Notebook 3: Ball and Beam System Identification

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

Mar. 19, 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. In addition to the original motor hardware, this notebook's setup includes an additional sensor to measure variable resistance for determining position. Figure 1 references the new setup of the system.

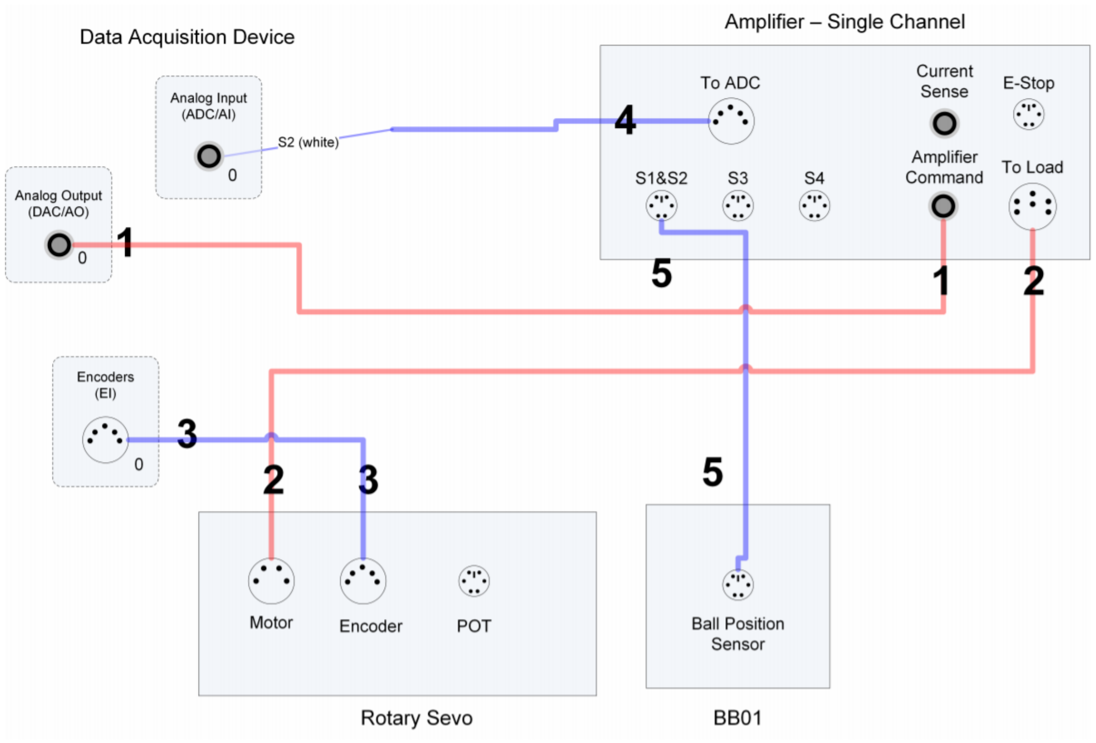


Figure 1: Connecting the SRV02 to a single channel amplifier and two channel DAQ (Image courtesy of the SRV02 User Manual). For the purposes of the lab, the tachometer output is connected directly to the S1&S2 port of the amplifier (not depicted), and the ball position sensor instead is connected to S3.

In this notebook, one objective is to design various control laws for a pendulum system and implement the compensation techniques to achieve minimal steady state error, rise time, and overshoot. The model for the pendulum system has been derived and identified using time- and frequency-based system identification in the previous notebook. Another objective of the notebook begins the design cycle again with the ball and beam module. The notebook ultimately concludes by comparing various control schemes on the new module.

2 Theory

2.1 Pendulum 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). The electro-mechanical coupling that occurs between the components is defined using coefficients k_m and k_τ, gained from the back voltage and torque-current relations of the motor.

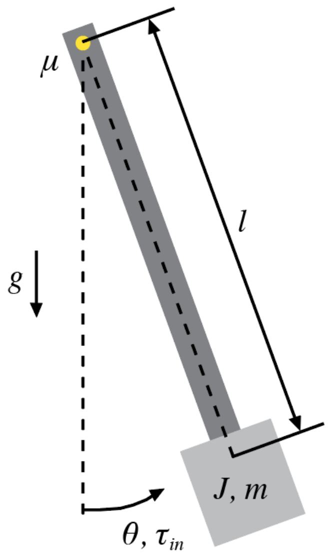


Figure 2: The pendulum setup observed in the previous lab notebook. The effects of gravity introduce a disturbance to the system, and its impact is mitigated through the use of different controller designs.

Recall the pendulum model has the following representation from the previous notebook:

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

For simplicity, the respective coefficients of $\dot{\theta}$, θ, and v_{in} are lumped and labeled as b, a, and c; Eq. (2) is used for any proceeding theoretical equation developments within this lab report.

$$\ddot{\theta} + b\dot{\theta} + a\sin\theta = cv_{in} \tag{2}$$

Assuming initial conditions are set to zero and the system is linearized around the equilibrium point of zero angular position and velocity, Eq. (2) simplifies to Eq. (3); doing so would produce less than 16% error for $|\theta| < 1$ radian. For theoretical simulations, this is considered acceptable.

$$\ddot{\theta} + b\dot{\theta} + a\theta = cv_{in} = cu \tag{3}$$

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.

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{4}$$

$$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \ddot{\theta} \end{bmatrix} \tag{5}$$

Eq. (3) can thus be constructed in state space form, Eq. (8), using the following relations.

$$\dot{x_1} = x_2 \tag{6}$$

$$\dot{x_2} \approx -ax_1 - bx_2 + cu \tag{7}$$

$$\dot{x} \approx Ax + Bu = \begin{bmatrix} 0 & 1 \\ -a & -b \end{bmatrix} x + \begin{bmatrix} 0 \\ c \end{bmatrix} u \tag{8}$$

2.1.2 State Space for Tracking a Sine Wave

One of the additional goals for the pendulum setup is to track an input sine wave, using the state space model. In order to track a sine wave, the plan input reference, u_r, needs to also be a sine wave of with desired frequency, ω, as shown in Eq. (9).

$$u_r(t) = \gamma \sin(\omega t + \beta) \tag{9}$$

Assuming a γ and β of 1 and 0, respectively, the state equations can be represented as:

$$\begin{aligned} \dot{z} &= 0z + 1(r - y) \\ \nu &= 1z + 0(r - y) \end{aligned} \tag{10}$$

In state space, the parameters a modeled by:

$$\dot{z} = Az + B(r - y) = \begin{bmatrix} 0 & \omega \\ -\omega & 0 \end{bmatrix} z + \begin{bmatrix} 1 \\ 0 \end{bmatrix} (r - y) \tag{11}$$

These replacement of u with $(r - y)$ (or e) are accounted for within the Simulink, however this change does not need to accounted for with the usage of an LQR or LQI filter explained below.

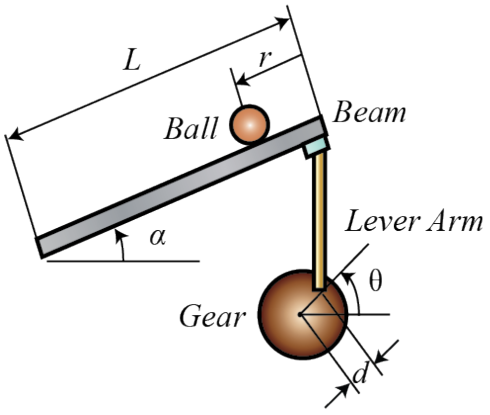


Figure 3: A model of the ball beam driven by the motor lever arm where α is near zero. (Image courtesy of University of Michigan)

2.1.3 Transfer Function

Using the Laplace transform of Eq. (3), the voltage-angle relationship of the system simplifies to an approximate second-order transfer function. Assuming the linear approximation is about $\theta = 0$ rad, Eq. (12) can be used to approximate the system.

$$\frac{\Theta}{U} = \frac{\Theta}{V_{in}} \approx \frac{c}{s^2 + bs + a} \tag{12}$$

2.2 Ball and Beam Theoretical Model

Outside of kinematics, another common way to model systems and determine theoretical models is through Euler-Lagrange equations or Lagrangian dynamics. The Lagrangian is defined as:

$$\mathcal{L} \equiv \mathcal{K} - \mathcal{V} \tag{13}$$

where $\mathcal{K}$ and $\mathcal{V}$ represent kinetic energy and potential energy of the system, respectively. In particular, the Euler-Lagrange equation is equal to:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{x}}\right) - \left(\frac{\partial \mathcal{L}}{\partial x}\right) = \tau \tag{14}$$

where the parameter x is a differentiable variable of the Lagrangian, which the system is dependent on, and τ is the generalized force. By computing the Euler-Lagrange equation, the resultant differential equation describes the dynamics of the system.

The ball and beam module, seen in 3 can be decompose into several components to obtain a model for the position of the ball. The kinetic energy of the system can be described using the angular velocity of the beam, rotational energy of the ball, and linear velocity of the ball. Using,

$$\mathcal{K} = \frac{1}{2}J_b\omega_b^2 + \frac{1}{2}mv_b^2 + \frac{1}{2}J_{beam}\dot{\alpha}^2 \tag{15}$$

where J_b and J_{beam} describe the mass moment of inertia of the ball and beam, and α is the angular position of the beam. The potential energy of the system can be described as:

$$\mathcal{V} = mgp \sin\alpha \tag{16}$$

Replacing the angular velocity of the ball in terms of the linear velocity of the ball, the Lagrangian can be simplified to the following:

$$\mathcal{L} = \frac{1}{2} J_b \left(\frac{\dot{p}}{r} + \dot{\alpha} \right)^2 + \frac{1}{2} m \left(\dot{p}^2 + p^2 \dot{\alpha} \right) - mgp \sin\alpha \tag{17}$$

Converting the expression into the Euler-Lagrange equation results in Eq. (18):

$$\left(\frac{J_b}{R^2} + m \right) \ddot{p} + mg \sin\alpha - mp\alpha^2 = 0 \tag{18}$$

By assuming the torque applied to the beam is proportional to the vertical distance the beam is displaced, it is also assumed that the linearization around α is equal to zero. The expression for α can be expressed in terms of the overall beam length (l) and radius of the gear radius (d):

$$\alpha = \frac{d}{l} \theta \tag{19}$$

Using this approximation, Eq. (18) is simplified to the expression

$$\left(\frac{J_b}{R^2} + m \right) \ddot{p} = -\frac{mgd}{l} \theta \tag{20}$$

where the system is approximated to operate about the equilibrium position of α equal to zero. The differential equation describes how the motor position affects the ball acceleration. For the purposes of system identification, the coefficients of the equation are combined into a single parameter describing the relationship between motor shaft angle θ and linear acceleration of the ball $\ddot{p}$. The resulting relation is shown below.

$$\ddot{p} = c\theta \tag{21}$$

2.2.1 Motor Dynamics

In order to incorporate the differential equation into a model of the entire system, it is important to consider the dynamics of the motor as well. Recall the differential equations that govern a DC motor, revised to reflect a force (F_y) exerted at a lever arm (d) shown in Figure 3.

$$J\ddot{\theta} = \tau - \mu\dot{\theta} - F_y d \cos\theta \tag{22}$$

$$v_{in} = Ri + L\frac{di}{dt} + v_m \tag{23}$$

$$\tau = k_\tau i \tag{24}$$

$$v_m = k_m\dot{\theta} \tag{25}$$

When combining Eqs. (22) – (25) and assuming negligible motor inductance, the resultant equation is a non-linear differential equation, as shown in Eq. (26).

$$\ddot{\theta} + \frac{\mu R + k_\tau k_m}{JR}\dot{\theta} + \frac{F_y d}{J}\cos\theta = \frac{k_\tau}{JR}v_{in} \tag{26}$$

Assuming a negligible external force on the motor shaft (F_y), the working equation simplifies further:

$$\ddot{\theta} + \frac{\mu R + k_\tau k_m}{JR}\dot{\theta} = \frac{k_\tau}{JR}v_{in} \tag{27}$$

For the purposes of system identification, the coefficients are represented by lumped variables and are determined experimentally. The resultant equation is:

$$\tau\ddot{\theta} + \dot{\theta} = Kv_{in} \tag{28}$$

where τ is a time constant and K is the DC gain of the system.

2.2.2 State Space

Representing the system in state-space is beneficial for state-feedback and state controller designs. Using the relation from Eqs. (28) & (21), the system is fully described and can be represented in state space as

$$\dot{x} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & \frac{-1}{\tau} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{K}{\tau} \end{bmatrix} v_{in} \tag{29}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} v_{in} \tag{30}$$

where the states of the system are defined as:

$$x = \begin{bmatrix} p \\ \dot{p} \\ \theta \\ \dot{\theta} \end{bmatrix} \tag{31}$$

One detail to note about the system is that the sensor data does not encompass all the states. In order to utilize all the states of the system, a form of state estimation is implemented to obtain an estimated measurement. There are several filters that provide state estimates; of the most famous are a Kalman filter and a derivative filter. In this notebook, the derivative filter modelled by Eq. (32) is used to estimate the the velocity $\dot{p}$ of the ball by taking the derivative of the sensor measurement p.

$$T(s) = \frac{s}{(s/\omega_o + 1)^2} \tag{32}$$

The derivative filter acts as a derivative until a particular frequency of ω_o where it drops off.

2.2.3 Transfer Function

The transfer function from applied voltage to the location of the ball is useful in ultimately controlling the system. However, arguably more important is the decomposition of the governing transfer function into decoupled components that can be experimentally identified. The relationship between applied voltage and output motor shaft angle, as a transfer function, can be described as:

$$\frac{\Theta(s)}{V_{in}(s)} = \frac{K}{s(\tau s + 1)} \tag{33}$$

These are the parameters that are intrinsic to the motor itself. In particular, it is useful to perform the same identification to determine if the assumption that the resting load on the motor lever arm is negligible in the dynamics of the system. If it is appropriate to use, the motor identification provides the same parameters as previously identified with the motor itself.

In addition to the relationship between applied voltage and motor position, the secondary relationship related to the new module is the dynamics of the ball with respect to the beam elevation angle. Given the approximately proportional relationship between motor output angle, the equation of motion is represented by Eq. (21) or as a transfer function by:

$$\frac{P(s)}{\Theta(s)} = \frac{c}{s^2} \tag{34}$$

The multiplication of transfer functions in Eq. (33) and Eq. (34) determine the overarching model and the identified parameters can be applied equivalently to the state-space equations.

2.3 System Identification Methods

2.3.1 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}). The steady state value is related to the DC-gain of a system and can be used to quickly determine relationships between 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 $OS = \exp(-\pi\zeta/\sqrt{1-\zeta^2})$. For this lab, the step response technique is used to model the dynamics of an unstable system over a short time-scale. The ball rolling down the beam is an unstable relationship, the position of the ball grows exponentially.

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 provide 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 of the motor; however, the time domain is used to compare the rise times, overshoots, and steady state error with different values of the controller inputs.

2.3.2 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 (Δt) 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.

2.4 Controller Design

The efforts succeeding the ball and beam system identification involve the use of feedback control, shown in Figure 4, 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: PID controller, LQR controller, and LQI controller.

Figure 4 illustrates the general feedback loop, applicable to both the pendulum and the ball and beam systems. The reference input, V_{in}, is the desired input of the system, which defaults to $1V$ for most controller experiments of the lab. In general, the objective of the ball and beam system is to reject disturbances and center the ball on the beam. The system output, $P(s)$, represents the position of the ball along the beam as defined in Figure 3. The goal for the pendulum is to reach the desired angle, $\Theta(s)$.

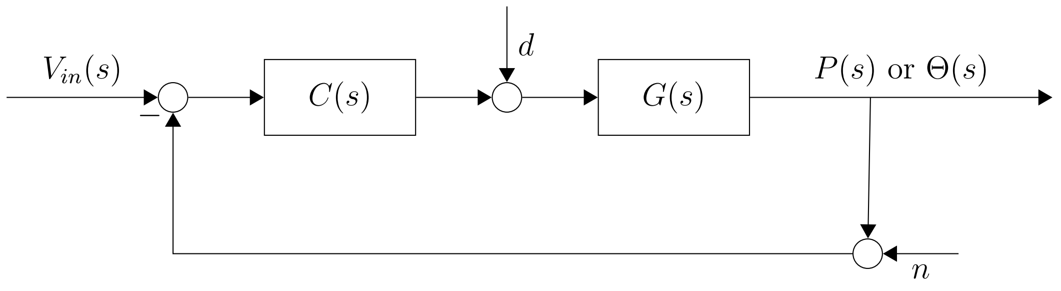


Figure 4: General closed-loop feedback system. In this instance, $R(s)$ represents the input voltage, which is set to $1V$ for comparative studies. $C(s)$ is the controller design of the feedback system, explained further in proceeding subsections. $G(s)$ is the ball and beam or pendulum transfer function with input $U(s)$, and output ball position $P(s)$ or pendulum angle $\Theta(s)$. **Note:** V_{in} represents the reference input voltage, analogous to reference position/angle, and not the input voltage of the Quanser motor.

2.4.1 PID Controller

A PID controller is a widely used control law that provides the benefits of PD control while also introducing an integrator to drive steady state error to zero. By tuning the controller with the three gains K_P, K_D, and K_I, a desired overshoot, rise time, and steady state value can be achieved. The controller has the form of Eq. (35), and has a DC gain of 1 from reference to output in closed loop.

$$C(s) = K_P + K_D s + \frac{K_I}{s} \tag{35}$$

2.4.2 LQR Controller

A linear-quadratic regulator (LQR) controller implements the theory of optimal control law. It implements a cost function and seeks to minimize the resultant cost, which provides controller designers the opportunity to weigh various feedback loop parameters more or less than others. The LQR controller is represented according to the pendulum's state space form shown in Eq. (8). This type of controller is especially useful when PID control is difficult/infeasible to use or when the state vector is greater than size 2x1.

To commence this optimization technique, the designer must first define continuous-time cost function parameters in state space form, shown in Eq. (36).

$$J_c = \int_0^{\infty} [(x - x_r)^T Q (x - x_r) + (u - u_r)^t R (u - u_r)] dt \tag{36}$$

where x is the state vector of a feedback loop and u is the plant input. Vectors x_r and u_r represent the desired reference outputs of the feedback loop. Square matrix Q with size NxN is the weight matrix associated with state vector x of size Nx1. Square matrix R of size MxM and is the weight matrix associated with M plant inputs u.

For the pendulum problem, Q is a 2x2 matrix, where $Q_{1,1}$ places weight on θ and $Q_{2,2}$ places weight on $\dot{\theta}$. Similarly, R is a scalar weight parameter that impacts the single plant input, u. For the ball and beam problem, Q is a 4x4 matrix, where $Q_{1,1}$ places weight on p, $Q_{2,2}$ places weight on $\dot{p}$, $Q_{3,3}$ places weight on θ, and $Q_{2,2}$ places weight on $\dot{\theta}$.

MATLAB function, `lqr`, is used to find appropriate K values associated with the values for state space vector, x. In the pendulum instance, the output of `lqr` is analogous to $K = [K_P, K_D]$, though the same is not for the ball and beam module.

The sensitivities of these weights are assessed further in the Implementation section, however general rule of thumb suggests that increasing weight parameters of Q or R prioritizes the effort for state elements to reach their corresponding references system, so as to minimize cost function, J_c.

2.4.3 LQI Controller

The linear-quadratic integrator (LQI) controller is similar to the LQR filter but with an added integrator control law, as shown in Figure 5.

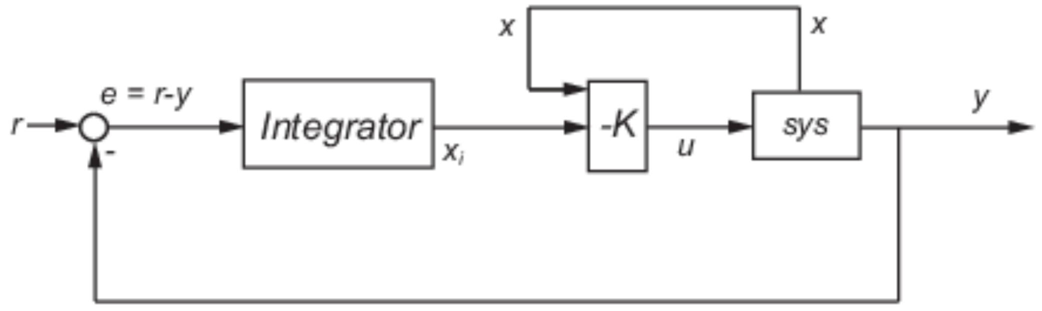


Figure 5: LQI diagram from MATLAB function, `lqi`, documentation. This diagram illustrates how one can implement the output of the function, K to a Simulink model.

The only differences in form between LQR and LQI for the pendulum application is that the Q matrix is of size 3x3 instead of 2x2, and the output K is a 3x1 vector instead of 2x1. $Q_{3,3}$ is the weight parameter that affects the emphasis for error, e, shown in Figure 5. Additionally, the output of MATLAB's `lqi` function is $K = [K_P, K_D, K_I]$ for the pendulum application. The ball and beam module is not assessed using an LQI filter due to the inability to reach this point prior to the in-person lab shutting down.

3 Implementation

The frequency method validated in the previous Lab Notebooks is used on the motor system to identify a second-order model for the voltage-theta relationship. The setup is in open loop for the determining the parameters τ and K from Eq. (33). In addition to the motor dynamics, the ball and beam model parameters is identified while attached to the motor.

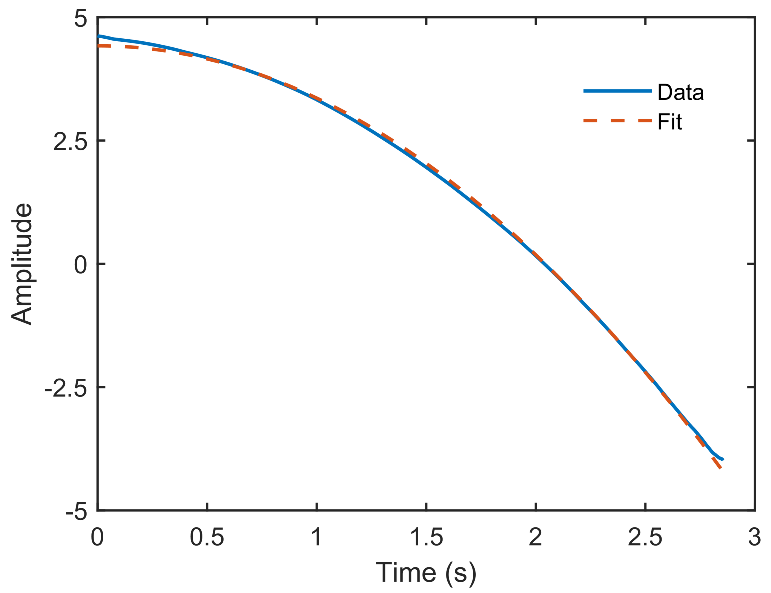


Figure 6: The experimental step response to the unstable ball beam system. The trend is as expected for a double integrator, where position exponentially grows with respect to time. The fit corresponds to a coefficient of c= -1.059.

The identification will use a step response via the motors output shaft angle to determine the acceleration profile of the ball, thereby evaluating the parameters from Eq. (34). Once the experimental models of the system are determined, the controllers described in the proceeding sections are implemented through Simulink.

3.1 System Identification

3.1.1 Ball and Beam

The ball and beam module is interesting because the ball accelerates down the beam when disturbed by a beam angle, inherently being an unstable system. This can be seen by taking the inverse Laplace transform of Eq, (34) multiplied with a step input ($1/s$),

$$p(t) = ct^2 \tag{37}$$

The coefficient c is negative, such that the profile of the ball position with respect to time is a negative quadratic. To take advantage of this, the motor system is placed in negative feedback to allow for a reference. In open loop, a step input would continually increase the angular position θ; however, in closed loop the step response is with respect to a reference angular position. The response is recorded until the ball collides with the stop. Afterwards, a quadratic can be fit to the experimental data thereby providing the value of c. It is important to note that the signal data spans approximately (-5,5) volts, suggesting the fit is required to include an offset. This is only to adjust the start position of the fitting function and is unrelated to the dynamics of the ball and beam system.

The experimental data of the ball beam and the associated fit can be seen in Figure 6.

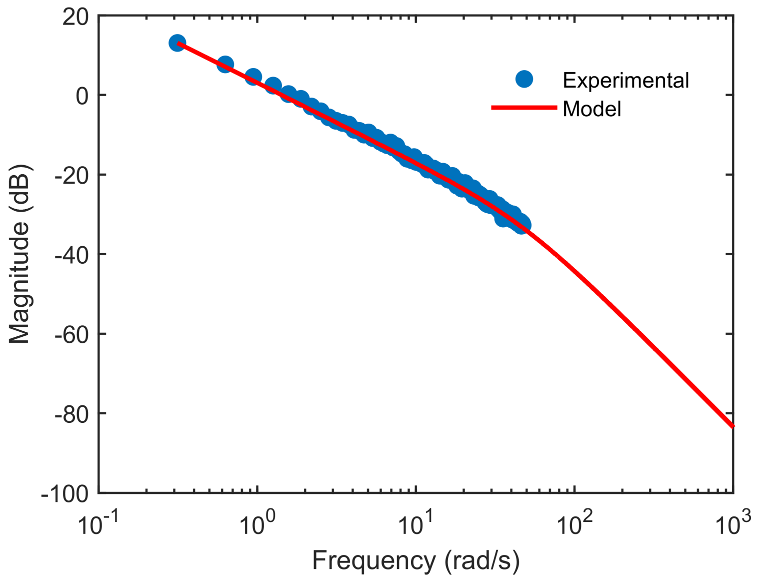


Figure 7: Experimental Bode plot of motor system with estimated pole at 47.6 and DC gain of 1.421, verifying the additional force that the beam exerts on the motor is negligible.

where the parameter c corresponding to the relationship between motor shaft angle and ball position is measured to be -1.059.

3.1.2 Motor Dynamics

To determine the characteristics of the motor system and verify that the parameters identified in the first Lab Notebook remain the same, a multi-sine input is used to produce frequencies up to approximately 50 rad/s to collect points along the magnitude Bode plot. The ball and beam module is attached to the motor shaft by a lever arm with the ball removed from the system. Using the theoretical model described by the transfer function in Eq. (33), the experimental data collected is used to determine the pole location $1/\tau$.

The experimental Bode plot of the motor system with the estimated model are shown in Figure 7. 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 $\Delta t = 2$ 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 $\pi/\Delta t$. From the plot, it is estimated that the pole of the system occurs at $p = 47.6$, giving a time constant parameter $\tau = 0.021$. An approximation of the DC gain is 1.421. The resultant system is identified to have a transfer function of the form in Eq. (33) with values of $[0.021, 1.421]$ for $[\tau, K]$ respectively.

3.2 Controller Design

With the system identified, it is necessary to define closed loop performance requirements in order to implement a control law. Due to the ball beam module being a fourth order system, it is unlikely to directly correlate metrics such as rise time, settling time and percent overshoot. Additionally, a root locus approach to the controller design is difficult due to number of poles. Placing additional poles and zeros have a direct effect on the closed loop pole locations of the system. Using proportional control or otherwise, the control law is complex. Instead, the state space approach and state feedback is used to ensure stability and approximate performance. In particular, we can determine the desired location of the closed loop poles and design a state feedback controller to satisfy those requirements. The control law can then be simulated and verify the controller is not saturated to be implemented on the system. For the analysis of the ball and beam, the reference input (v_{in}) and the system output are set to 0 V to place the ball in the center of the beam.

3.3 State Feedback Implementation

3.3.1 Ball and Beam

In order to implement this experiment in state feedback, the block diagrams are adjusted to compensate for MIMO systems. In the case of the ball and beam, the states described in Eq. (31) are the ball position along the beam, the linear velocity of the ball, motor shaft angular position, and angular velocity of the motor shaft, as shown in Eq. (31). 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 8 using this functionality. In the case of the ball beam module, the state feedback gains are difficult to calculate analytically, and are computed by the MATLAB command `place` for MIMO systems to determine the position of the closed loop poles. The operation solves for the eigenvalues of the matrix $A - BK$ where the matrix K corresponds to the gain. In practice, this is used to determine the K matrix, however for a PID controller, the values have significance. The gains can correspond to the values of K_P and K_D, and then is iterated to determine the value for the integral gain. The results are effectively the same as calculating analytically for lower-order systems, but the state feedback approach can be used for more complex systems.

The ball and beam system is a higher order system, fourth order. As previously mentioned, the derivation of pole locations from performance constraints does not have simple estimations as are relevant for first or second order systems. However, in an attempt to determine the desired closed loop poles, the approximate speed of the response of the system can be set and the roots to a polynomial representing a general fourth order system is used as the desired closed loop locations, shown in Eq. (38).

$$d_4 = s^4 + 2.1w_n s^3 3.4w_n^2 s^2 + 2.7w_n^3 s w_n^4 \tag{38}$$

Theoretically, the step response does not saturate the controller for an w_n of 3.5 and contains minor oscillations as seen in Figure 9.

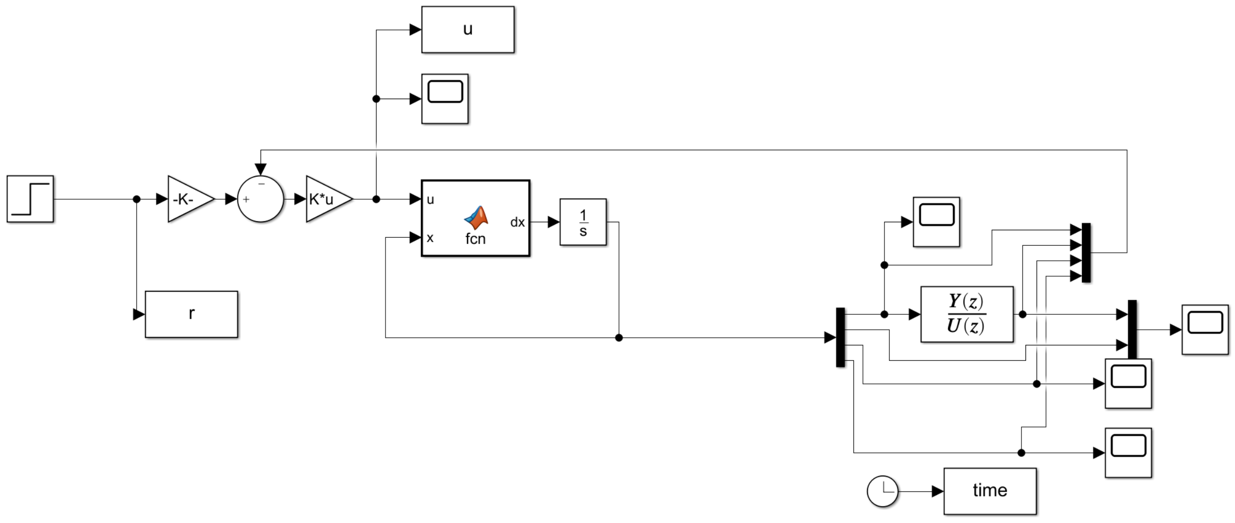


Figure 8: Simulink block utilizing state feedback of the Quarc's sensor measurements. The control law implemented in this case does not have significant intuition but places the closed loop poles of the system at their desired locations of $(-1.4839 + 4.4205i)$,and $(-2.1911 + 1.4495i)$ with multiplicity of two.

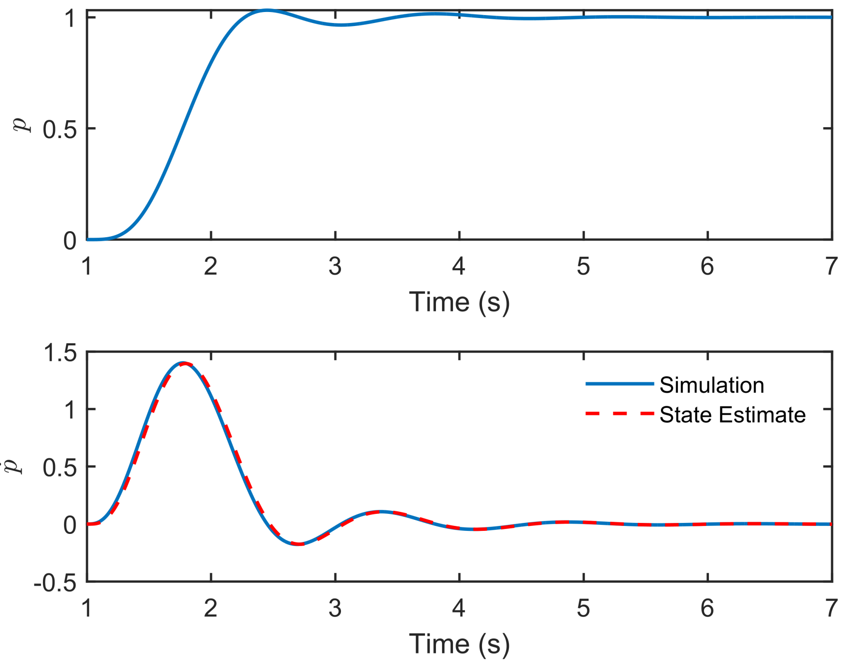


Figure 9: The theoretical step response of the system in state feedback with using the closed loop pole locations derived from Eq. (38) and a derivative filter Eq. (32) with ω_o of 100.

Additionally, in order to properly implement the state feedback with all of the states a state estimate is needed for the linear velocity of the ball. As mentioned in the theoretical section, the derivative filter is implemented of the form 32 with an ω_o of 100 rad/s. The z-transform of the transfer function is taken to convert the theoretical controller into an implementable digital version. When the gain matrix K and the derivative filter are implemented, the response exceeds the small angle approximations baked into the formulation of the system parameters and produces motor angles exceeding 3 radians or half of a revolution. In general, the control law is adequate and stabilized the system theoretically, but the performance is limited by explicitly determining the locations of the closed loop poles. Unsurprisingly, the system does not perform as expected due to the large variations in motor angle.

3.4 LQR Control

3.4.1 Analyzing Sensitivity of Parameters

The first step in implementing the LQR filter is first to assess and understand the sensitivities of $Q_{1,1}$, $Q_{2,2}$, and R. Figures 10 – 12 demonstrate the sensitivities of each parameter for the pendulum setup. In each respective figure, the other parameters are set equal to 1.

Figure 10 demonstrates the impact of $Q_{1,1}$ on the both $u(t)$ and $y(t)$. As can be seen, an increased emphasis of this parameter results in quicker rise times, which makes sense because the error of θ is a larger priority for the pendulum system. If the emphasis of $Q_{1,1}$ is far greater than $Q_{2,2}$ and R, however, $u(t)$ starts to exceed Quanser's saturation voltage of 10V. Thus, based off this analysis alone, it's important to maximize the weight of $Q_{1,1}$ to ensure speedy rise times.

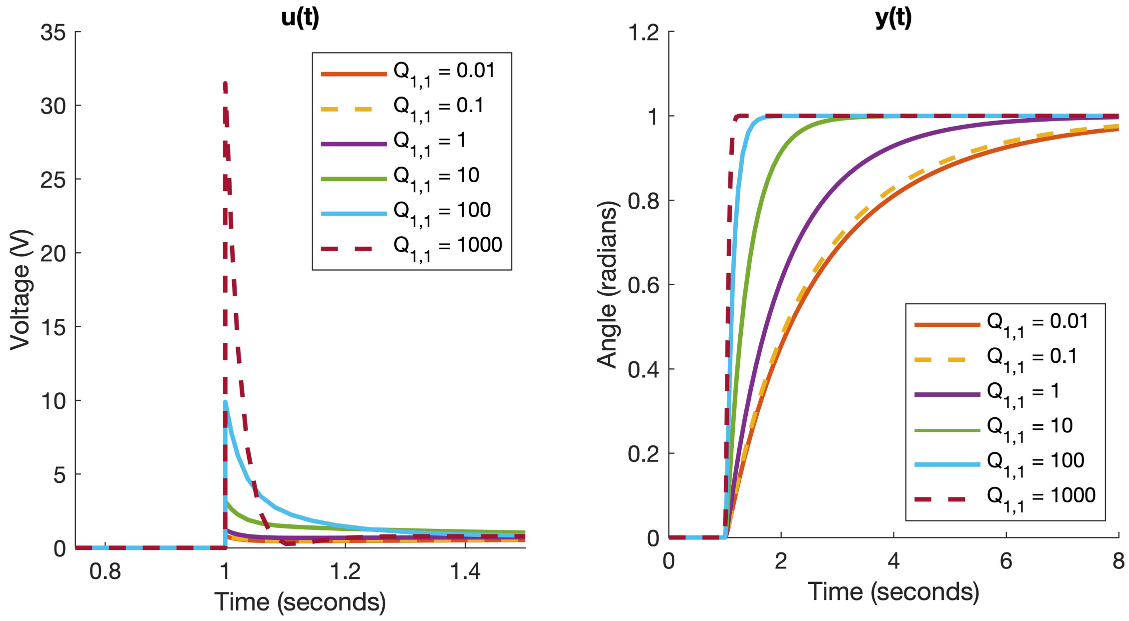


Figure 10: Sensitivity analysis for $Q_{1,1}$. As can be seen the $\mathbf{y(t)}$ plot, an increase in $Q_{1,1}$ results in a quicker rise time, however the $\mathbf{u(t)}$ plot demonstrates an increase in input voltage.

When observing the impact of increasing $Q_{2,2}$, one can see that increasing this parameter results in a quicker steady state angular velocity; though this may result in high steady state error of the pendulum's angle since the velocity settles quicker with no overshoot to achieve the desired angle. This proves, however, increasing $Q_{2,2}$ results in decreasing settling time of the angular velocity, which makes sense because $Q_{2,2}$ is associated with reducing the error in $\dot{\theta}$.

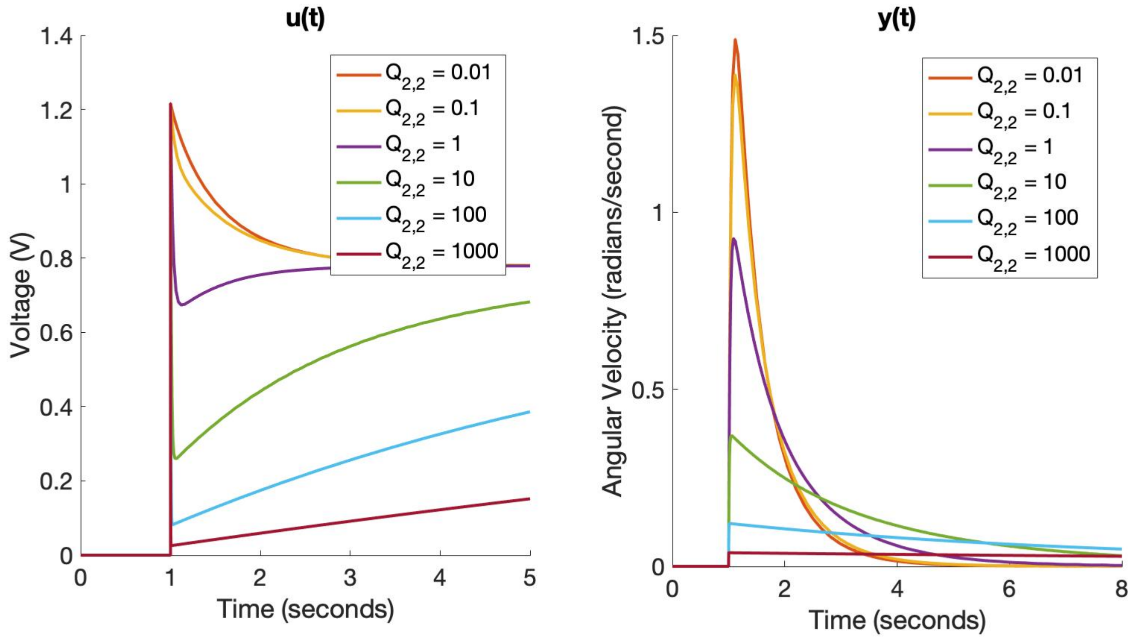


Figure 11: Sensitivity analysis for $Q_{2,2}$. Since $Q_{2,2}$ is associated with K_D, increasing it results in quicker steady state error for the pendulum's angle, though potentially not at the desired reference of 1.

Observing the effects of increasing R, one can see that an increased R results in the reduced steady state value of $u(t)$, while the rise times for each $u(t)$ plot is identical. In this analysis, we can conclude that the system is stable when $R \geq 0.1$ when $Q_{1,1} = Q_{2,2} = 1$.

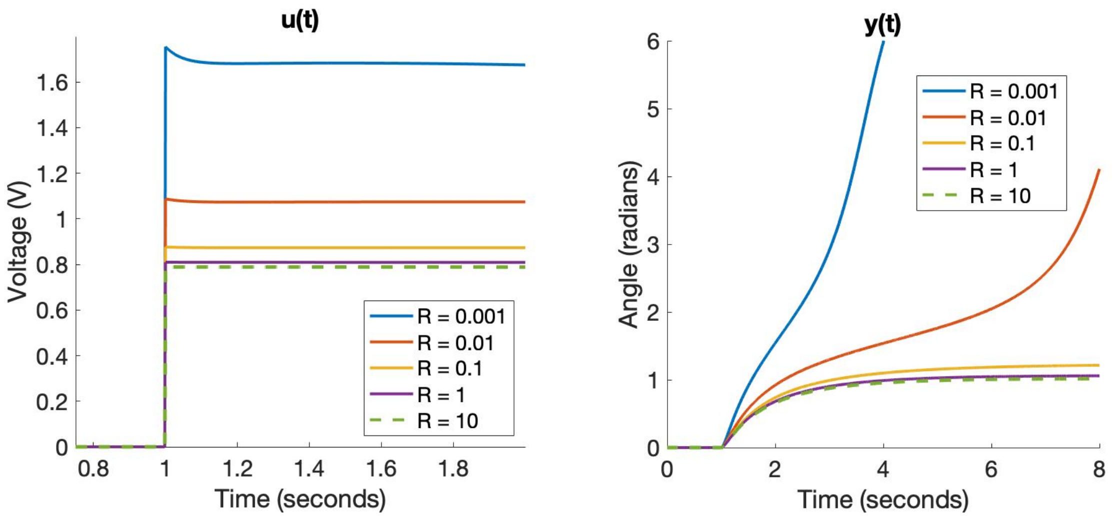


Figure 12: Sensitivity analysis for R. As can be seen in the $\mathbf{u(t)}$ plot, increasing R results in a lower steady state input voltage.

An additional observation to note for this analysis is the magnitudes of $Q_{1,1}$, $Q_{2,2}$, and R are relative to each other. That is, if $Q_{1,1} = Q_{2,2} = R$ for any magnitude, the results is the same. Thus, it's important to prioritize which parameters are most important. In this analysis, $Q_{1,1}$ is the priority.

3.5 Optimal LQR Parameters

3.5.1 Pendulum

After running several simulation on MATLAB with different weights, the final selected magnitudes are $Q_{1,1} = 100$, $Q_{2,2} = 0.01$, and $R = 1$. Thus, the utmost priority to settle quickly is θ, which is weighted at two magnitudes greater than R, the second most prioritized parameter. Based on Figures 10 – 12, the order of these priorities makes sense since $Q_{1,1}$ achieved minimal steady state error and rise times, and relatively higher R values maintain stability.

These parameters yield K_P and K_D values of 9.1168 and 0.343, respectively, and also results in a step response with a peak $u(t)$ of 9.896V, a rise time of 0.137 seconds, and a 1-radian steady state value with no overshoot. In fact, no overshoot is exhibited using any of the tested weight parameters. This is attributed to the setup of this cost function, specific to the step input.

The final experimental results are a steady state of 0.987 radians and a rise time of 0.137 seconds, which is remarkably close to the simulated expectations. These results are the optimal results for reducing the rise time without saturating the input voltage. Further efforts to improve rise time / controller design are thus considered for the pendulum setup using an LQI filter instead because of its ability to allow overshoot.

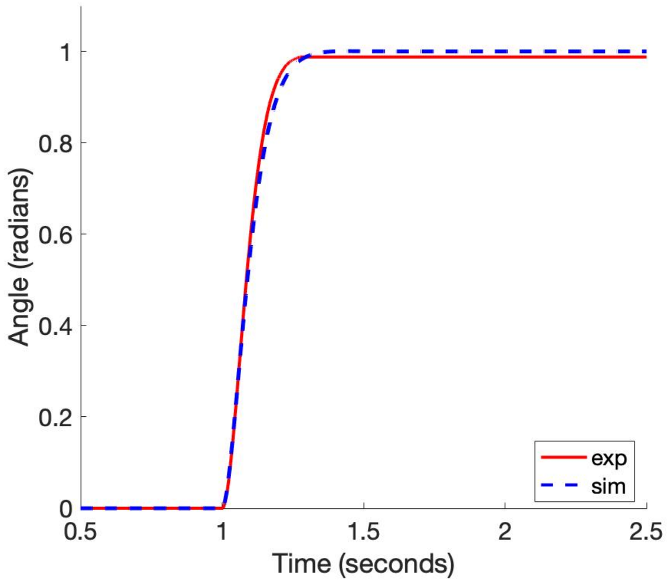


Figure 13: Experimental vs. theoretical data for selected LQR filter. The experimental steady state value is 0.987 radians and a rise time of 0.137 seconds, which are identical to the simulated test.

3.5.2 Ball and Beam

The ball and beam system is not successful with state feedback using defined closed loop pole placement. One method of accomplishing better performance is through the use of LQR, which is more suitable because the controller optimizes the control structure based on different weighting. For the ball and beam system, this is implemented by weighting the error with respect to ball position the highest. The value corresponding to this is $Q_{1,1}$ which is set to 100 based off the error weight of 0.05. The rest of the weights are adjusted to avoid saturation and maintain stability. To test the controller, the system was subject to a pulsed reference of $\pm 2V$ at a period of 15 seconds with a 50% duty cycle. The result is a waveform that varies changes position every 7.5 seconds and holds the position for the same amount of time.

As seen in Figure 14, the experimental response mimics the theoretical response closely but does not exhibit the same overshoot as is seen in the theoretical. A possible explanation for the error may be due to the signal noise of the sensor, when acquiring the data for system identification the results fluctuated significantly. Another reason may be due to additional friction or momentum of the ball rolling down the ramp or deviation from the linear approximation about the equilibrium position of the beam.

3.6 LQI Filter for Pendulum

An additional sensitivity analysis is conducted to decide upon a $Q_{3,3}$ value for the LQI filter. For this analysis, the selected values of $Q_{1,1}$, $Q_{2,2}$ that are used for the pendulum's LQR filter are implemented for proper $Q_{3,3}$ selection. As shown in Figure 15, lower $Q_{3,3}$ values

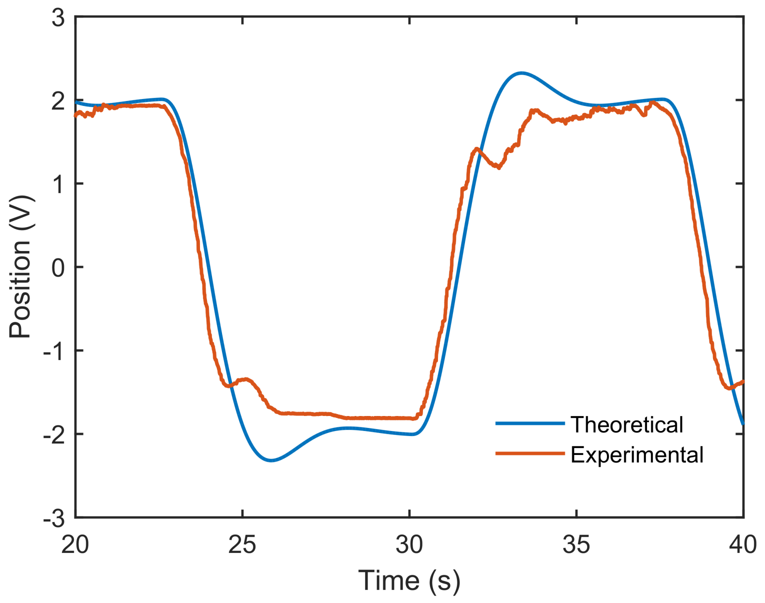


Figure 14: An LQR controller for tracking a pulsed wave at a frequency of 15 seconds. The parameters for the controller are $Q_{1,1} = 400$, $Q_{2,2} = 1$, $Q_{3,3} = 0.01$, $Q_{4,4} = 0.01$, with $R = 100$.

are not sufficient for further reducing the rise time or settling time. A $Q_{3,3}$ selection also exceeds the input saturation voltage, leaving the selection of $Q_{3,3} = 100$.

Using the above weight parameters, the output of MATLAB function `lqi` is $K = [K_P, K_D, K_I] = [10.1, 0.377, -10]$, which theoretically yields a 1-radian steady state value, a rise time of 0.1574 seconds, and an overshoot of 1.54%.

When testing these parameters experimental, the experimental demonstrate identical behavior to that of the simulated run, as seen in Figure 16. The experimental steady state value is 1.0017 radians, which is the closest encoder output to 1 radian. The experimental overshoot is 0.5% and the rise time is 0.1384 seconds, which are both less than the theoretical outcomes. Though the rise times are not improved through LQI implementation, it's important to recall that the LQI filter presents advantages in control law if the system is not known.

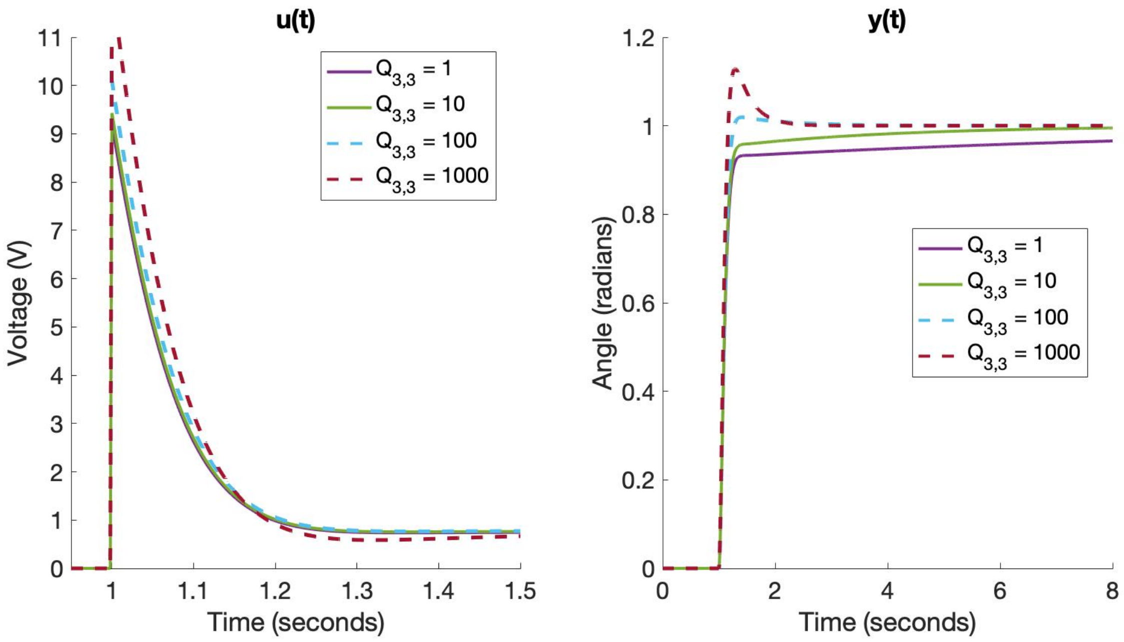


Figure 15: Sensitivity analysis for $Q_{3,3}$. Both a $Q_{3,3}$ of 100 and 1000 present similar rise times, however the 1000 weight exceeds the input voltage and also present more overshoot. Thus a $Q_{3,3}$ of 100 is selected.

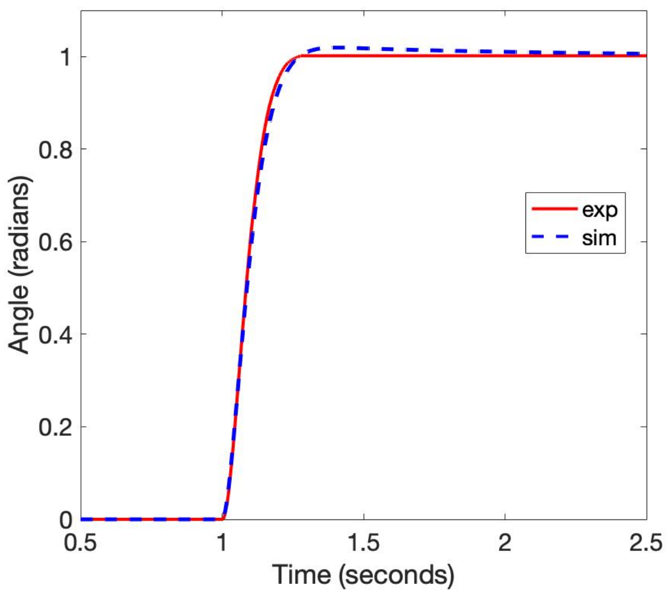


Figure 16: The step response of the pendulum system with an LQI controller to mitigate overshoot and eliminate steady-state error.

3.6.1 Using LQI Filter for Tracking a Sine Wave

The final observation is to assess whether the LQI filter is sufficient for tracking a sine wave with the pendulum system. Incidentally, the best tracking for a sine wave of 1 rad/s are with the PID values used for the step input. Figure 17 demonstrates the similarity; the experimental data portrays a slight phase delay, however we consider the tracking abilities sufficient for the sine wave reference

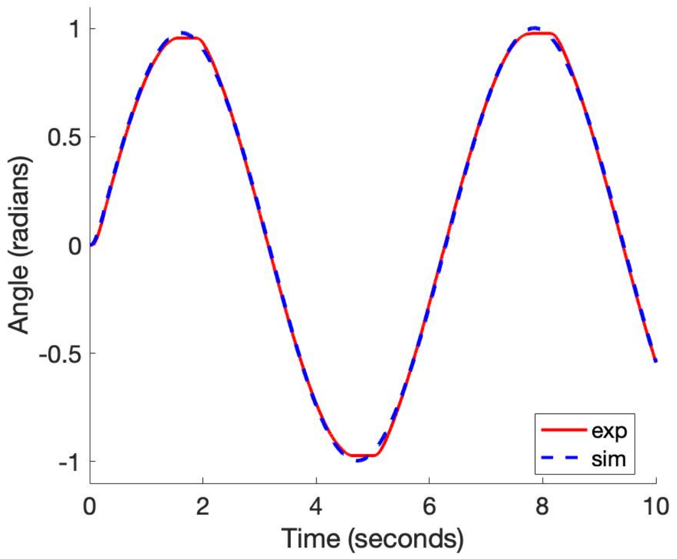


Figure 17: The use of an LQI filter for tracking a sine wave. A slight phase delay between the experimental and theoretical data is notable, however it is sufficiently small in this application.

4 Conclusion

In this notebook, the theoretical model of the system is derived based on electro-mechanical equations of motion for a dc motor and the Lagrangian of the ball and beam module. In series, the two sub-system transfer functions describe the overarching behavior of the module. The parameters of the motor 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 and the DC gain. The behavior of the ball and beam is modelled using a step response due to the unstable behavior of a ball moving along on an inclined plane. The response to a step input of the motor shaft angle produces a negative quadratic response that is fit to a modelling coefficient, shown in Eq. (38). Converting the coefficients of the transfer system's to a general form allows for an additional representation in state space to perform state feedback. The preceding methods of identification produce an estimated DC term of $K = 1.421$ with a poles located at 47.6

rad/s for the motor model. The ball and beam is identified to have a coefficient of -1.059 describing the relationship between motor shaft angle θ and ball position along the beam p.

In addition to identifying the model of the system, a large component of the lab is focused on controller design. Of the various controllers considered in this lab, PID and root locus methods of design were considered infeasible since the ball and beam module is a fourth-order system. State feedback is implemented using an approximation for placing closed loop poles based on the desired speed of the system. The results produce motor shaft angles exceeding half a revolution, violating the small angle approximations made to construct the model of the overall system. In theory, however, the performance is acceptable and the derivative filter designed with a cutoff ω_o of 10 rad/s reasonably estimated the linear velocity of the ball $\dot{p}$. The more applicable controllers that were successfully implemented were using LQR and LQI control. In the case of the pendulum system, both the LQR and LQI systems produced positive results that mimic the intended simulations; both experimental outcomes exhibit quicker rise times than the theoretical as well. Though the LQI filter experiments demonstrates a slower rise time than the LQR one, the steady state error is closer with LQI by a slim margin. For this reason, and the fact that the LQI filter can account for systems that may be difficult to identify, the LQI filter is considered a more appropriate filter for this application if we omit gravity compensation from the pendulum.

The ball and beam system is tuned using LQR to provide improved performance over the direct pole placement method using a static controller gain matrix K. With the use of LQR the system is stable and performed similar to theoretical simulations. When developing the controller, the state estimate of $\dot{p}$ was observed to have large error upon the beginning of the experiment. This is remedied by bounding the output to ± 2 (rad/s). Any estimate greater or less than the bounds saturates, resulting in a more accurate value of $\dot{p}$ to be used in state feedback. The implemented controller does not exhibit the overshoot of the theoretical model, likely due to deviation from linear approximations or more likely sensor noise. In future controller designs for the ball and beam, the sensor measurements for ball position along the beam can be passed through a low-pass filter before being introduced to the derivative filter for estimating linear velocity of the ball. A more advanced control law such as LQI can also be implemented and provide more favorable results with respect to rise-time.

5 Appendix II: MATLAB Code

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

%% Experiment Definition
% Define Parameters
dt = 0.002;                        % Sampling time
T = 20;                             % Time of experiment
N = 150;                           % Number of sine waves
A = 30 / N;
i = (1:N)';
w = 2*pi/T*i;                      % Frequency
phi = 2*pi*rand(1,N)';             % Random Phase

% 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

% figure(),plot(t,u)
```

```
%% Load experimental data
load motor_id_bb.mat;
theta = exp_model.theta;
theta_dot = exp_model.theta_dot;
p = exp_model.p;
u = exp_model.u;
t = exp_model.t;
%% Theoretical Parameters

% Frequencies of fft
w_nyq  = pi/dt;
w_res  = 2*pi/T;
```

```
w_plot = (0:w_res:2*w_nyq-w_res)';

K   = 1.421;
tau = 0.021;

num = K;
den = [tau 1 0];

% Obtain discrete bode plot of transfer function
G = tf(num,den);
[mag,phase] = bode(G,w_plot);
mag   = squeeze(mag);
phase = squeeze(phase);


%% Post Processing Script
offset = 0.0220;                              % Resting voltage
y = theta;
t = (0:dt:T-dt)';

% Plot input and output
figure
subplot(2,1,1)
plot(t,u);
subplot(2,1,2)
plot(t,y);

% Fourier Transform
U = fft(u);                     % Frequencies of input signal
Y = fft(y);                     % Frequencies of output signal
figure,
subplot(2,1,1)
stem(w_plot,abs(Y));            % Create subplot of fft Y
xlim([0 w_plot(end)]);
subplot(2,1,2)
stem(w_plot,abs(U));            % Create subplot of fft U
xlim([0 w_plot(end)]);

% Transfer Function
H = Y./U;                       % Output/Input
H(abs(U) < 1e-3) = 0;           % Remove noise in input signal

% Overlay theoretical and experimental
figure,
semilogx(w_plot,db(abs(H)),'LineWidth',2);     % Experimental
```

```
hold on
semilogx(w_plot,db(abs(mag)),'r','LineWidth',2);   % Theoretical
legend('Theoretical','Experimental')
```

```
%% Step identification ball beam
clear all; close all; clc;

% Load data
load ball_id_bb.mat
AVG = 400;
BOUNDS = [1.7281 4.5853];
% BOUNDS = [];
funct = @(c,t) c(1)*t.^2 + c(2);
tf_model = @(c) tf(c,[1 0 0]);

% Plot experimental data
if isempty(BOUNDS)
    figure, plot(time,p);
    uiwait(msgbox('Select an x-value from which to crop','modal'));
    [BOUNDS, ~] = ginput(2); % user select an x-value from which to crop.
end

% Filter data
loc = time>BOUNDS(1)&time<BOUNDS(2);
t_data = time(loc)-time(find(loc,1));
p_data = p(loc);
p_filt = smooth(p_data,AVG);

% Fit data and create model
% NOTE: The offset coefficient will be ignored because the ideal position
% is the center of the beam where the sensor reads zero.
coeff = nlinfit(t_data,p_filt,funct,[-2,4.2]);
exp_model = tf_model(coeff(1));

% Plot Results
figure, hold on
plot_prop.LineWidth = 1.5;
plot(t_data,p_filt,plot_prop)
plot(t_data,funct(coeff,t_data),plot_prop)
xlabel('Time (s)')
ylabel('Amplitude')
legend('Data','Fit')
```

```
%% Ball Beam Controller Design
% clear all; close all; clc;
```

```

dt = 0.002;
T  = 20;

% Theoretical model
c = -1.0588;
K   = 1.421;
tau = 0.021;
ball_model = tf(c,[1 0 0]);          % Ball contribution
motor_model = tf(K,[tau 1 0]);       % Motor contribution
bb_model = ball_model*motor_model;   % System model

% Root locus infeasible (proportional or otherwise)
%% State Space
A = [0 1 0 0; 0 0 c 0; 0 0 0 1; 0 0 0 -1/tau];
B = [0;0;0;K/tau];
% C = [1 0 0 0; 0 0 1 0; 0 0 0 1];
C = [1 0 0 0];
D = 0;
bb_ss_model = ss(A,B,C,D);       % State Space Model
w_n = 3.5;
test = roots([1 2.1*w_n 3.4*w_n^2 2.7*w_n^3 w_n^4]);

% State Feedback
K_c = place(A,B,test);

% Need estimator for velocity of ball - Derivative filter
s = tf('s');
w_0 = 100;
filt_deriv = s/((s/w_0 + 1)^2);
filt_deriv_z = c2d(filt_deriv,dt);
num = filt_deriv_z.Numerator{1};
den = filt_deriv_z.Denominator{1};

% Did not implement - wierd theta

%% LQR
close all;
r_0 = 0;              % Initial Condition
error_p = 0.05;
error_dp= 1;
error_th = 10;
error_dth = 10;
error_u = .1;
Q11 = 1/(error_p^2);
```

```
Q22 = 1/(error_dp^2);
Q33 = 1/(error_th^2);
Q44 = 1/(error_dth^2);
Q = diag([Q11,Q22,Q33,Q44]);
R = 1/(error_u^2);

[K_c,S,CLP] = lqr(A,B,Q,R);
sim Controllers\StepBlock_lqr_bb

% run saveModelData.m

figure,
subplot(3,1,1)
plot(t,u);
title("Q1 = " + num2str(Q11) + ", Q2 = " + num2str(Q22) + ", R = " + num2str(
ylabel('u')
subplot(3,1,2)
plot(t,x(:,1));
ylabel('p')
subplot(3,1,3)
plot(t,x(:,2));
ylabel('$\dot{p}$','interpreter','latex')


rt = risetime(x(:,1),t)

%% LQI

close all;
r_0 = 0;            % Initial Condition
error_p = 0.05;
error_dp= 1;
error_th = 10;
error_dth = 10;
error_u = .1;
error_e = 0.1;
Q11 = 1/(error_p^2);
Q22 = 1/(error_dp^2);
Q33 = 1/(error_th^2);
Q44 = 1/(error_dth^2);
Q55 = 1/(error_e^2);
Q = diag([Q11,Q22,Q33,Q44,Q55]);
R = 1/(error_u^2);

[K_c,S,CLP] = lqi(bb_ss_model,Q,R);
```

```
sim Controllers\StepBlock_lqi_bb

% run saveModelData.m

figure,
subplot(3,1,1)
plot(t,u);
title("Q1 = " + num2str(Q11) + ", Q2 = " + num2str(Q22) + ", R = " + num2str(
ylabel('u')
subplot(3,1,2)
plot(t,x(:,1));
ylabel('p')
subplot(3,1,3)
plot(t,x(:,2));
ylabel('$\dot{p}$','interpreter','latex')


rt = risetime(x(:,1),t)
```

```
%% Lab 03
% Document each day as a section in the script
%% Pendulum System
% Performance Requirements
OS  = 0.05;
t_r = 0.10;
w_n = 1.8 / t_r;
zeta = sqrt(log(OS)^2 / (pi^2 + log(OS)^2));

% Estimates based on experimental identification (Pendulum)
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

% Determine controller gains - based on performance (2nd order approx.)
K_P = w_n^2;
K_D = 2 * zeta * w_n - b;

% Formulate system matrices
A = [0 1; -a -b];
B = [0 c]';
K = place(A,B,P);

% Tune integral gain
```

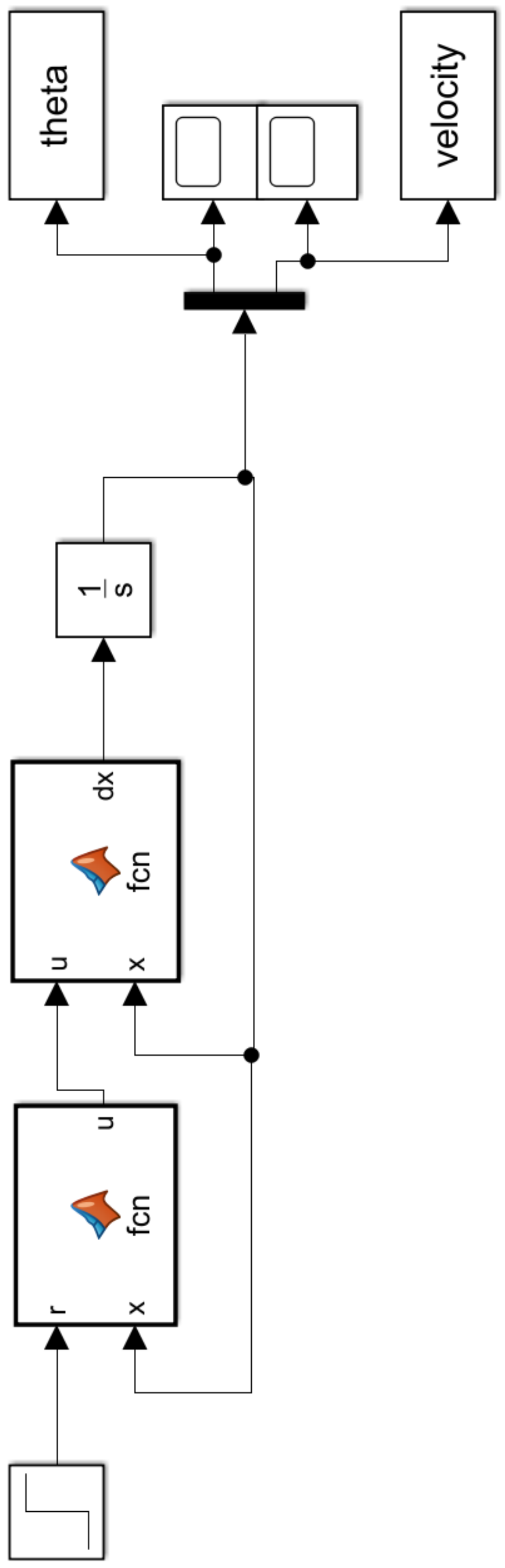
```
K_I = 200/c;

% Run experiment setup
% run initialize_ss.m;

%% 02-25-2020
% Design test with LQR to see how parameters vary performance
Q = eye(2);
R = 1;

[K,S,CLP] = lqr(A,B,Q,R);    % Not explicitly choosing CLP
K_place = place(A,B,CLP);    % Explicit CLP

%% Vary R
close all;
w_r = logspace(-3,3,7);
Q = eye(2);
figure(1)
figure(2)
for i = 1:length(w_r)
     R = w_r(i);
     [K,S,CLP] = lqr(A,B,Q,R);
     sim StepBlock_PID_place
     run saveTheoData.m
     figure(1), hold on
     plot(time,u,'DisplayName',"R = " + num2str(R))
     figure(2), hold on
     plot(time,theta,'DisplayName',"R = " + num2str(R))
end
figure(1)
legend('show')
xlim([0 2])
title('u(t)')

figure(2)
legend('show')
xlim([0 8])
title('y(t)')
%% Vary Q simultaneously
close all;
R = 1;
w_q = logspace(-3,3,7);
figure(1)
figure(2)
for i = 1:length(w_q)
```

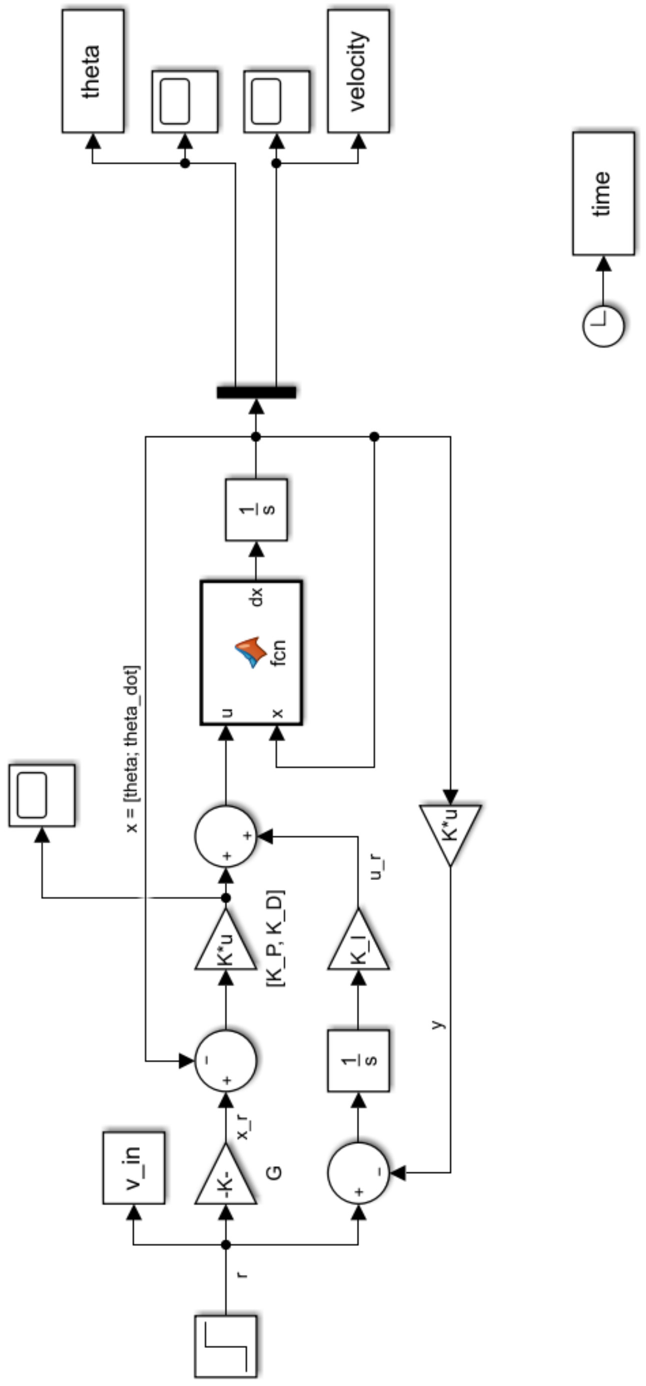
```
    Q = w_q(i)*eye(2);
    disp(Q)
    [K,S,CLP] = lqr(A,B,Q,R);
    sim StepBlock_PID_place
    run saveTheoData.m
    figure(1), hold on
    plot(time,u,'DisplayName',"w_q = " + num2str(w_q(i)))
    figure(2), hold on
    plot(time,theta,'DisplayName',"w_q = " + num2str(w_q(i)))
end
figure(1)
legend('show')
xlim([0 2])
title('u(t)')
figure(2)
legend('show')
xlim([0 8])
title('y(t)')

%% Vary Q individually Q1 - error term

close all;
R = 1;
w_q = logspace(-3,3,7);
figure(1)
figure(2)
for i = 1:length(w_q)
    Q = diag([w_q(i), 1]);
    disp(Q)
    [K,S,CLP] = lqr(A,B,Q,R);
    sim StepBlock_PID_place
    run saveTheoData.m
    figure(1), hold on
    plot(time,u,'DisplayName',"w_q = " + num2str(w_q(i)))
    figure(2), hold on
    plot(time,theta,'DisplayName',"w_q = " + num2str(w_q(i)))
end
figure(1)
legend('show')
xlim([0 2])
title('u(t)')
figure(2)
legend('show')
xlim([0 8])
title('y(t)')
```

```

%% Vary Q individually Q2

close all;
R = 1;
w_q = logspace(-3,3,7);
figure(1)
figure(2)
for i = 1:length(w_q)
     Q = diag([1, w_q(i)]);
     disp(Q)
     [K,S,CLP] = lqr(A,B,Q,R);
     sim StepBlock_PID_place
     run saveTheoData.m
     figure(1), hold on
     plot(time,u,'DisplayName',"w_q = " + num2str(w_q(i)))
     figure(2), hold on
     plot(time,theta,'DisplayName',"w_q = " + num2str(w_q(i)))
end
figure(1)
legend('show')
xlim([0 2])
title('u(t)')
figure(2)
legend('show')
xlim([0 8])
title('y(t)')

%% 02-27-2020
% Implementing LQR
close all;
error_pos = 0.1;
error_vel = 10;
error_u = 1;
Q11 = 1/(error_pos^2);
Q22 = 1/(error_vel^2);
Q = diag([Q11,Q22]);
R = 1/(error_u^2);

[K,S,CLP] = lqr(A,B,Q,R);
sim StepBlock_PID_place
run saveTheoData.m

figure,
subplot(3,1,1)
```

```
plot(time,u);
title("Q1 = " + num2str(Q11) + ", Q2 = " + num2str(Q22) + ", R = " + num2str(
legend('u')
subplot(3,1,2)
plot(time,velocity);
legend('velocity')
subplot(3,1,3)
plot(time,theta);
legend('theta')

rt = risetime(theta,time)

%% LQR Experimental vs theoretical data
close all;
load figures\lqr_settings0.mat

figure,
subplot(3,1,1)
hold on
plot(exp_lqr.time,exp_lqr.theta,'r','LineWidth',1.5);
plot(sim_lqr.time,sim_lqr.theta,'b--','LineWidth',1.5);
legend('exp','sim')
ylabel('theta')

subplot(3,1,2)
hold on
plot(exp_lqr.time,exp_lqr.velocity,'r','LineWidth',1.5);
plot(sim_lqr.time,sim_lqr.velocity,'b--','LineWidth',1.5);
legend('exp','sim')
ylabel('velocity')

subplot(3,1,3)
hold on
plot(exp_lqr.time,exp_lqr.u,'r','LineWidth',1.5);
plot(sim_lqr.time,sim_lqr.u,'b--','LineWidth',1.5);
legend('exp','sim')
ylabel('u')

%% COMPLETE LATER! :)
close all;
w_r = logspace(-3,3,7);
Q = eye(3);
figure(1)
figure(2)
for i = 1:length(w_r)
```

```
    R = w_r(i);
    [K,S,CLP] = lqr(A,B,Q,R);
    sim StepBlock_PID_place
    run saveTheoData.m
    figure(1), hold on
    plot(time,u,'DisplayName',"R = " + num2str(R))
    figure(2), hold on
    plot(time,theta,'DisplayName',"R = " + num2str(R))
end
figure(1)
legend('show')
xlim([0 2])
title('u(t)')

figure(2)
legend('show')
xlim([0 8])
title('y(t)')

%% LQI Theoretical

close all;
error_pos = 0.1;
error_vel = 10;
error_u = 1;
error_e = 0.1;
Q11 = 1/(error_pos^2);
Q22 = 1/(error_vel^2);
Q33 = 1/(error_e^2);
Q = diag([Q11,Q22,Q33]);
R = 1/(error_u^2);

% Define state-space system
C = [1 0];
D = 0;
sys1 = ss(A,B,C,D);

[K,S,CLP] = lqi(sys1,Q,R);
sim StepBlock_PID_lqi_theoretical
run saveTheoData.m

figure,
subplot(3,1,1)
plot(time,u);
title("Q1 = " + num2str(Q11) + ", Q2 = " + num2str(Q22) + ", R = " + num2str(
legend('u')
subplot(3,1,2)
plot(time,velocity);
legend('velocity')
subplot(3,1,3)
plot(time,theta);
legend('theta')

%% LQI Experimental vs theoretical data
close all;
load figures\lqi_settings0.mat

figure,
subplot(3,1,1)
hold on
plot(exp_lqr.time,exp_lqr.theta,'r','LineWidth',1.5);
plot(sim_lqr.time,sim_lqr.theta,'b--','LineWidth',1.5);
legend('exp','sim')
ylabel('theta')

subplot(3,1,2)
hold on
plot(exp_lqr.time,exp_lqr.velocity,'r','LineWidth',1.5);
plot(sim_lqr.time,sim_lqr.velocity,'b--','LineWidth',1.5);
legend('exp','sim')
ylabel('velocity')

subplot(3,1,3)
hold on
plot(exp_lqr.time,exp_lqr.u,'r','LineWidth',1.5);
plot(sim_lqr.time,sim_lqr.u,'b--','LineWidth',1.5);
legend('exp','sim')
ylabel('u')

%% LQI state feedback - reference sine
w = 5;
sim SineBlock_lqi_theoretical

% A = [0 w; -w 0];
% B = [1;0];
% C = [1 0];
% D = 0;
```


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


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