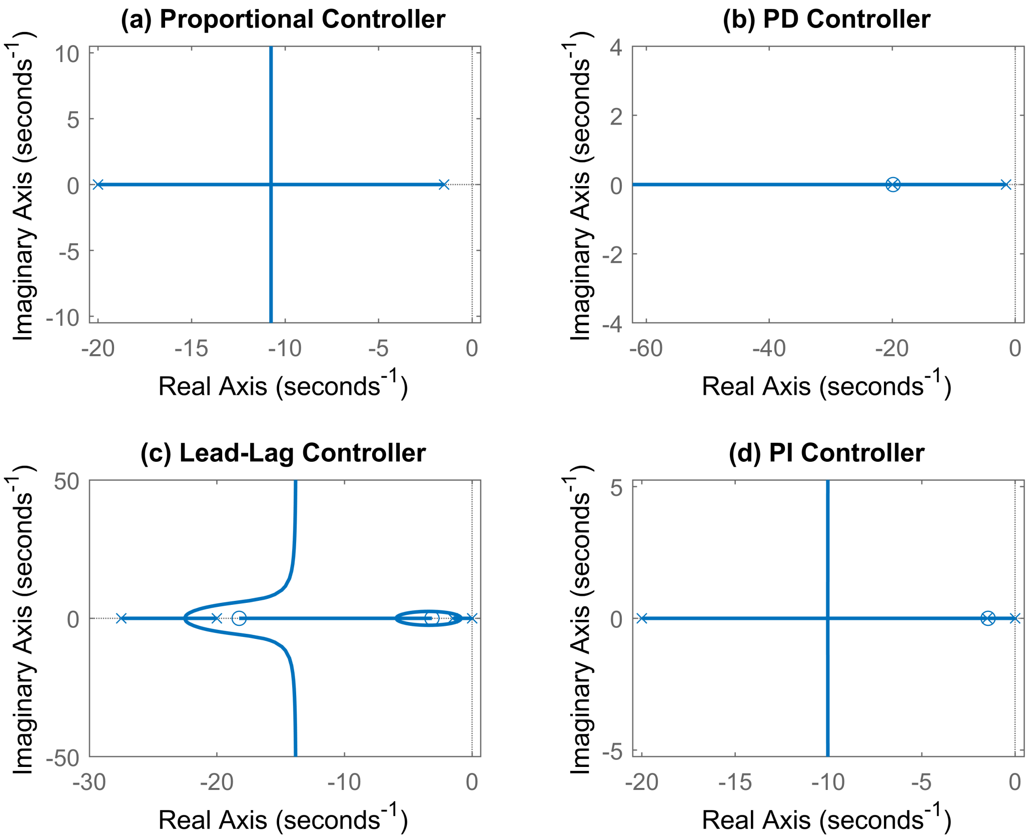


Figure 5: Root locus of various controllers. (a) represents a proportional gain as the controller, (b) is a pure zero, (c) is a lead-lag, which has the benefits of reducing steady state error while decreasing overshoot, (d) is a lag controller with the pole centered on the origin, known as a PI controller.

$$u(t) = \frac{1}{c}(a\sin(\theta) - K_P(\theta - r) - K_D\dot{\theta}) \tag{22}$$

With gravity compensation, the theoretical transfer function of the system under PD control, shown in Eq. (23) can be formed with no need for Taylor series approximation.

$$\frac{\Theta}{V_{in}} = \frac{K_P}{s^2 + (b + K_D)s + K_P} \tag{23}$$

This equation assumes the general second-order transfer function, Eq. (24), to estimate values for K_P and K_D. Equations (25) & (26) shows the relation between the controller constants and inherent parameters defined by the desired rise time and overshoot.

$$\frac{Y}{R} = \frac{w_n^2}{s^2 + 2w_n\zeta s + w_n^2} \tag{24}$$

$$K_P = w_n^2 \tag{25}$$

$$K_D = 2\zeta w_n - b \tag{26}$$

2.3.3 PID Controller

A PID controller is one of the default and widely used control laws. The controller provides the benefits of PD control while introducing an integrator to drive steady-state error to zero. By tuning the controller with the three gains K_P, K_D, K_I, a desired overshoot, rise time, and desired steady-state value can be achieved. The controller has the form of Eq. (27), and has a DC gain of 1 from reference to output in closed loop.

$$C(s) = K_P + K_D s + \frac{K_I}{s} \tag{27}$$

3 Implementation

The frequency method validated in the previous Lab Notebook is used on the physical system to identify a second-order model for the voltage-position relationship. The setup is in open loop for the determining the voltage-velocity relationship. Once the experimental model of the system is determined, the controller described in the proceeding sections is implemented through Simulink.

3.1 System Identification

To determine the characteristics of the new system, a multi-sine input is used to produce frequencies up to approximately 50 rad/s to collect points along the magnitude Bode plot. Using the theoretical model described by the transfer function in Eq. (19), the experimental data collected is used to determine the two pole locations p_1 & p_2. The relation of the poles to the coefficients of the model are shown in the following relations Eq. (28) & (29).

$$p_{1,2} = \frac{-b \pm \sqrt{b^2 - 4a}}{2} \tag{28}$$

$$\begin{aligned} a &= p_1 p_2 \\ b &= p_1 + p_2 \end{aligned} \tag{29}$$

The experimental Bode plot of the pendulum system with the estimated model are shown in Figure 6. The experiment time is run for 40 seconds, while data is collected for the last 20 seconds (T = 20 s) to remove the effect of the transient response on the reconstructed signal. The sampling rate for the Quanser hardware is fixed at $dt = 2$ ms. Based off this information, the frequency range of the multi-sine is at harmonics of the sampling frequency given by $2\pi/T$, up to twice the Nyquist frequency given by π/dt. From the plot, it is estimated that the poles of the system occur at $p_1 = 1.5$ and $p_2 = 20$. An approximation of the DC gain is 1.08. The resultant system is identified to have a transfer function of the form in Eq. (12) with values of $[30, 21.5, 32.4]$ for $[a, b, c]$ respectively.

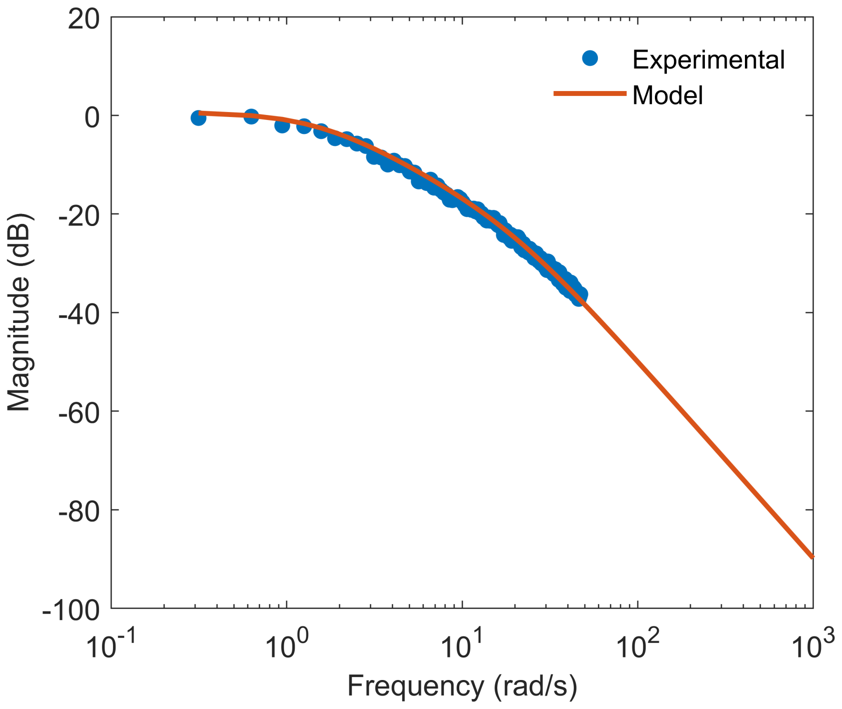


Figure 6: Experimental Bode plot of pendulum system with estimated pole locations at $p_1 = 1.5$ and $p_2 = 20$.

3.1.1 Preliminary Adjustments to Lab Experiment

In the first iterations of system identification, a pendulum without any additional weight is tested. When identifying the lone pendulum model, however, the results are identical to that of the first-order system from the previous Lab Notebook where $K = 1.421$ and $\tau = 0.021$. The moment inertia of the pendulum bar alone is negligibly small ($J \approx 0$), causing Eq. (4) to appear as a first-order system. This initial setup is adjusted by attaching a 100-g weight to the end of the pendulum, thereby obtaining the results explained in the previous subsection.

3.2 Controller Design

With the system identified, it is necessary to define closed loop performance requirements in order to implement a control law. For the purposes of preliminary development, the requirements of rise time and percent overshoot are defined to be no greater than 0.1 sec and 0.5%. These metrics have a direct effect on the closed loop pole locations of the system. For this analysis, the reference input (v_{in}) and the system output are set to 1 V and 1 rad.

3.2.1 PI Control

MATLAB function, *rltool*, is used to decide optimal parameters for K_P and K_I that achieve the desired rise time and overshoot for a step response, resulting in $K_P = 9.05$ and $K_I = 13.07$.

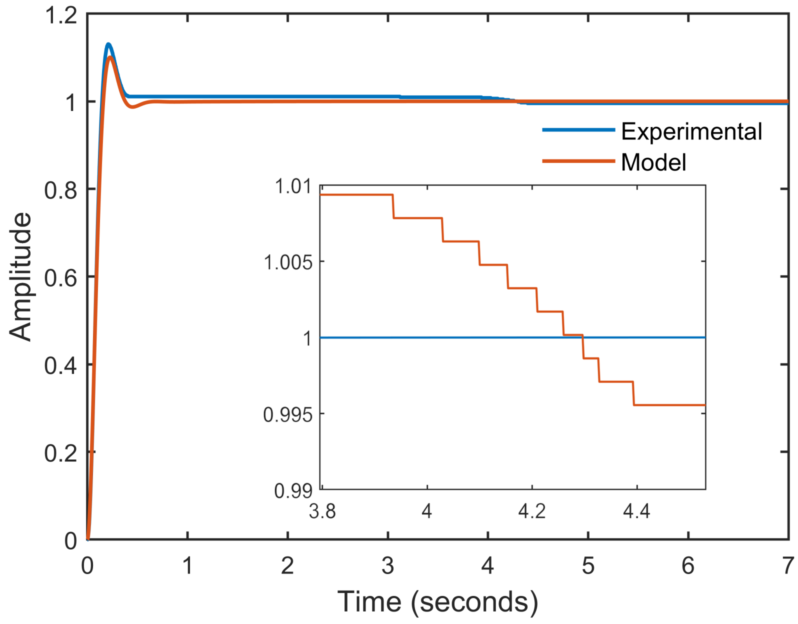


Figure 7: The model and experimental step responses when using a PI controller, where $K_P = 9.0$ and $K_I = 13.07$. The experimental output exhibits a slightly higher overshoot, and its steady state error falls below 1 rad due to the difference between kinetic and static friction.

As shown in Figure 7, the experimental output exhibits a higher overshoot than the model. Between 4 – 5 seconds, the experimental data goes from 1.0094 rad to 0.9956 rad, which is caused by the difference between static and kinetic friction. Since the integrator detects error between the reference input and system output, the controller attempts to reduce the pendulum's angle but must first overcome the motor's static friction force. Once this force is overcome the pendulum moves, but it overshoots its intended angle adjustment since kinetic friction force is naturally smaller than its respective static friction force. This results in a prolonged settling time of the pendulum's angular velocity.

Since the Quarc software available limits the data export to 20 seconds, this controller is considered insufficient for achieving minimal steady-state error. Though the use of an integrator in this controller can guarantee minimization, the ideal settling time should be within the 20-second constraint. For this reason, the other controllers are considered.

3.2.2 PD Control with Gravity Compensation

The desired results of the PD controller are a rise time of 0.1 s and an overshoot of 5%. With these inputs, the resultant K_P and K_D values from Eq. (25) & (26) are 324 and 3.343, respectively. Under these parameters, the experimental results exhibit an 8.15% overshoot and a rise time of 0.0988 seconds. While the overshoot exceeds the desired output, the motor does not reach saturation voltage, deeming this behavior acceptable in this application.

The steady state value of the step response is 1.0017 radians, which is the closest amplitude, greater than 1, that the encoder's resolution can achieve signifying a minimum steady-state error of the system. Additionally, the PD controller does not exhibit the same drop in amplitude seen in the previous controller design. Thus, compared to the PI controller, this PD controller design is a more suitable application for less steady state error and similar rise times and overshoots.

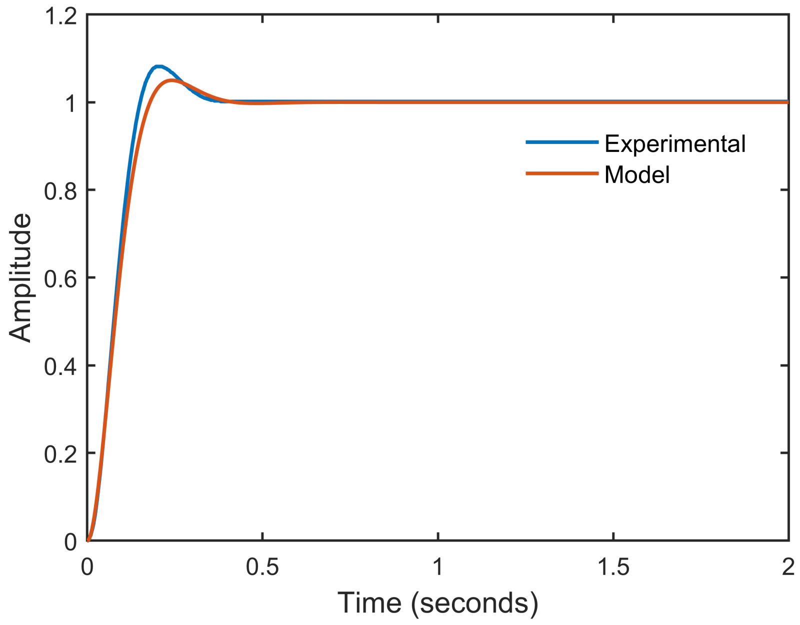


Figure 8: The model and experimental step responses when using a PD controller with gravity compensation, where $K_P = 324$ and $K_D = 3.343$. The experimental output exhibits an overshoot 3.15% greater than what is desired, but its steady state error reaches a minimum according to the encoder resolution.

Though the results of the PD controller with gravity compensation are positive in this experiment, there exists a potential problem in its physical application. That is, the PD controller assumes it reaches steady state when the angular velocity of the motor settles at 0 rad/s, and the change in error (e) of the feedback loop across two or more iterations reaches 0 rad (i.e. the encoder reads the same measurement after multiple iterations). If both conditions are met, the PD controller maintains this angular position regardless of whether the output angle actually equals the desired reference angle. Because the disturbance of the system is known in this experiment, it can be compensated, but there are certain scenarios where the disturbance cannot be predicted and account for in advance, which requires a PID or other advanced controller instead. In preparation for future labs, the PID controller is still tested to compare the results of a system when the disturbance is known and to recognize any other trade-offs between the controller designs.

3.2.3 PID Control

By introducing an integrator, particularly in the controller, the effect of disturbances are mitigated. In the instance of the pendulum, the effect of gravity becomes more significant since the angular position deviates from the equilibrium position. This, in essence, is an additional input to the plant in the form of a disturbance. The majority of the controller remains the same, with the gain coefficients for derivative and proportional gain staying constant. An integral coefficient is determined through experimentation, as shown in figure (9). The best fit for the system is a K_I value equal to 300. In the state space formulation, shown in Eq. (17), the controller output (u) is multiplied by a constant c. The value of K_I is divided by c to yield a gain of 9.26 and is used in the state-space formulation in Simulink. The theoretical model is compared to the experimental system in Figure 10, where the two closely correlate and exemplify the fact that introducing an integrator in the controller yields the desired steady state within the resolution of the system's encoder.

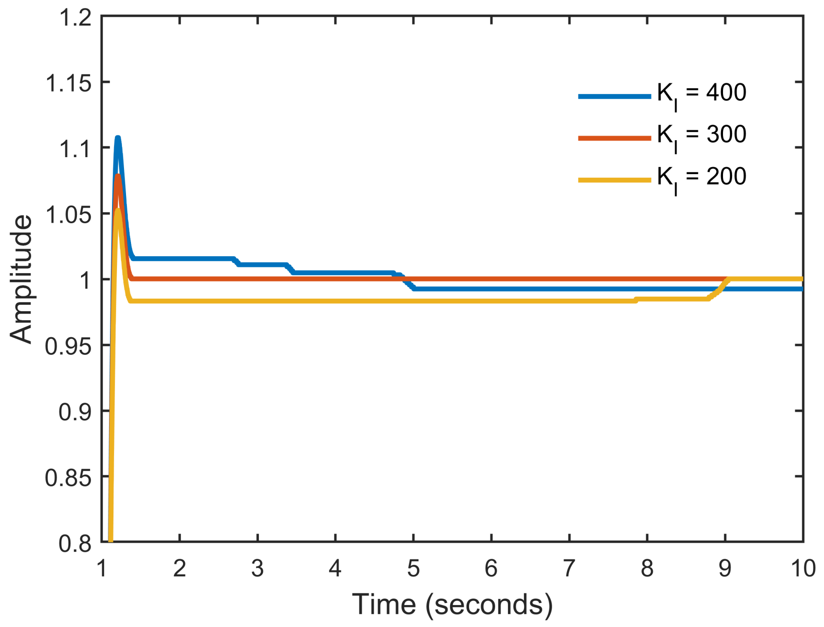


Figure 9: For the PID controller design, three K_I constants are considered: 200, 300, and 400 to reduce the overshoot caused by static friction. Of the three values tested, a K_I of 300 demonstrate the smallest effect from overshoot.

One of the main failure modes of this controller in the context of this system is due to static friction. If the integrator is not tuned to a correct gain, the integrator will accumulate to the point of overcoming the friction. This phenomenon results in oscillation over time from continuous overshooting resulting from the difference between kinetic and static friction. In addition to the problem described, adding an integrator can cause excessive overshoot when initial error is high. Although this is not a problem due to our tests to find an optimal integrator constant, one method to avoid the overshoot is to saturate the integrator at a desired threshold. This also prevents the windup phenomenon that can occur if saturation is not considered when implementing an integrator that continuously compounds the error.

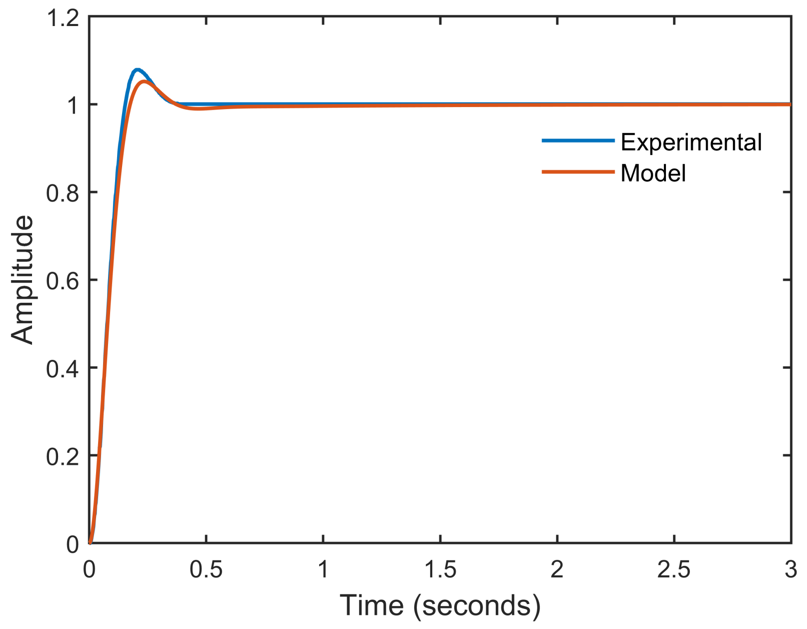


Figure 10: The model and experimental step responses when using a PID controller, where K_P and K_D are the same values as the PD controller with gravity compensation. The experimental output shows an 8.152% overshoot and 0.0995-second rise time, which are identical to the PD controller.

3.3 State Feedback Implementation

In order to implement this experiment in state feedback, the block diagrams are adjusted to compensate for MIMO systems. In the case of the pendulum the states, described in Eq. (13), are the angular position and velocity. In simulation, this is implemented through the use of MATLAB functions to describe the dynamics of the model, and construct the controller output at each time step. The Simulink is shown in the Figure 11 using this functionality. Although the system is not complex in terms of description (i.e. the values for the proportional and derivative gain can be calculated analytically) this is not the case for larger systems. To make the block diagrams more modular for MIMO systems, the MATLAB command *place* is used to determine the position of the closed loop poles by solving for the eigenvalues of the matrix $A-BK$ where the matrix K corresponds to the values of $[K_P, K_D]$. In practice, this is used for the PID controller by assuming the system is compensating for the effects of gravity (to obtain the same values as the PD controller) and then iterating

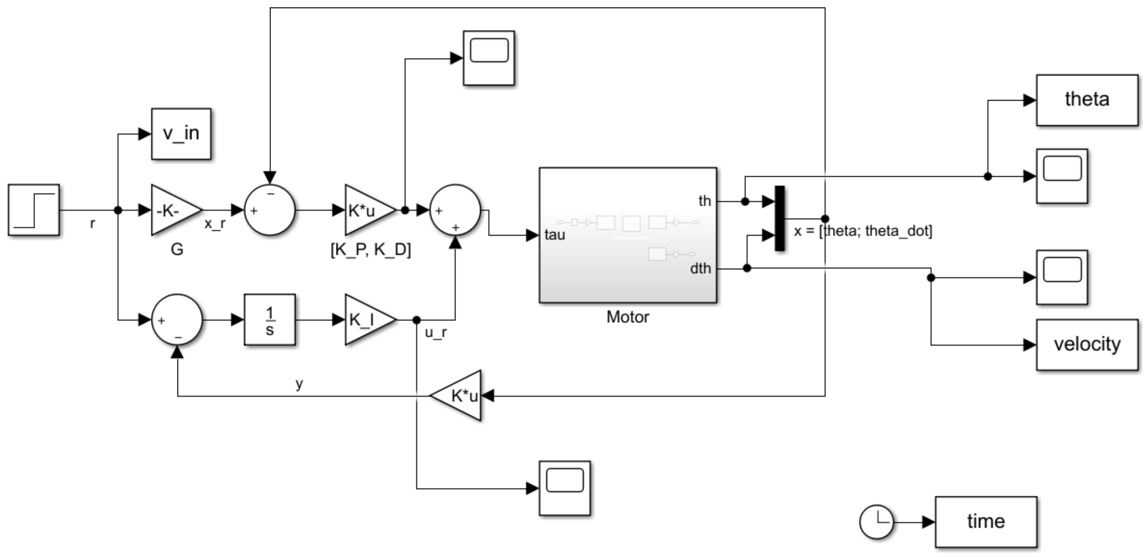


Figure 11: Simulink block utilizing state feedback of the Quarc's sensor measurements. The control law implemented in this case is PID.

the value for the integral gain. The results are effectively the same, but the state feedback approach can be used for more complex systems in the future.

4 Conclusion

In this notebook, the theoretical model of the system is derived based on electro-mechanical equations of motion. It is feasible to model the pendulum setup as a second-order system to relate input voltage to output position. The parameters of the system are identified through a multi-sine input to the system and the resulting magnitudes of the output frequencies. This is done assuming linearity and time invariance such that the superposition principle holds. The experimental Bode plot is used to estimate the pole location of the system. Converting the poles to the general form of the system's transfer function allows for an additional representation in state space to perform state feedback. The preceding methods of identification produced an estimated DC term of $K = 1.08$ with poles located at $p_1 = 1.5$ and $p_2 = 20$. In addition to identifying the model of the system, a large component of the lab is focused on controller design. Of the various controllers used in this lab, the theoretical PI controller responds the quickest with minimal overshoot and has a zero steady-state error, as shown in Figure 4. The theoretical use of a PD controller has the best performance but is not used because the controller is a pure zero and is therefore non-causal.

The PI controller design for the pendulum exhibits the stick-slip phenomenon, where the effect of static friction is significant enough to stop the position of the pendulum from reaching the desired steady state; this effect compounds the error accumulated by the integrator until a torque large enough to overcome the static friction is produced. In this instance, static friction changes to kinetic friction and the pendulum overshoots the desired position. This oscillation continues and the system does not reach the desired reference. Although this performance is not satisfactory, there are methods to reduce the effect, as shown through the PD and PID controllers.

A gravity compensator is used in conjunction with a PD controller to achieve a lower steady-state error. In the case of this system, the effect of gravity on the mass can be modeled as a disturbance. The performance also does not exhibit the stick-slip phenomenon that the PI controller introduced. This controller design proves useful in this pendulum application, however it has potential for steady-state error if the disturbance is unknown.

Another way to reduce the stick-slip phenomenon while accounting for an unknown disturbance is to tune the K_I gain of a PID controller, which also avoids any saturation and reduces overshoot. For this pendulum setup, the gain of the integrator performs best when set to 300. If overshoot is of a concern, the integrator can also be saturated to negate excessive overshoot caused by the initial error in position. A benefit of using an integrator, aside from theoretically matching a reference step, is disturbance rejection. Although the effect of gravity on the pendulum is known in this case, it is not always quantifiable. In order to reject the effects of an unknown disturbance on a system, the controller is required to have an integrator.

5 Appendix II: MATLAB Code

```
%% Initialize
close all;clear all; clc;

%% Desired Parameters

OS  = 0.05;
t_r = 0.10;

p = [1.5, 20];
temp = conv([1 p(1)],[1 p(2)]);

a = temp(3);  %       c
b = temp(2);  % ------------------
c = 1.08 * a;  %  s^2 + bs + a
% c = 0.721 * a;

w_n = 1.8 / t_r;
zeta = sqrt(log(OS)^2 / (pi^2 + log(OS)^2));

K_P = w_n^2;
K_D = 2 * zeta * w_n - b;
K_I = 300;
%% Load data
% load stepdata_01_28.mat          % For sytemID_time
% load fakeStepData.mat
% load freqdata_T5_N150_A30.mat   % For postProcess.m
load ('Old Data/freqdata_closed_T5_N150_A30.mat')

% K = 1;                          % Fake step data
% tau = 0.15;
% tau = 0.026296888692402;        % First round of ID
% K =  1.337053064415507;
tau = 0.0209;        % Second round of ID
% K = 1.340980879186844;
K = 1.4204;
% Define Parameters
dt = 0.002;                       % Sampling time
T = 20;                            % Time of experiment
N = 150;                          % Number of sine waves
A = 15 / N;
i = (1:N)';
w = 2*pi/T*i;                     % Frequency
```

```
% phi = 2*pi/N*i;                      % Chirp
phi = 2*pi*rand(1,N)';                 % Random Noise

% System Model
num = K;
% den = [tau 1];        % velocity
den = [tau 1 0];      % Theta

% Frequencies of fft
w_nyq = pi/dt;
w_res = 2*pi/T;
w_plot = 0:w_res:2*w_nyq-w_res;

% Check input
dt = 0.002;
t_f = 2*T-dt;
t = 0:dt:t_f;

% Create input multisine
u = A*sin(w*t+phi);
u = sum(u);

% Verify bounded between (-10,10) otherwise re-iterate
while all(u <= -10 & u >= 10)
    phi = 2*pi*rand(1,N)';  % Random Noise
    u = A*sin(w*t+phi);
    u = sum(u);
end

% Controller
load('c_bar_2.mat')
C_z = c2d(C,dt,'zoh');
a = C_z.z{1};
b = C_z.p{1};
K_lead = C_z.K;
```

```
%% Initialization

close all

run initialize.m
t_save = t;
offset = 0.0220;          % Resting voltage

%% Fitting Omega
```

```
if strcmp(sim_names{sim_idx}, 'System Identification Omega')

        w_max2 = 20;

        idx = floor(length(velocity) / 2) + 1;
        y = velocity(idx:end) - offset;
        u = v_in(idx:end);
        t = t_save(1:length(y));
        w_plot = w_plot(1:length(y));
        plot_idxs = [4];

        Y = (fft(y)) / length(y) * 2; % Frequencies of output signal
        U = (fft(u)) / length(u) * 2;

        G_mag_exp = Y ./ U;
        G_mag_exp(1) = G_mag_exp(2);
        K2     = mean(abs(G_mag_exp(w_plot < w_max2)));


%         f = @(beta,t) beta(1) * (1 - exp(-t / beta(2)));
%         beta0 = [1;1];
%
%         beta = nlinfit(t,y,f,beta0);
%         K    = beta(1);
%         tau  = beta(2);

        fprintf('The fitted parameters are K = %.4f and tau = %.4f.\n', K, ta

        G_ol_th       = tf(K,[tau 1]);
        [G_ol_mag_th,~] = bode(G_ol_th, w_plot);
        G_ol_mag_th = squeeze(G_ol_mag_th);

%         figure3 = figure;
%         axes1   = axes('Parent',figure3);
%         hold(axes1,'on');
%         plot2 = plot(t,y);
%         plot1 = plot(t,f(beta,t));
%         set(plot1,'DisplayName','Theoretical', 'LineWidth',2);
%         set(plot2,'DisplayName','Experimental','LineWidth',2);
%         ylabel('Amplitude','HorizontalAlignment','center');
%         xlabel('Time (s)');
%         box(axes1,'on');
%         set(axes1,'FontName','Times New Roman','FontSize',font_size,...
%             'XMinorTick','on');
%           legend1 = legend(axes1,'show');
%           set(legend1,'FontSize',font_size,'Location','best');

          print(gcf,filename+'_theo_vs_exp.png','-dpng','-r300');

          figure;
          semilogx(w_plot,db(G_ol_mag_th))
          hold on;
          semilogx(w_plot,db(abs(G_mag_exp)))

          % sim_names{sim_idx} = 'System Identification Theta Open Loop';

end

%% Post Processing Script

switch sim_names{sim_idx}
    case 'Code Validation'
        y = lsim(tf(num,den), u, t);
        plot_idxs = [1, 2, 3];
        is_closed = false;
    case 'System Identification Theta Open Loop'
        t     = (0:dt:T-dt)';
        idx   = length(v_in) - length(t) + 1;
        u     = v_in(idx:end);
        y     = theta(idx:end);
        G_ol  = tf(num,den);
        plot_idxs = [2,4];
        is_closed = false;
    case 'System Identification Theta Closed Loop'
        t     = (0:dt:T-dt)';
        idx   = length(v_in) - length(t) + 1;
        u     = v_in(idx:end);
        y     = theta(idx:end);
        G_ol  = tf(num,den);
        G_cl  = C * G_ol / (1 + C * G_ol);
        plot_idxs = [2,4];
        is_closed = true;
end

% Fourier Transform
U = fft(u); % Frequencies of input signal
Y = fft(y); % Frequencies of output signal

% w_plot = w_plot(w_plot <= w_max);
% U        = U(1:length(w_plot));
% Y        = Y(1:length(w_plot));

% Obtain discrete bode plot of transfer function
G_ol = c2d(G_ol,dt,'zoh');
[G_ol_mag_th, ~] = bode(G_ol, w_plot);
G_ol_mag_th    = squeeze(G_ol_mag_th);

if is_closed
    G_cl = c2d(G_cl,dt,'zoh');
    [G_cl_mag_th, ~] = bode(G_cl, w_plot);
    G_cl_mag_th    = squeeze(G_cl_mag_th);
end

% Calculate experimental transfer function
G_mag_exp = Y./U;                   % Output/Input
G_mag_exp(abs(U) < 1e-3) = 0;    % Remove noise in input signal
G_mag_exp(1) = 0;                   % Remove non-existent DC signal

%% Plots

for plot_idx = plot_idxs
    switch plot_idx

        case 1 % Experimental Time Domain Plot (input vs. output)

            figure1 = figure;
            axes1    = axes('Parent',figure1);
            hold(axes1,'on');
            plot1 = plot(t,u);
            plot2 = plot(t,y);
            set(plot1,'DisplayName','Input', 'LineWidth',2);
            set(plot2,'DisplayName','Output','LineWidth',2);
            ylabel('Amplitude','HorizontalAlignment','center');
            xlabel('Time (s)');
            box(axes1,'on');
            set(axes1,'FontName','Times New Roman','FontSize',font_size,...
                'XMinorTick','on');
            legend1 = legend(axes1,'show');
            set(legend1,'FontSize',font_size,'Location','best');
            axis tight
            print(gcf,filename+'_time.png','-dpng','-r300');

        case 2 % Bode Plot Comparison

```

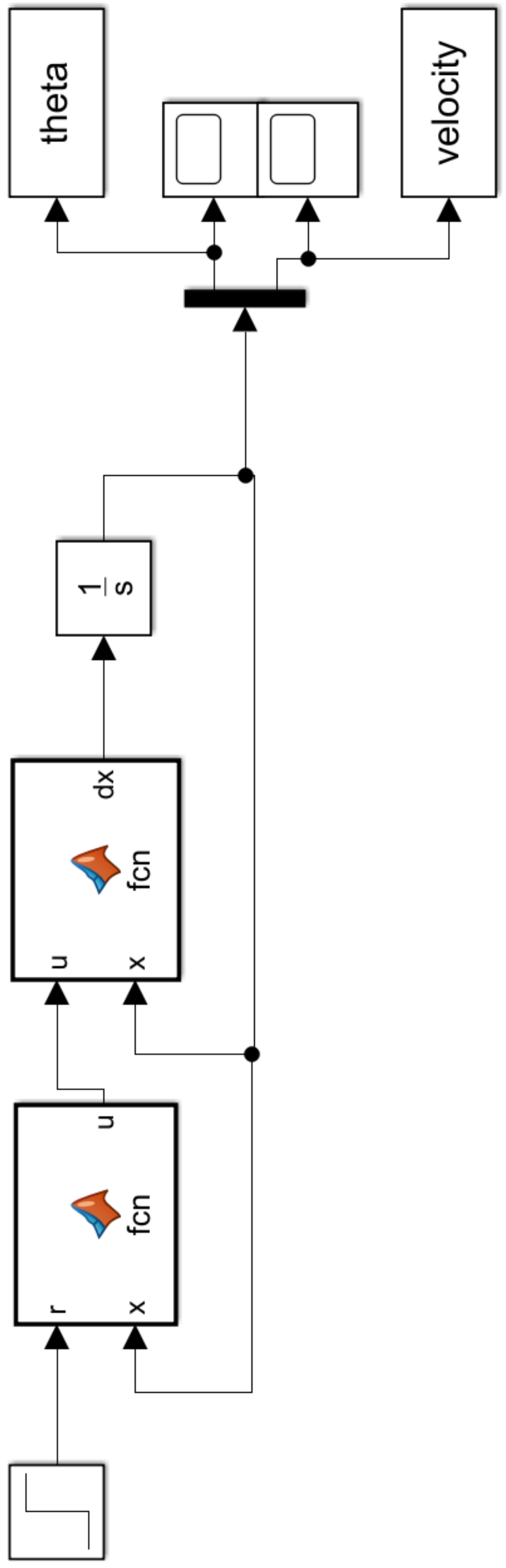
```
            idxs = w_plot < w_max;

            % Overlay theoretical and experimental
            figure2 = figure;
            axes1   = axes('Parent',figure2);
            hold(axes1,'on');
            if is_closed
                semilogx1 = semilogx(w_plot(idxs),[db(abs(G_cl_mag_th(idxs)))
                    db(abs(G_mag_exp(idxs)))]);
            else
                semilogx1 = semilogx(w_plot(idxs),[db(abs(G_ol_mag_th(idxs)))
                    db(abs(G_mag_exp(idxs)))]);
            end
            set(semilogx1(1),'DisplayName','Theoretical','LineWidth',2);
            set(semilogx1(2),'DisplayName','Experimental','Marker','o',...
                'LineStyle','none');
            ylabel('Magnitude (dB)','HorizontalAlignment','center');
            xlabel('Frequency (rad/s)');
            xlim(axes1,[3 500]);
            box(axes1,'on');
            axis tight
            set(axes1,'FontName','Times New Roman','FontSize',font_size,'XMi
                'XScale','log');
            legend2 = legend(axes1,'show');
            set(legend2,'FontSize',font_size,'Location','best');

            print(gcf,filename+'_freq.png','-dpng','-r300');

        case 3 % Fourier transform plot

            figure('Name','Fourier Transorms');
            subplot(2,1,1)
            stem(w_plot,abs(U));            % Create subplot of fft Y
            xlim([0 w_plot(end)]);
            ylabel('Input')
            subplot(2,1,2)
            stem(w_plot,abs(Y));            % Create subplot of fft U
            xlim([0 w_plot(end)]);
            xlabel('Frequency (rad/s)');
            ylabel('Input')

        case 4 % Theoretical vs. Experimental Time Domain Plot (step)

            if is_closed
                y_th = lsim(G_cl,u,t);
            else
                y_th = lsim(G_ol,u,t);
            end

            figure3 = figure;
            axes1    = axes('Parent',figure3);
            hold(axes1,'on');
            plot2 = plot(t,y);
            plot1 = plot(t,y_th);
            set(plot1,'DisplayName','Theoretical', 'LineWidth',2);
            set(plot2,'DisplayName','Experimental','LineWidth',2);
            ylabel('Amplitude','HorizontalAlignment','center');
            xlabel('Time (s)');
            box(axes1,'on');
            axis tight
            set(axes1,'FontName','Times New Roman','FontSize',font_size,...
                'XMinorTick','on');
            legend1 = legend(axes1,'show');
            set(legend1,'FontSize',font_size,'Location','best');

            print(gcf,filename+'_theo_vs_exp.png','-dpng','-r300');

    end
end
```

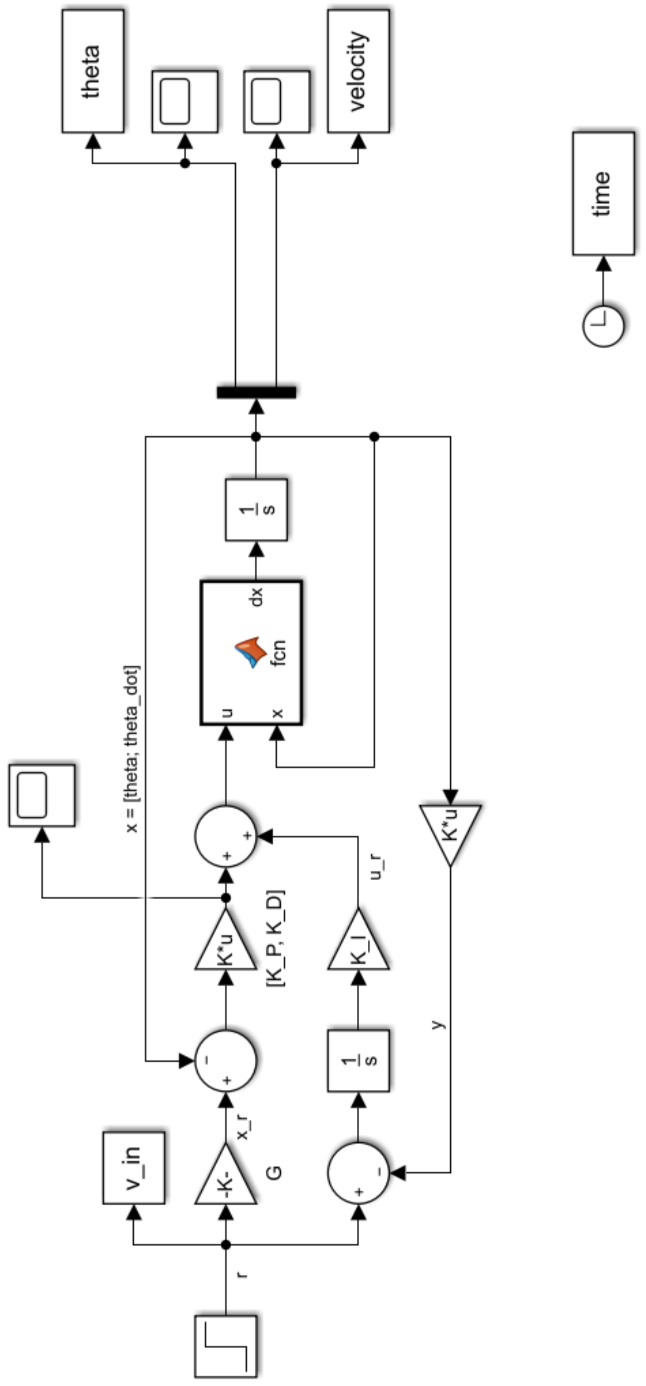
```
%% Lab 02 Full
% System Identification - Pendulum
clear all; close all; clc;

% Load in Experimental Data
load  bar_T20_N150_A.1.mat

% Experiment Params
dt = 0.002;                                   % Sampling time
T = 20;                                       % Time of experiment (Simulink: 2*T-d

% Frequencies of fft
w_nyq = pi/dt;
w_res = 2*pi/T;
w_plot = 0:w_res:2*w_nyq-w_res;               % Sampled Frequencies

% Determine Bode Plot
offset = 0.0220;                              % Resting voltage
u = v_in;                                     % Re-assign variables
y = theta;
```

```
t = (0:dt:T-dt)';
U = fft(u);                          % Assumes transient has been cut
Y = fft(y);
H = Y./U;                            % Output/Input
H(abs(U) < 1e-3) = 0;                % Remove noise in input signal

% From experimental data, determined the model of the system to be:
p = [1.5, 20];                       % system poles
temp = conv([1 p(1)],[1 p(2)]);
a = temp(3);                         %       c
b = temp(2);                         % -------------
c = 1.08 * a;                        %  s^2 + bs + a
G = tf(c,[1 b a]);

% Theoretical Bode Plot
[mag,phase] = bode(G,w_plot);
mag = squeeze(mag);
phase = squeeze(phase);

% Generate Bode Plot Experimental and Theoretical
figure,
semilogx(w_plot,db(abs(H)));
hold on
semilogx(w_plot,db(abs(mag)));
xlim([0.1,1000])

%% Controller Root Locus
% Designed a lag controller using rltool - want to increase performance of
% system while not introducing additional overshoot
figure,

% Proportional
subplot(2,2,1)
rlocus(G);

% PD (Pure zero)
subplot(2,2,2)
load RootLocusPlot_Bar\C_PD.mat
C_PD = C;
rlocus(C_PD*G);

% Lead-Lag
subplot(2,2,3)
load RootLocusPlot_Bar\C_leadLag.mat
C_leadLag = C;
```

```
rlocus(C_leadLag*G);

% PI
subplot(2,2,4)
load RootLocusPlot_Bar\C_bar_2.mat
C_PI = C;
rlocus(C_PI*G)

%% Step Responses with Controllers
figure,
hold on
[sys1,t1] = step(feedback(9.96*G,1),1.5);
[sys2,t2] = step(feedback(C_PD*G,1),1.5);
[sys3,t3] = step(feedback(C_leadLag*G,1),1.5);
[sys4,t4] = step(feedback(C_PI*G,1),1.5);
plot(t1,sys1,t2,sys2,t3,sys3,t4,sys4,'LineWidth',1.5)
legend('P','PD','Lead-Lag','PI')

%% Lead-Lag Implemented
load data_step_C_bar_2_trial_1.mat
L = C*G;
[data,t] = step(c2d(feedback(L,1),dt),20);
figure,
hold on
plot(time-1,theta);
plot(t,data);
xlim([0 20])
legend('Experimental','Model')
pos = get(gcf, 'Position');
set(gcf, 'Position',[pos([1 2 4]) pos(4)]);
plot(t,data)

%% PD with Gravity Compensator
load ('Bar Data 2_20\data_PD_gravcomp_exp_1.mat','theta','time')
exp = theta;
load ('Bar Data 2_20\data_PD_gravcomp_theo_20.mat','theta')

figure,
hold on
plot(time-1,exp,time-1,theta)

%% PID
figure
hold on
load ('Bar Data 2_20\data_PID_KI300_exp.mat','time','theta')
```

```
plot(time-1,theta)
load ('Bar Data 2_20\data_PID_KI300_theo.mat','time','theta')
plot(time-1,theta)
legend('Experimental','Model')
```


r
x
fcn
u
u
x
fcn
dx
1/s
theta
velocity


r
v_in
-K-
G
x_r
K*u
[K_P, K_D]
x = [theta; theta_dot]
1/s
K_I
u_r
y
u
x
fcn
dx
1/s
K*u
theta
velocity
time