# MTHE 430 Lab Manual

June 30, 2022

# Table of Contents

# Lab 1
# Feedback and PID control

In this lab you will be examining the effects of feedback on system performance. In particular, you will design a control, $R_C$, for the motor using the principles of PID control as shown in Figure 1.1.

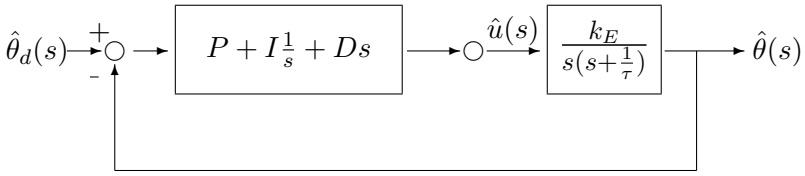


Figure 1.1: PID control system

## 1.1 Key Concepts

In this lab, you will be implementing a PID controller into a closed loop system. The main issue with open loop systems is that we only have control over the reference trajectory, and therefore cannot account for disturbances. Now, if we were to use feedback, we could attempt to control the *error* signal, which is the difference between the reference trajectory (i.e. desired angle, or state of the motor) and the measured output.

A goal of a standard PID controller is to tune the system to behave a certain way by using various constants to correct the error signal. The three contstants of a PID controller are *Proportional*, *Integral*, and *Derivative* controls.

- **Proportional** control acts on the present value of the error signal. This is the most dominant of the three terms, but it can also leave some steady state error.

- **Integral** control accounts for the past values of error which is accumulated over time. This term will eliminate the steady state error that is left behind by the proportional control term, and also reduce rise time.

- **Derivative** control looks at the rate of change in the error signal, and attempts to correct for possible future values of error. For example, if the system is rapidly approaching its reference trajectory, (i.e. the error signal is rapidly approaching zero) the system will be able to slow down and avoid overshoot. Derivative control helps improve the settling time and stability of the system.

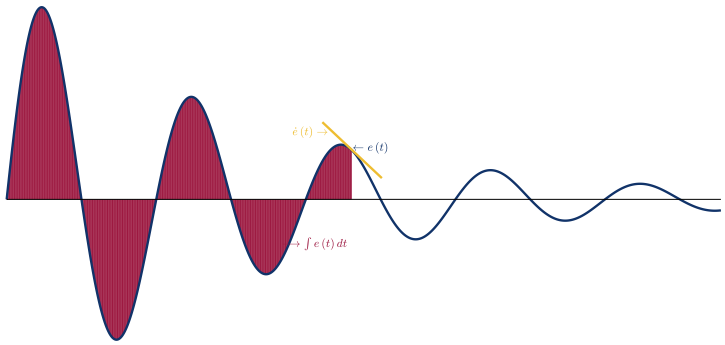


Figure 1.2: Arbitrary error signal with a PID controller acting on it

- Figure 1.2 illustrates the concepts of a PID controller. You have some error signal, where the proportional term acts on the current state of the system, the integral term sums up all previous error, and the derivative term attempts to predict and control future error.

It is important to remember that when using PID controllers, there will always be an element of compromise within your system. For example, the ideal system will have a very low rise time, with little to no steady state error or settling time, and no overshoot. However, this is very unrealistic in the real world. If you were designing a highly precise robotic arm, you may need to tune your controller in a way that there is practically no steady state error, but in order to do so, you will need to have a higher rise time (i.e. the arm will move slowly, but it will go exactly where you need it). On the other hand, you may have a system that requires a quick reaction time, for which you may need to accept that there could be overshoot, or some steady state error.

You will also examine the concept of system types and their relation to PID controllers. Essentially, systems can be type 0, 1, 2, etc ... A systems "type" determines its ability to track the error on a given reference trajectory. For example, a system of type $k$ can track a reference trajectory with a bounded error for polynomials up to degree $k$ (See Proposition 8.11 from the course notes for further clarification). We will see how the different terms from the PID controller transfer function affect the system type.

## 1.2 Background Information

- TODO: Include some stuff from course notes?
- TODO: Include simulink model rather than have them make it in procedure?

- Learn the definitions for rise time, settling time, overshoot, steady-state value and steady state error.
- The closed loop transfer function of the system in 1.1 is given by

$$T(s) = \frac{R_C(s)R_P(s)}{1 + R_C(s)R_P(s)}$$

where $R_C = P + I\frac{1}{s} + Ds$ is the controller and $R_P = \frac{\kappa_E}{s(s+\frac{1}{\tau})}$ is the plant. If using P, I, and D controls individually, the

- Determine the location of the closed loop poles when using P, I, D controls individually. What condition is required on the poles such that the system is BIBO stable?

## 1.3 Procedure

1. Prepare a Simulink model to implement a PID controller as shown in Figure 1.3. *Modify your model from lab 1 or 2.* Figure 1.3. The `PID` block can be found in the

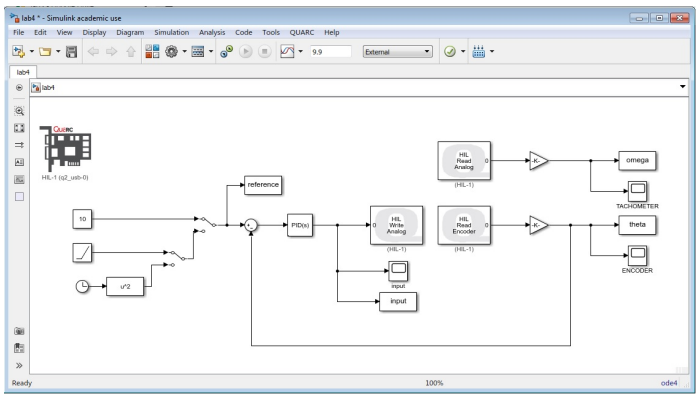


Figure 1.3: Simulink model for a DC Servo Motor system

`Simulink` menu under the `Continuous` section parameters.

As shown in Figure 1.4, the `PID` block contains three parameters: $P$ is the proportional gain, of the controller; $I$ is the integral action parameter which is equal; the $D$ term provides the derivative action.

2. Set the desired angle to 10 radians. The desired input is entered as shown in Figure 1.3 in the `Constant` block. Remember to use the appropriate gain values from Table C.1 for the encoder and tachometer to show results in radians. Do not forget to change the `solver` to `ode 4` in `Configuration Parameters` (`Ctrl+E`).

3. **Proportional Term**

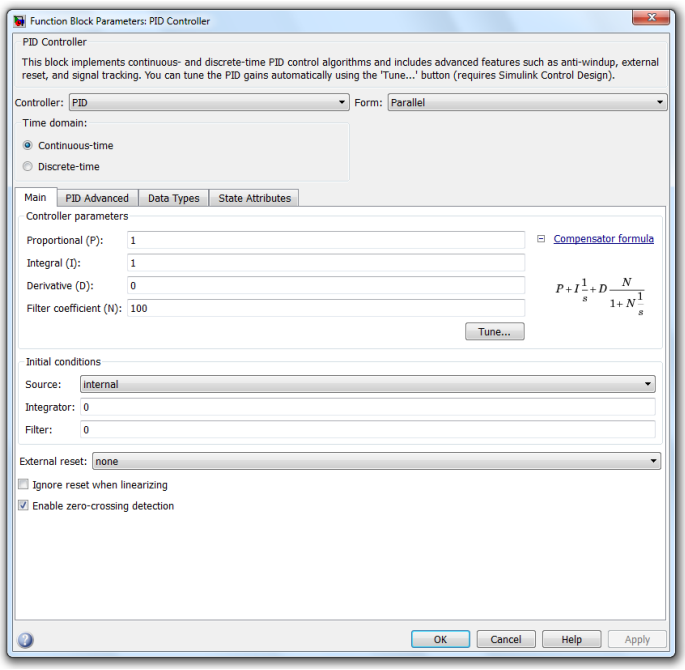


Figure 1.4: Screen shot of the `PID Controller` block

(a) Build and run the Simulink model using only the proportional term (i.e., set $I$ and $D$ to zero). Comment on the effects of using various values of $P$. In particular, comment on the overshoot, rise time, settling time, and the steady-state error for different values of $P$, and be sure to tabulate this data (a table in Excel is an efficient way to do this). Use at least three different values of $P$.

(b) Keep $D$ and $I$ at 0 and increase the value of $P$ so the system oscillates about the desired value of 10 radians without (seemingly) decreasing in magnitude. Print a copy of the encoder output at that value, and a copy when using a slightly lower value of $P$.

(c) Now reset your $P$ value to the "good" value (i.e. not the value that makes the system oscillate about 10 radians, but the value where you get a "desirable" response). Plot the error response of this system (i.e. the difference between input and output).

(d) Repeat the previous step but this time replace the constant input with linear input (i.e. disconnect the "10" block and connect the linear block). Plot the error response.

(e) Repeat the previous step but with the quadratic input (you may have to divide the quadratic term by 2 to prevent the encoder from going out of range).

(f) Based on the above error results, what is the type of the system when using only proportional control?

4. **Derivative Term**

   (a) Test various $D$ values while keeping $P$ constant (at the "good" value you found in the previous section), and comment on overshoot, rise time, settling time, and steady-state error. Again, tabulate this data, and decide on some "good" value of $D$.

   (b) Now, set $P$ to 0 and repeat steps (c)-(e) from 3 using only derivative control.

   (c) Based on the above error results, what is the type of the system when using only derivative control?

5. **Integral Term**

   (a) Set your values of $P$ and $D$ to the values you found above, and test various $I$ values. Comment on the overshoot, rise time, settling time, and steady-state error for these values of $I$, and tabulate the data.

   (b) Now, set $P$ and $D$ to zero and repeat steps (c)-(e) from 3 using only integral control.

   (c) Based on the above error results, what is the type of the system when using only integral control?

6. Now we will consider a more complicated function input. To make things interesting, have the desired angle behave as four different functions on different intervals. This can be achieved by preparing the Simulink model shown in Figure 1.5. *You do not need to use the exact same functions as shown in the figure.* Pick four functions that you think would be interesting. In this model, a number of possible sources have been introduced along with the switching logic. The switching logic is based on the value of the function, `4 sin(0.2*t)^{2}+1`. This function, although arbitrary, assumes a value between 1 and 5 which is associated with ports 1 through 5 of the `Multiport Switch` block. The switching function is entered using the `Fcn` block as indicated in Figure 1.6.

7. Run the system and prepare a plot comparing the angle and the desired angle trajectory, making sure to note the values of the PID coefficients. You may have to tweak the PID values you found in previous sections in order to get a better trajectory.

   When you have completed the lab, make sure you save your files in the folder you created in Lab **??**.

## 1.4 Deliverables

Prepare a brief write up describing what you learned from this lab. This does not need to be a formal report, but all material should be presented in a clear and logical manner, with concise descriptions where necessary. Include the following / answer the following questions:

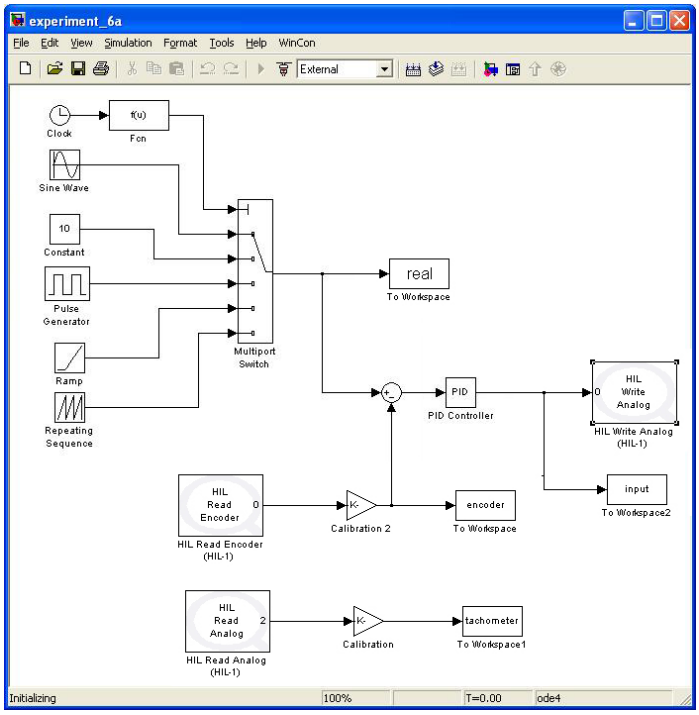


Figure 1.5: Simulink model for the implementation of multiple inputs and PID control

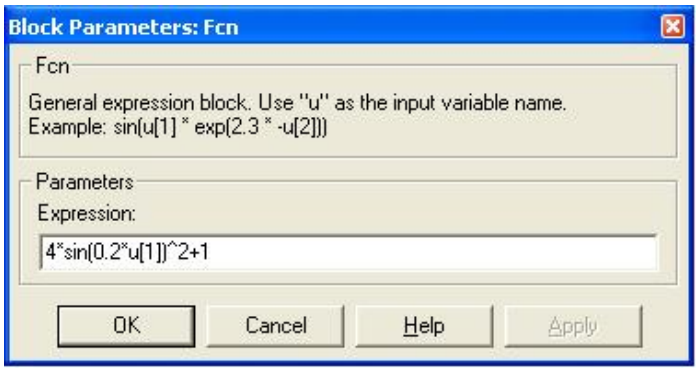


Figure 1.6: Configuration of the `Fcn` block for the implementation of multiple inputs and PID control

1. Include tabulated data of the response characteristics from steps 3, 4, and 5. Be sure that you are using your "good" $P$ value (i.e. NOT the value that makes the system oscillate about 10 rad / s) when collecting data for $I$ and $D$ tests.

2. Comment on how varying $P$, $I$, and $D$ values impact response characteristics (i.e. rise time, settling time, etc...).

3. Include the plot of the output that oscillates consistently about the reference trajectory of 10 radians generated in step 3. Why do higher $P$ values increase the oscillation of the output? What is happening to the location of the closed loop poles in equation (1.2)?

4. Include the plots of the error response when using only proportional, derivative, or integral control. Using your plots, what is the system type in each case? Is this consistent with the BIBO stability criteria in (1.2)?

5. Include a plot of the output using your PID controller when the desired angle is generated by the multi-input switch. Be sure to specify your final P, I, and D values.

# Lab 2
# The Nyquist criterion

In this lab, you will use the Nyquist contour to determine if a system is IBIBO stable. The Nyquist plot is a conformal mapping of a closed loop in complex plane and it is a tool for the understanding of the stability of a closed-looped system. Furthermore, you will be examining the relationship between the Nyquist contour and the Bode plot, thus familiarizing yourself with such terms as *crossover frequency*, *gain margin*, and *phase margin*.

## 2.1 Background Info

This references the Bode plots from 393, should we include Bode plots here?

## 2.2 Procedure

### 2.2.1 Using the Bode plot to sketch the Nyquist contour

For this part of the lab, we will be using the motor in the unity gain feedback configuration shown in Figure 2.1. This setup should be quite familiar to you by now, so you should

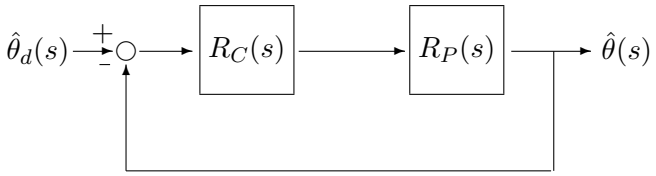


Figure 2.1: Feedback system with disturbance

have a well-developed intuition concerning the IBIBO stability of such a system. We will be taking the output of the system to be the motor velocity, $\omega$, so our plant transfer function is $R_P(s) = \frac{k_E}{s+\frac{1}{\tau}}$. In Lab 2 you constructed the Bode plot for this system experimentally.

1. Using the Bode plot from Lab 2, determine the gain and phase crossover frequencies, and the phase and gain margins at these frequencies; see Section 7.2.2 from the course text.

2. Using the Bode plot determine the general transfer function, from the transfer function sketch the corresponding Nyquist contour. Looking also at the transfer function, determine whether or not the system is IBIBO stable.

3. Start up Matlab and define `t` to be the transfer function of the open-loop motor scheme. Essentially, we are just setting $R_C(s) = 1$ to see how motor behaves on its own.

4. To plot the Nyquist contour, use the `Nyquist` command in Matlab.

5. Is the Nyquist contour consistent with what you know about the IBIBO stability of the system? Comment on both the poles of the transfer function, and the zeroes of the characteristic polynomial.

### 2.2.2 Using a proportional controller

We will now turn our attention to the task of using the Nyquist contour as a tool in determining what range of values of a proportional controller can take on in order to make the system IBIBO stable.

The plant transfer function $R_C(s) = K$ will be used in the feedback configuration depicted in Figure 2.1. The transfer function for this system is simply

$$T(s) = \frac{KR_P(s)}{1 + KR_P(s)}.$$

What is important to note is that the condition $KR_P(s) = -1$ is equivalent to the condition $R_P = -\frac{1}{K}$. This seems obvious in itself, but what this allows you to do is to use the Nyquist contour for $R_P(s)$ on its own (i.e., with $K = 1$), and determine where you would like the point $\frac{1}{K}$ to be. This allows you to get a lot of mileage out of only one plot. The alternative is to construct the Nyquist contour for many values of $K$, and see if each contour has the required number of encirclements of the point $-1 + \mathrm{i}0$. This may not seem like a bad alternative if you have access to a computer, but the method of normalizing the proportional gain is much more efficient.

1. In Matlab, define the plant transfer function to be $R_P = \frac{s+1}{s(\frac{s}{10}-1)}$.

2. Produce both the Bode plot and the Nyquist contour. Take a minute to further reaffirm your faith in the relationship between the two.

3. What are the crossover frequencies, and what are the gain and phase margins at these frequencies?

4. Since our plant transfer function has one pole in the right-half complex plane, we require one counterclockwise encirclements of the point $\frac{1}{K}$. Using this information, in which region should we place the point $-\frac{1}{K}$?

5. What values of $K$ satisfy the above requirement?

6. Repeat the above procedure for the plant transfer function $R_P(s) = \frac{1}{s(s+1)^2}$. Remember, Nyquist plots always close clockwise.

### 2.2.3 The phase margin

In this part of the lab we will quickly demonstrate the phase margin as it applies to the Nyquist contour.

1. We will use the plant transfer function $R_P(s) = \frac{1}{s(s+1)^2}$. Produce the Bode plot and the Nyquist contour of this system.

2. Determine how much you have to boost the gain (in decibels) to provide a phase margin of $0^\circ$ at a gain cross over frequency of $10^n \frac{\text{rad}}{\text{sec}}$, $n \in \mathbb{Z}$. Will the resulting system, with this gain boosted by using a proportional controller, be IBIBO stable?

3. The number you get is a scalar that represents the factor by which you can "stretch" (or "shrink" as the case may be) the Nyquist contour towards the point $-1 + \text{i}0$ and still maintain IBIBO stability (assuming the system is stable for $K = 1$, which our system is). Looking at the Nyquist contour, does this scalar quantity seem to make sense? Is this consistent with your work from Section 2.2.1?

When you have completed the lab, make sure you save your files in the folder you created in Lab **??**.

# Lab 3
# Controllability and observability

This lab will demonstrate the fundamental ideas behind the controllability and observability properties of a system. You will analyze a simple circuit and determine the conditions for observability and controllability. You will then proceed to simulate the system using Simulink under various conditions.

## 3.1 Prelab

You may wish to read Sections 2.3.1 and 2.3.2 from the course notes to recall some background on controllability and observability.

Using Figure 3.1 as a reference, define $x_1$ as the voltage across the capacitor, and $x_2$ as

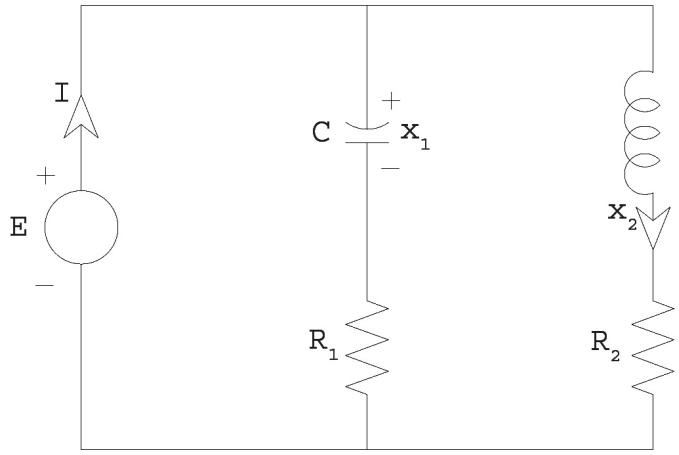


Figure 3.1: State space configuration

the current through the inductor. Take the output to be the current entering the circuit, denoted $I$ in Figure 3.1.

1. Write out the system equations in the state-space form:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}u,$$
$$y = \boldsymbol{c}^t + \boldsymbol{D}u.$$

   If you are having trouble getting the differential equations, or just not "electrically inclined", you should review past course notes on electrical circuits and differential equations. Suck it up, Apple Mech students!

- When finding the differential equations for a system, your goal should be to determine an expression for the derivative of each state variable in terms of the state variable(s) and the forcing function. Once you have these, you can determine $\boldsymbol{A}$ and $\boldsymbol{b}$.
- The following facts may be useful. The current through a capacitor is given by $I_c = C\frac{dV_C}{dt}$, the voltage across an inductor is given by $V_L = L\frac{dI_L}{dt}$, and the voltage across a resistor is given by Ohm's law, $V_R = I_R R$. By Kirchoff's voltage law, the voltage across each branch of the circuit is simply $E$. You can use this fact to get expressions for $\boldsymbol{A}$ and $\boldsymbol{b}$.
- Since the output is the current entering the circuit, and you will by this point have expressions available for the current in each branch, you can use Kirchoff's current law (i.e., conservation of current) to determine expressions for $\boldsymbol{c}$ and $\boldsymbol{D}$.

2. Calculate the controllability matrix $\boldsymbol{C}(\boldsymbol{A}, \boldsymbol{b})$.
3. Calculate the observability matrix $\boldsymbol{O}(\boldsymbol{A}, \boldsymbol{c})$.
4. Determine the conditions under which the system is uncontrollable. Recall that a square matrix has full rank if and only if its determinant is non-zero.
5. Determine the conditions under which the system is unobservable.
6. When the system is uncontrollable, determine the set of reachable points for zero initial conditions. This is going to be a one-dimensional vector space, so there is a simple relationship between $x_1$ and $x_2$. Determine this relationship when the system is uncontrollable.
7. When the system is unobservable, determine the set of initial conditions that yield the same output, and the linear relationship between the initial conditions.

## 3.2 Procedure

Via simulation, you will determine the conditions under which the circuit in Figure 3.1 is controllable and/or observable.

### 3.2.1 Controllability

When analyzing the controllability, you will be examining the behaviour of the system states. Recall that a rough definition of controllability is: "starting from the origin, you can reach any point in state space by applying an appropriate input."

1. Build the Simulink model as shown in Figure 3.2. The model applies a constant voltage to the dynamic model defined by $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{D}$ defined previously. The output is the current in the circuit.

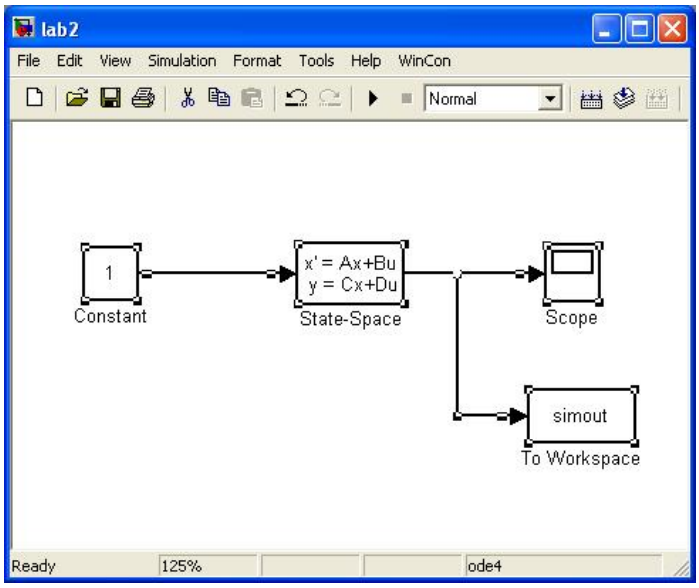


Figure 3.2: Simulink model for Lab 3

2. Define the matrices $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{D}$ by double-clicking on the `State-space` block. Each matrix is entered using the following convention. The following is an example on how to properly input values into the `State-Space` block. The values shown are *not* the proper values. Use values that correspond to the matrices in the prelab. The matrix

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -1 & 2 \end{bmatrix}$$

is entered by typing `[0,1;-1,-2]` in the line reserved for $\boldsymbol{A}$. Note that elements of a row are delimited by comma (or spaces) and each row is delimited by a semicolon.

The entries of Figure 3.3 correspond to the dynamical system

$$\begin{aligned} \dot{\boldsymbol{x}} &= \begin{bmatrix} 0 & 1 \\ -1 & -2 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u, \\ y &= \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x} + [0] u. \end{aligned}$$

Note that the last line specifies the initial conditions for the states. In this case, we have set $x_1(0) = 0$ and $x_2(0) = 0$. Come up with an appropriate variable to show the linear relationship between $x_1$ and $x_2$ when the system is uncontrollable (make sure you record this in your lab report). Define this variable as one of your outputs in your model. Note that you can add as many outputs as you want just by adding rows to the matrix $\boldsymbol{c}$.

Define the initial conditions to be zero. The default final time is set to 10 by Simulink. You may change it by opening the window

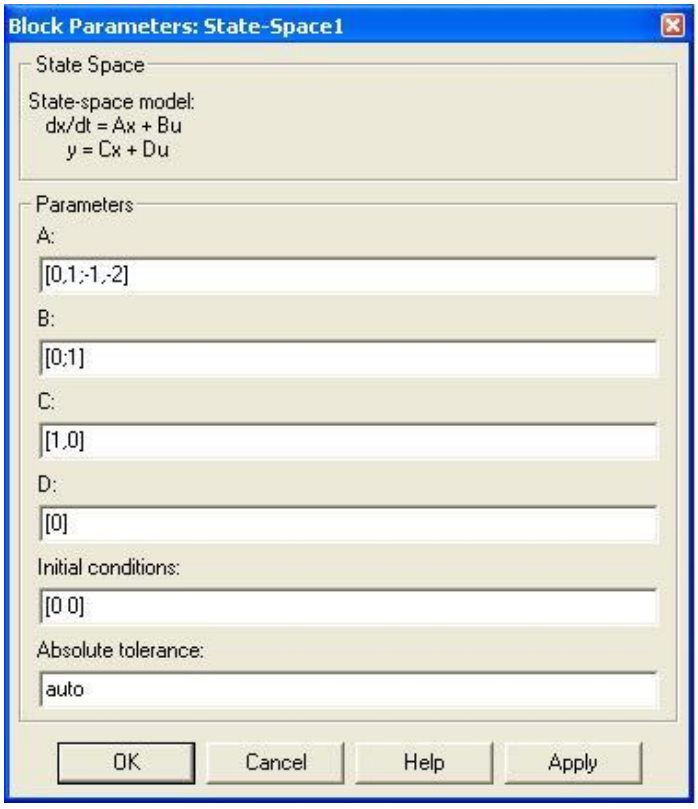


Figure 3.3: State space configuration

`Simulation→Configuration Parameters`

and entering the required time in the `Stop Time` box.

3. Run the Simulink block under uncontrollable conditions. This will depend on your choice of $R_1$, $R_2$, $C$, and $L$. Recuperate the state variable values written to the workspace and plot $x_1$, $x_2$, and the output variable you defined. Does the output variable you chose show that there is, in fact, a linear relationship between $x_1$ and $x_2$? Under these conditions, is it possible to find an input voltage $E$ that will allow you to reach a point that is off this line?

4. Add an appropriate title to your graph and print it.

5. Rerun the system, but this time use conditions that make the system controllable. Plot and print a graph of $x_1$ and $x_2$ and your output variable. What can you now say about your output variable? What is the set of reachable points under these conditions?

6. Again, give an appropriate title to your graph and print it.

### 3.2.2 Observability

When analyzing observability, you will be examining the output behaviour of the system. Recall that a rough definition of observability is: "a change in initial conditions and/or input results in a change in the output."

1. Using the work from your prelab, enter the value of $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{D}$ into the `State-Space` block. Change the name of the output variable from `simout` to `I`.

2. We will want to examine the output using many different initial conditions. Build and run the system using a pair of initial conditions that are in the kernel of the observability matrix. Plot, and print a graph of the output variable $I$. Make sure that you include values of the constants in the title of the plot.

3. Rerun the system using a different pair of initial conditions that are also in the kernel of the observability matrix. Include a plot of the output. Does the result of this experiment confirm your work from the prelab? What happens if you use initial conditions that are not in the kernel of the observability matrix?

When you have completed the lab, make sure you move the files created in the directory created in Lab **??**.

# Lab 4

# Stability, Feedback Control & Luenberger Observer for Balancing an Inverted Rotary Pendulum

## 4.1 Introduction

The purpose of this lab is to study another equilibrium of the rotary pendulum system and to develop a control technique that balances the pendulum around this equilibrium using partial state feedback and a linear observer. The prelab focuses on the linearization of the EOMs for the rotary pendulum about the equilibrium point where the pendulum is completely inverted, and also focuses on analyzing the stability of the system at this equilibrium point. In the lab, you will first use full state feedback and pole placement techniques to develop a controller that balances the inverted pendulum. Then, you will carry out the same task for the case where you do not have access to the full state information. To do this, you will develop a Luenberger observer to estimate the states that you do not have access to and you will use the estimated states to develop a controller that balances the inverted pendulum.

Figure 4.1: Qanser SRV02 with pendulum module, shown in an inverted orientation [**?**].

## 4.2 Background Information

### 4.2.1 Linearization

We need the linearized versions of the EOMs of the rotary pendulum from Section **??** about the equilibrium point $\theta_0 = 0$ and $\alpha_0 = 0$. Recall that to linearize a multivariate function $f$ of variables $z^T = [\theta\ \alpha\ \dot{\theta}\ \dot{\alpha}\ \ddot{\theta}\ \ddot{\alpha}]$ around an equilibrium point $z_0^T = [\theta_0\ \alpha_0\ \dot{\theta}_0\ \dot{\alpha}_0\ \ddot{\theta}_0\ \ddot{\alpha}_0]$, we use

$$f_{\text{lin}} = f(z_0) + \left(\frac{\partial f(z)}{\partial \theta}\right)\Big|_{z_0}(\theta - \theta_0) + \left(\frac{\partial f(z)}{\partial \alpha}\right)\Big|_{z_0}(\alpha - \alpha_0) + \cdots + \left(\frac{\partial f(z)}{\partial \ddot{\alpha}}\right)\Big|_{z_0}(\ddot{\alpha} - \ddot{\alpha}_0)$$

First, for the generalized coordinate $\theta$, we compute

$$\left(m_p L_r^2 + J_r\right)\ddot{\theta} - \frac{1}{2}m_p L_p L_r \ddot{\alpha} = \tau - B_r\dot{\theta}$$

And for the generalized coordinate $\alpha$, we compute

$$-\frac{1}{2}m_p L_p L_r \ddot{\theta} + \left(J_p + \frac{1}{4}m_p L_p^2\right)\ddot{\alpha} - \frac{1}{2}m_p L_p g\alpha = -B_p\dot{\alpha}$$

### 4.2.2 State-Space Representation

In order to find the state space representation, we first write the linearized EOMs in matrix form, i.e.,

$$\begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix}\begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} + \begin{bmatrix} f_{11} & f_{12} \\ f_{21} & f_{22} \end{bmatrix}\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + \begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix},$$

and group all non double-derivative terms to the right. We can now explicitly solve for $\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix}$ and put it in a relatively compact form. Letting our state be

$$\mathbf{x}(t) = \begin{bmatrix} \theta(t) \\ \alpha(t) \\ \dot{\theta}(t) \\ \dot{\alpha}(t) \end{bmatrix}$$

we obtain

$$\begin{bmatrix} \dot{\theta}(t) \\ \dot{\alpha}(t) \\ \ddot{\theta}(t) \\ \ddot{\alpha}(t) \end{bmatrix} = \frac{1}{J_T}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1}{4}m_p^2 L_p^2 L_r g & -\left(J_p + \frac{1}{4}m_p L_p^2\right)B_r & -\frac{1}{2}m_p L_p L_r B_p \\ 0 & \frac{1}{2}m_p L_p g\left(J_r + m_p L_r^2\right) & -\frac{1}{2}m_p L_p L_r B_r & -\left(J_r + m_p L_r^2\right)B_p \end{bmatrix}\begin{bmatrix} \theta(t) \\ \alpha(t) \\ \dot{\theta}(t) \\ \dot{\alpha}(t) \end{bmatrix}$$
$$+ \frac{1}{J_T}\begin{bmatrix} 0 \\ 0 \\ J_p + \frac{1}{4}m_p L_p^2 \\ \frac{1}{2}m_p L_p L_r \end{bmatrix}\tau$$

Evaluating the symbolic state-space matrices $A$ and $B$ using the following values:

$$\begin{cases} J_p = 0.001199\ kg \cdot m^2 \\ J_r = 0.000998\ kg \cdot m^2 \\ B_p = 0.0024\ \frac{N \cdot s}{m} \\ B_r = 0.0024\ \frac{N \cdot s}{m} \\ L_p = 0.3365\ m \\ L_r = 0.2159\ m \\ m_p = 0.1270\ kg \\ m_r = 0.2570\ m. \end{cases}$$

gives

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 81.38 & -46.05 & -0.93 \\ 0 & 122.03 & -44.31 & -1.39 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 83.64 \\ 80.48 \end{bmatrix},$$

### 4.2.3 Stability Analysis

Given that the physical system's sensors are limited to reading the rotor angle and the deflection angle, our state space matrices $C$ and $D$ are

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

and

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

The open loop poles can be found by computing the roots of the system's characteristic equation, i.e., finding the roots of $\det(sI - A) = 0$. In MATLAB, this is easily done by using the command $eig(A)$. The open-loop poles are

$$\{-48.63,\ 7.05,\ -5.86,\ 0\}.$$

It can be concluded that the system is internally unstable about the chosen equilibrium as $\text{spec}(A) \cap \mathbb{C}^+ \neq \emptyset$. This makes physical sense as the inverted pendulum is not in a stable orientation and any perturbation about this equilibrium point will drive the pendulum away from the inverted orientation.

TODO: will student have done this? Recall that in Lab 2 you linearized the rotary pendulum EOMs about another equilibrium. Was the linearized system stable at this equilibrium (and if so, what type of stability is ensured)? Does this make sense from a physical point of view?

## 4.3 Pole Placement & Full State Feedback

*Preliminary question:*

Q1: Can we hope to use full state feedback to steer the linearized rotary pendulum system, linearized about $\alpha = 0$ (the unstable equilibrium)?

*Control technique:*

In this section, you will be designing a feedback controller that stabilizes the system about the inverted pendulum orientation. That is, to balance the pendulum in the inverted orientation, you must design a feedback gain that renders the closed-loop system *internally asymptotically stable*. You saw in Lab 2 that when linearizing the EOMs of the rotary pendulum about the stable equilibrium (i.e., the pendulum oriented downwards), there is a domain of validity for your linearization. The same applies for the linearization that you performed in this prelab, so when designing the balance controller, you will need to lift up the pendulum from the downwards orientation to the inverted orientation while holding the servo base, and then have the controller engage in a small domain centered at the unstable equilibrium. Thus, the controller will be of the form

$$u(t) = \begin{cases} K(\mathbf{x_d} - \mathbf{x}(t)) & |\alpha| < \epsilon; \\ 0 & \text{otherwise}, \end{cases}$$

where $\epsilon$ is the angle designating when the controller will engage and $x_d$ is the desired reference state. To balance the pendulum in the inverted orientation, one should choose a desired reference state of

$$\mathbf{x_d} = \begin{bmatrix} \theta_d \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $\theta_d$ is a desired rotor base angle (one may wish that the rotor base track a desired trajectory, thus $\theta_d = \theta_d(t)$). To design an appropriate control gain that will balance the inverted pendulum, consider our closed-loop system when the controller is engaged:

$$\begin{cases} \dot{\mathbf{x}}(t) = (A - KB)\,\mathbf{x}(t) + B\mathbf{x_d}; \\ \mathbf{y}(t) = C\mathbf{x}(t). \end{cases} \tag{4.1}$$

*Let us review some theory from class:*

Recall that for $A \in \mathcal{M}^{n\times n}(\mathbb{R})$ and for $B \in \mathbb{R}^n$, if the pair $(A, B)$ is controllable, then there exists a transformation $T \in \mathcal{M}^{n\times n}(\mathbb{R})$ where $\det(T) \neq 0$ and

$$\tilde{A} = TAT^{-1} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-1} \end{bmatrix}$$

where $\chi_A = \chi_{\tilde{A}} = s^n + a_{n-1}s^{n-1} + \cdots + a_1 + a_0$ (i.e., $\tilde{A}$ is uniquely determined by the coefficients of the characteristic polynomial of $A$), and

$$\tilde{B} = TB = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix},$$

which is called the canonical controllable form (recall that $(A, B)$ is controllable if and only if $(\tilde{A}, \tilde{B})$ is controllable since system controllability is invariant under similarity transformations). To find an appropriate feedback gain $K$ for your system (4.1) that will place the system's poles in $\mathbb{C}^-$, you will first design a feedback gain $\tilde{K}$ for the transformed system and then transform $\tilde{K}$ into $K$. To do that, design $\tilde{K} = [k_0,\ k_1,\ \dots,\ k_{n-1}]$ so that you obtain the desired poles:

$$\tilde{A} + \tilde{B}\tilde{K} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \\ (k_0 - a_0) & (k_1 - a_1) & (k_2 - a_2) & \dots & (k_{n-1} - a_{n-1}) \end{bmatrix}$$

which yields $\chi_{(\tilde{A}+\tilde{B}\tilde{K})} = s^n + (a_{n-1} - k_{n-1})s^{n-1} + \cdots + (a_1 - k_1) + (a_0 - k_0)$. In class, you found an explicit formula for computing the transformation $T$:

$$T = \mathcal{C}_{(\tilde{A},\tilde{B})}\mathcal{C}^{-1}_{(A,B)}.$$

*Back to your lab:*

You can use these results to design the appropriate feedback gain $K$ to balance the inverted pendulum.

Q2: Find the coefficients of the characteristic polynomial for your open-loop state-space model by multiplying the eigenvalues together, as shown in the following:

$$\chi_A = (s - \lambda_1)(s - \lambda_2)(s - \lambda_3)(s - \lambda_4) = s^4 + a_3s^3 + a_2s^2 + a_1s^1 + a_0.$$

What is $\tilde{A}$?

Q3: Find a $\tilde{K}$ such that the poles of the closed-loop system are $\{-2.8 + 2.86i,\ -2.8 - 2.86i,\ -30,\ -40\}$.
**Hint:** You can use the *sym2poly(p)* MATLAB command to find the coefficients of the characteristic polynomial generated by the desired poles.

Q4: Calculate the transformation $T = \mathcal{C}_{(\tilde{A},\tilde{B})}\mathcal{C}^{-1}_{(A,B)}$. A quick way to find $\mathcal{C}_{(A,B)}$ is to use the *ctrb(A,B)* MATLAB command. What is the ensuing feedback gain, $K$?
**Hint:** $A - BK = T^{-1}(TAT^{-1} + TBKT^{-1})T \implies A - BK = T^{-1}(\tilde{A} + \tilde{B}\tilde{K})T$, thus $\tilde{K} = KT^{-1}$.

You will now use this feedback gain to experimentally balance the physical rotary pendulum system. Open **balancing_model.mdl** and make sure that your state-space matrices $A$, $B$, $C$ and $D$ as well as your feedback gain $K$ is loaded in the MATLAB workspace. The Simulink model is shown in Figure 4.2. You may need to connect the physical system to the computer, as was done in Section **??** of Lab 2. The feedback controller is activated by the Control Switch block when the pendulum angle is within an $\epsilon$-range of $\alpha = 0$ (recall that $\alpha = 0$ corresponds to the pendulum being completely inverted). While the system is balancing the inverted pendulum, you will input to the system a square wave of amplitude 10, so that the servo motor (and hence $\theta$) tracks the square wave. This is done by changing the Amplitude block to 10, but only **after** the pendulum has started balancing. Note that you will have to hold the servo motor when initially lifting the pendulum to the inverted orientation, and then let it go once the pendulum is completely inverted. It may take a few tries to get the pendulum to balance.

The curious reader may concerned with the workings of the Find State X subsystem block found in the Simulink model shown in Figure 4.2. This subsystem block uses a numerical differentiator technique to find the derivative estimations of the measured signals, $\theta$ and $\alpha$. This technique is based on using a high-gain observer [**?**] to estimate the derivatives of the signals. While the angular velocity of the rotor base and the pendulum are not measured and outputted from the physical system (neither are they outputted from the state-space model), one can usually estimate these variables by using such methods.

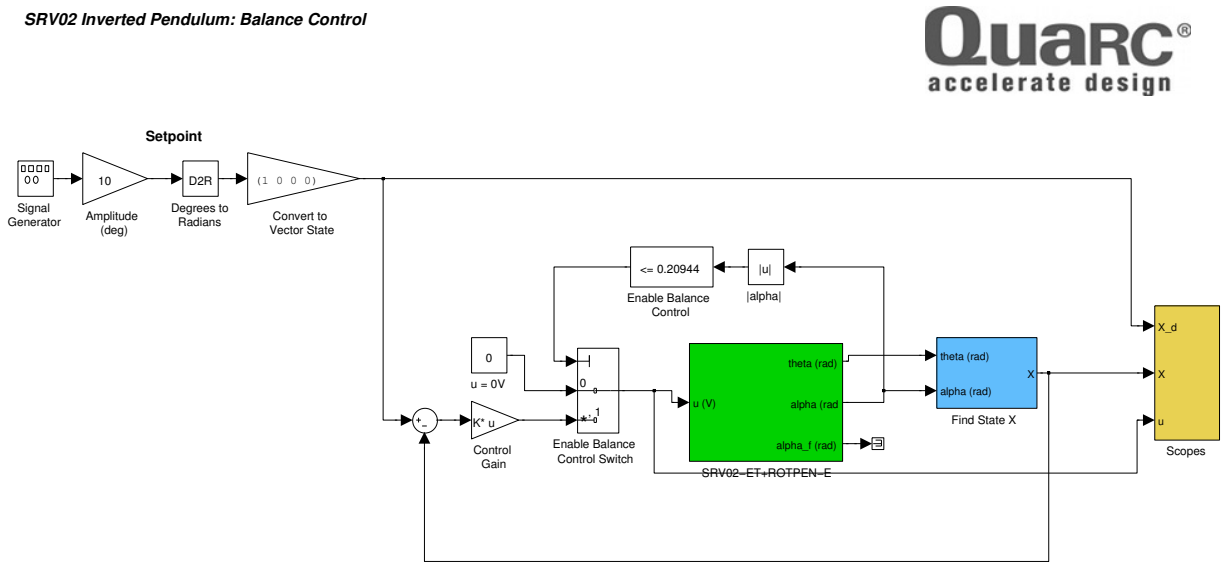


Figure 4.2: A Simulink model which interacts with the physical rotary pendulum system and uses state feedback in a small domain about the unstable equilibrium (i.e., inverted pendulum orientation) to balance the inverted pendulum. [**?**]

Q5: Record the results of the balancing experiment and plot your data. On one plot, compare the square wave input for the rotor base and the measured rotor base angle, $\theta$. On another plot, include the pendulum angle data, $\alpha$. Does your feedback controller manage to balance the inverted pendulum while tracking the square wave trajectory? What happens if you keep increasing the square wave amplitude, and why does this happen? What happens when you change the poles of the closed-loop system to be more negative in the left half-plane of $\mathbb{C}$ (i.e., designing the feedback gain, $K$, such that the closed-loop poles are much more negative)? For balancing the inverted pendulum, discuss the *trade-offs between designing* the control system such that the poles of the closed-loop system are close to the origin or far in the left half-plane.

## 4.4 Linear Observer

Recall our linear time-invariant system dynamics

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t) \qquad \mathbf{y}(t) = C\mathbf{x}(t)$$

and consider the case where the encoder that reads the pendulum angle is damaged, i.e.,

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

In this case, you do not have access to one of the system's internal states, $\alpha$. However, you may be able to determine the unknown state by using the system's input and output data.

To do this, you will construct a linear dynamical system that produces an estimate of the physical system's state, also known as a *linear state observer*.
*Let us review some theory from class:*

Let us denote the state estimate by $\hat{\mathbf{x}}(t)$. David G. Luenberger proposed the following linear state observer [**?**]:

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}}(t) + Bu(t) + L(\mathbf{y}(t) - C\hat{\mathbf{x}}(t)), \tag{4.2}$$

where $L$ is the observer gain. Note that the input and measured output data are both used in (4.2). The observer in (4.2) is composed of two parts: the first part, $A\hat{\mathbf{x}}(t)+Bu(t)$, is the rate of change of $\hat{\mathbf{x}}(t)$ as computed by the physical system's state-space model; the second part, $L(\mathbf{y}(t) - C\hat{\mathbf{x}}(t))$, is the difference between the observed output and the estimated output, scaled by the observer gain. Notice that without the second term, (4.2) becomes familiar:

$$\dot{\hat{\mathbf{x}}}(t) = A\hat{\mathbf{x}}(t) + Bu(t). \tag{4.3}$$

Let us denote the estimation error by $\tilde{\mathbf{x}}(t)$ and define it by $\tilde{\mathbf{x}}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$. If one were to study the estimation error of the simplified observer (4.3), then they would conclude that $\tilde{\mathbf{x}}(t)$ would go to zero when $\text{spec}(A) \subset \mathbb{C}^-$, as the error dynamic for (4.3) is

$$\begin{aligned}\dot{\tilde{\mathbf{x}}}(t) &= A\left(\mathbf{x}(t) - \hat{\mathbf{x}}(t)\right) + Bu(t) - Bu(t)\\ &= A\tilde{\mathbf{x}}(t).\end{aligned}$$

That is, when matrix $A$ has all of its eigenvalues in the left half-plane, then the state estimate would converge to the actual state. But this convergence depends on the physical system's dynamics (thus the convergence rate can be quite slow, depending on the system) and the spectral constraints on $A$ are very restrictive. In contrast, the error dynamic for the Luenberger observer in (4.2) is

$$\dot{\tilde{\mathbf{x}}}(t) = (A - LC)\tilde{\mathbf{x}}(t). \tag{4.4}$$

Thus one can design an observer gain $L$ such that $\text{spec}(A-LC) \subset \mathbb{C}^-$ (i.e., an *asymptotically stable observer*) and one can tune $L$ such that $\hat{\mathbf{x}}(t)$ converges to $\mathbf{x}(t)$ fast enough for the desired application. An important question to ask is, "when does such an observer exist"? In class, you proved that one can build an observer with these properties when $(C, A)$ is *detectable*.
*Back to your lab:*

**Throughout this section, we'll be studying the rotary pendulum linearized about its stable equilibrium (i.e., the system from Lab 2). You will need your MATLAB files from Lab 2.**

Q6: From your previous assessment on the controllablility & observability of the rotary pendulum system when its pendulum encoder is damaged (Lab 2), can you hope to observe the unknown state $\alpha$ by designing and building an appropriate state observer? What does system detectability mean?

One way to *design an observer is by eigenvalue assignment*: the Luenberger observer error, $\tilde{\mathbf{x}}(t)$, governed by the differential equation (4.4), has a characteristic polynomial chosen by the designer. That is, the observer designer chooses $L$ such that the eigenvalues of $(A-LC)$ are at desired locations $p_1, p_2, \ldots, p_n$. This design method has some benefits over other methods, mainly:

1. theoretically, the observer error can decrease faster if the eigenvalues of $(A-LC)$ are more negative (however, there is a trade-off which you will explore), and;
2. if the observed states are being used for state feedback, then the least negative eigenvalue of $(A-LC)$ should be more negative than the eigenvalues of the state feedback system $A-BK$. Loosely speaking, this dictates that the observer dynamics will converge to the unobserved states *faster* than the system dynamics will change these states.

One may notice a similarity between finding a feedback gain $K$ such that $A-BK$ has eigenvalues in $\mathbb{C}^-$ and finding an observer gain $L$ such that $A-LC$ also has eigenvalues in $\mathbb{C}^-$. Recall that the eigenvalues of a matrix and its transpose are the same. The problem of observer design is in fact the *dual* of the state feedback problem, with the following equivalences [**?**]:

$$\begin{aligned}(A-LC)^T &= A^T - C^T L^T \\ &\leftrightarrow A - BK\end{aligned}$$

$$A \leftrightarrow A^T \quad B \leftrightarrow C^T \quad K \leftrightarrow L^T.$$

In Section 4.3, you solved for $K$ by using the canonical controllable form and a similarity transformation. In MATLAB, there's a shortcut to solving for $K$ which uses the *place* command. To solve for the feedback gain with desired poles at $\{-2.8+2.86i,\ -2.8-2.86i,\ -30,\ -40\}$, you can do the following in MATLAB:

$$p = [-2.8+2.86i \quad -2.8-2.86i \quad -30 \quad -40];$$

$$K = place(A, B, p);$$

Q7: In MATLAB, how could you use the *place* command to design observer gain $L$ such that $A-LC$ has its poles at $p = \{p_1,\ p_2,\ p_3,\ p_4\}$? **Hint:** duality.

### 4.4.1 Designing & Validating an Observer

You will now design an observer for the linearized rotary pendulum system, linearized about the *stable equilibrium*. For this section of the lab, you'll need the state-space matrices $A$, $B$, $C$ and $D$ that you calculated in Section **??** of Lab 2. You should alter your $C$ matrix so that only the rotor base angle, $\theta$, is outputted, as you are studying the case where the encoder reading the pendulum angle is damaged. Figure 4.3 shows a Simulink block diagram for a Luenberger observer; you will integrate this observer subsystem into a Simulink model to compare the actual and estimated states.

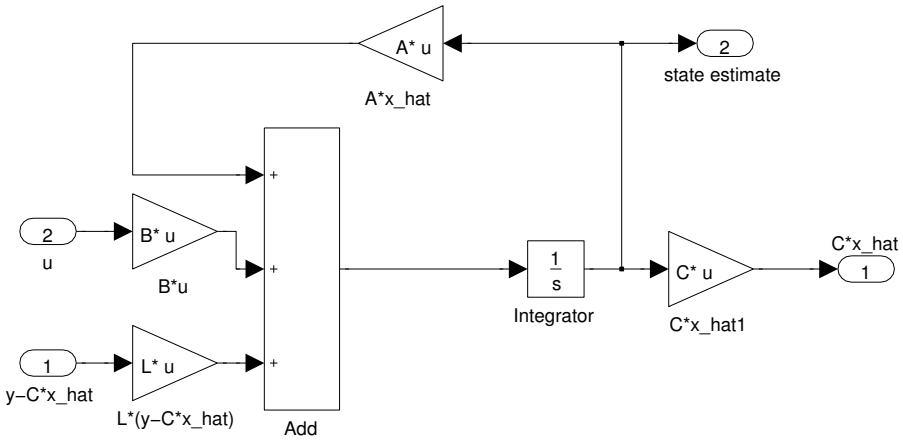


Figure 4.3: A Simulink block diagram of a Luenberger observer. This is a subsystem block that can be incorporated into a Simulink model, along with a state-space subsystem block, to compare the estimated and the actual states.

Open the Simulink model **luenberger_openloop_model.mdl**. Note that the regular State-Space block is not used to model the linearized rotary pendulum system; in place, we've created a subsystem block that does this, and it is shown in Figure 4.4 along with the Luenberger observer subsystem block.

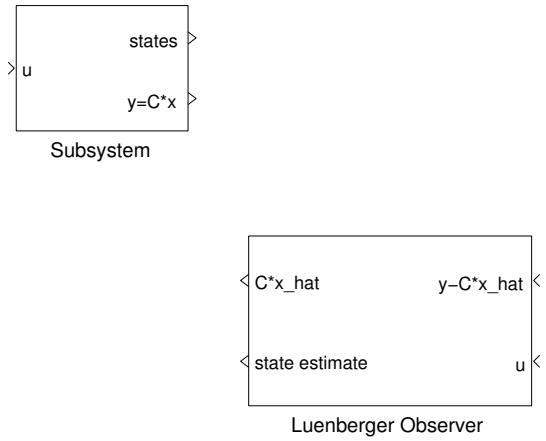


Figure 4.4: A Simulink block diagram of a state-space subsystem & Luenberger observer subsystem. The state-space subsystem block allows one to output the usual $y = Cx$ output as well as outputting all of the states at each time, making it useful for state estimate validations.

Connect these two Simulink subsystem blocks so that the observer can estimate the states of the linearized state-space model of the rotary pendulum. Now, add a Signal

Generator block, a Sum or Add block, and a DegreesToRadians block to the Simulink model and use a sine signal with amplitude 10 as your input (transforming your input to radians using the DegreesToRadians block). In the state-space subsystem, set the initial conditions (found in the integrator block) of your states to zero (in MATLAB, this is done with the vector $[0; 0; 0; 0]$), and set the initial conditions of your observer subsystem to non-zero values between $[-10, 10]$. Make sure to output your states and your state estimates to your workspace using To Workspace blocks.

Q8: Run various tests using this model while changing the observer's poles in $\mathbb{C}^-$. First, *design and tune your observer* so that the state estimates converges to the actual states very quickly; second, try to achieve state estimates that don't deviate from the actual states too much. *What metrics you would use to evaluate* your observer (and the resulting observer gain design) were it to be used for: state feedback to balance the inverted pendulum, and; monitoring an industrial process where large spikes cause the industrial plant to shut down.

### 4.4.2 Testing your Observer on the Physical System

In this section, you will verify that the observer you built for the state-space model works for the physical system. This will help you identify the benefits and drawbacks of different observer tunings for gain $L$. Open the Simulink model **luenberger_openloop_quanser.mdl**. When accessing the rotary pendulum subsystem block, one will notice that the Luenberger observer is nested into this subsystem, as shown in Figure 4.5. This subsystem outputs the measured and the estimated pendulum angle, so that these angles can be compared directly.

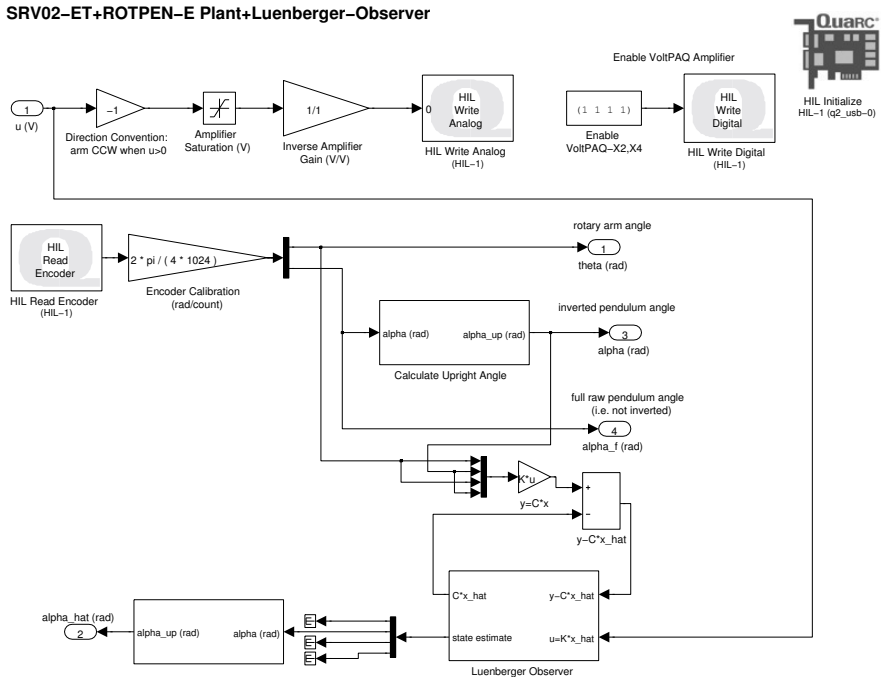


Figure 4.5: A Simulink model of a Luenberger observer estimating & outputting the pendulum angle, $\hat{\alpha}$, for the physical rotary pendulum system. Note that in the Luenberger observer, measured states $\dot{\theta}$ and $\dot{\alpha}$ are non needed, as they are scaled by $C$.

Q9: Build & run **luenberger_openloop_quanser.mdl**, with a sine wave input with amplitude of 1 and frequency of 5 $rad/s$. First, use $[0,\ 0,\ 0,\ 0]$ as the initial conditions for the observer. Plot the measured and the estimated pendulum angle, $\alpha$ and $\hat{\alpha}$, with respect to time when your observer has poles $\{-5,\ -6,\ -7,\ -8\}$. Repeat the experiment for observer poles $\{-15,\ -16,\ -17,\ -18\}$, and then for observer poles $\{-45,\ -46,\ -47,\ -48\}$ (creating the same plots for each experiment). Repeat these for the case where the initial conditions of the observer are not identically zero. From these plots, what can you conclude about the relationship between the accuracy of the state estimates produced by your Luenberger observer and *the design and tuning* of the observer gain, $L$? Give two different real-life applications where state observers could be used, and what *metrics* would you develop to evaluate these observers in the setting of their respective applications (specifically look at the relationship between observer behaviour and pole placement via observer gain, $L$).

## 4.5 Deliverables

To recap the lab objectives, this is what you need to include in your lab report to obtain full marks:

1. the answers to questions Q1, Q4, Q5, Q8 and Q9 (include plots where requested).

**Note: Save your lab report and have an electronic copy on-hand in all future labs, as you will need some of the values obtained in this lab to build upon this physical system.**

# Lab 5

# Lab 4: LQR Optimization for Feedback Control and Observer Design for the Rotary Flexible Beam

## 5.1 Introduction

The purpose of this lab is to design and optimize a feedback controller for the rotary flexible beam system. One way of achieving this is to use a Linear Quadratic Regulator, which automates the process of finding a state feedback gain $K$ in a way that optimizes the system for certain design parameters. The prelab focuses on analyzing the stability of the open-loop rotary flexible beam system. In the lab, you will first design and optimize the closed-loop state feedback system by using the LQR technique to minimize the flexible beam's deflections while the rotor base tracks an input signal trajectory. The closed-loop system will have to adhere to set restrictions on settling time, percent overshoot and maximal flexible beam deflection. Next, you will observe the effects of using only partial state feedback on the system response to an input. Finally, you will attempt to minimize these system response effects by building an observer for the unobservable state and using the ensuing state estimate as feedback.

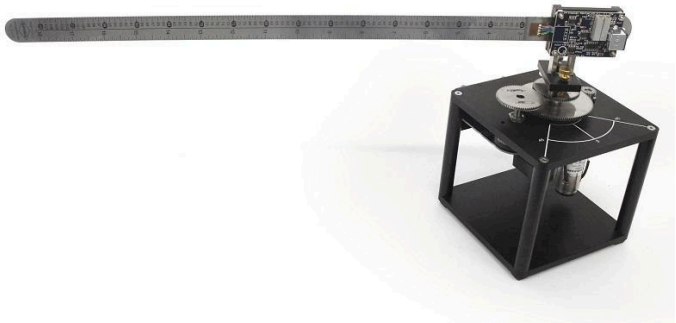

Figure 5.1: Qanser SRV02 with flexible beam module [**?**].

## 5.2 Background Information

Recall the 2-DOF state-space matrices that you found in Section **??** of Lab 1.

The open-loop poles are $\{0,\ -8.13 + 22.47i,\ -8.13 - 22.47i,\ -24.21\}$, and both the algebraic and geometric multiplicities are equal, thus the system is internally stable.

To verify whether we can use full state feedback to steer the system, one must compute the controllability matrix:

$$\mathcal{C}_{(A,B)} = \begin{bmatrix} 0 & 61.77 & -2501.26 & 62750.59 \\ 0 & -61.77 & 2501.26 & -41639.42 \\ 61.77 & -2501.26 & 62750.59 & -980692.86 \\ -61.77 & 2501.26 & -41639.42 & 125861.00 \end{bmatrix}$$

The system's controllability matrix has full rank, and hence the system is controllable. Thus, we can hope to use full state feedback and pole placement techniques to control the system.

## 5.3 Designing Feedback using the Linear Quadratic Regulator

A fundamental problem in control theory is operating a control system at minimum cost. When concerned with linear differential equations (e.g., $\dot{\mathbf{x}}(t) = A\mathbf{x}(t)+Bu(t)$) and quadratic cost functions, then one is really concerned with the *LQ problem*. In this lab, we will use a main result from this area of literature [**?**] which is the solution to the linear quadratic regulator (LQR). The solution to the LQR will help you design a feedback controller which minimizes the deflections of the flexible beam while the rotor base tracks a square wave trajectory (within certain *design constraints* on settling time and overshoot).
*Let us review some theory from class:*

The LQ problem that we are dealing with is an infinite-horizon problem that studies the control system

$$\begin{cases} \dot{\mathbf{x}}(t) = A\mathbf{x}(t) + Bu(t), & \text{where } \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{y}(t) = C\mathbf{x}(t) \end{cases}$$

subject to minimizing the cost function

$$\eta(u) = \int_0^\infty \mathbf{y}^T(t)\mathbf{y}(t) + u^T(t)u(t)\ dt = \int_0^\infty \mathbf{x}^T(t)C^TC\mathbf{x}(t) + u^T(t)u(t)\ dt.$$

You may think of the first part of the cost function, $\int_0^\infty \mathbf{y}^T(t)\mathbf{y}(t)dt$, as the *energy of the controlled output*, and the second part, $\int_0^\infty u^T(t)u(t)dt$, as the *energy of the control input*.

Two **main results** from class follow this reformulation: the first is that if $(A, B, C)$ is a minimal realization, then there exists a positive definite symmetric solution $\Pi_\infty \in \mathcal{M}^{n\times n}(\mathbb{R})$ to the continuous algebraic Ricatti equation (CARE):

$$A^T\Pi_\infty + \Pi_\infty A - \Pi_\infty BB^T\Pi_\infty = -C^TC;$$

the next main result is that with $(A, B, C)$ a minimal realization, the controller

$$u(t) = -B^T\Pi_\infty\mathbf{x}(t)$$

solves the LQ problem.

Q1: What does it mean to be a minimal realization? Is the state-space model of the rotary flexible beam system a minimal realization? Can one hope to use the LQR approach to optimize the feedback control inputs for the rotary flexible beam system?

*Back to your lab:*

There are a few differences in the theory that you've learned from class and the treatment we will apply here. First, we wish to minimize some of the system's outputs more than others, e.g., our focus is on minimizing the flexible beam deflections. With this focus, we present a more generalized form of the quadratic cost function:

$$\eta(u) = \int_0^\infty \mathbf{x}^T(t)Q\mathbf{x}(t) + u^T(t)u(t)\ dt, \tag{5.1}$$

where $Q$ is symmetric and $\mathbf{x}^T(t)Q\mathbf{x}(t) > 0$ for all $t$ (positive-definite). You will set the matrix $Q$ as a diagonal weighting matrix to assign different weights on the states within the cost function (5.1). In turn, this will determine how the control input $u(t)$ will minimize the cost function $\eta(u)$, and thus the weightings of matrix $Q$ will have direct effects on the ensuing feedback gain, $K$. Hence, the LQR design process is really an iterative process: tune the diagonal weightings of $Q$, observe the effects, compare against the desired performance metrics, and repeat.

In the theory from class, one had to verify that $(A, B, C)$ was a minimal realization in order for $u(t) = -B^T\Pi_\infty\mathbf{x}(t)$ to solve the LQ problem. Using the generalized quadratic cost function in (5.1), one must only verify that $(A, B)$ be stabilizable to ensure that this input solves the LQ problem. Can the LQR approach be used to optimize the feedback control inputs for the rotary flexible beam?

Open the Simulink model **lqr_model.mdl**, shown in Figure 5.2. You may have to connect the physical system to the computer, as was done in Section **??** of Lab 1. Notice the presence of a high-gain observer, used to estimate the time derivatives of the signals $\theta$ and $\alpha$, as was discussed in Section 4.3 of Lab 3. In this lab, you will use these signal derivative estimates as feedback. Notice also that the output signal is subtracted from the input signal before being scaled by the feedback gain. Thus when the system's outputs match the system's inputs, a signal of all zeros will be input into the physical system; when the system inputs and outputs do not match, the feedback gain should be designed to drive the non-zero states to zero (this will work if $K$ is designed to make the closed-loop system *asymptotically stable*).

Your first task will be to observe the effects of changing the diagonal weightings of $Q$. Using your state-space matrices from Section 5.2, and setting

$$Q = \begin{bmatrix} q_1 & 0 & 0 & 0 \\ 0 & q_2 & 0 & 0 \\ 0 & 0 & q_3 & 0 \\ 0 & 0 & 0 & q_4 \end{bmatrix}$$

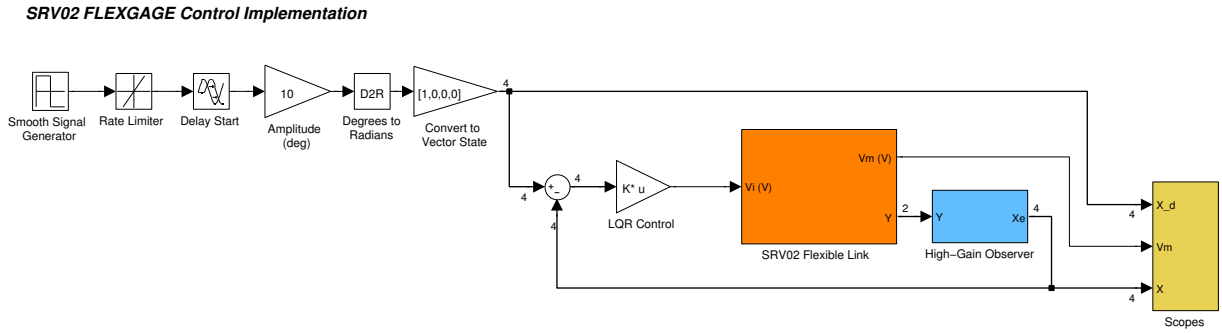


Figure 5.2: A Simulink model that inputs a signal into the rotary flexible beam system and observes its output. This model can be used to drive the rotor base to track a signal input while minimizing the flexible beam deflections via full state feedback. The feedback gain $K$ is designed using a linear quadratic regulator.

you can use the MATLAB command *care* to solve the continuous algebraic Ricatti equation (CARE). This will return the solutions to the Riccati equation, $\Pi_\infty$, which you will use to compute your feedback gain, $K = -B^T\Pi_\infty$.

Q2: Using a diagonal weightings matrix for $Q$, express the cost function symbolically. What would be the effect of increasing the diagonal elements $q_i$ on the feedback gain, $K$?

Q3: *Check your feedback gain with your TA*. Since the Simulink model *subtracts* the feedback from the input, make sure you define $K$ as $K = B^T\Pi_\infty$ and **not** $-K = B^T\Pi_\infty$. Vary the weightings $q_1,\ q_2,\ q_3,\ q_4$ independently from starting values $[q_1,\ q_2,\ q_3,\ q_4] = [1,\ 1,\ 1,\ 1]$. Run **lqr_model.mdl** with a square wave input of amplitude 30 **total** (there is a signal amplitude & an amplitude block, multiply these to get 30) and with frequency of 0.33 Hz. What effects does each modification have on the feedback gain, $K = [k_1,\ k_2,\ k_3,\ k_4]$? Plot each of the state responses with respect to time. What effects does each modification have on the physical system's state responses? What weighting(s) has(have) noticeable effects on reducing the rise time (one in particular is noticeable)? Did any decrease the flexible beam's deflections? *Formulate problems and their specifications* that you anticipate when tuning particular weightings in various *engineering applications* (e.g., for tidal power generators - or flexible beams mounted on the ocean floor in locations of notably strong tidal power - where the flexible beam's deflections trigger a power generator, but can also be strained to failure).

Q4: You will perform a similar experiment with the same input (square wave with amplitude of 30 and frequency of $0.33Hz$), except now there are *design restrictions*

on the rotor base's response: we require that maximum percent overshoot to be 10%, the rise time be within 0.5 seconds and the steady-state error be within 5% for use in our application. In addition, the maximum flexible beam deflection should be no more than $10^o$. Tune the diagonal weightings of $Q$ until you've achieved the desired results, and then plot the rotor's base angle, $\theta$, and the flexible beam's deflection angle, $\alpha$, with respect to time. Also plot the control input with respect to time. What weightings in the matrix $Q$ did you use to achieve these results, and what was the resulting feedback gain, $K$?

## 5.4 Effects of Partial State Feedback on LQR

Now, you will observe the effects of using only partial state feedback in your closed-loop system on the optimality of the control system. Open the Simulink model **lqr_partial_feedback_model.mdl**. Notice that the only state feedback being used is $\theta$ and the high-gain observer estimate of $\dot{\theta}$, as shown in Figure 5.3.

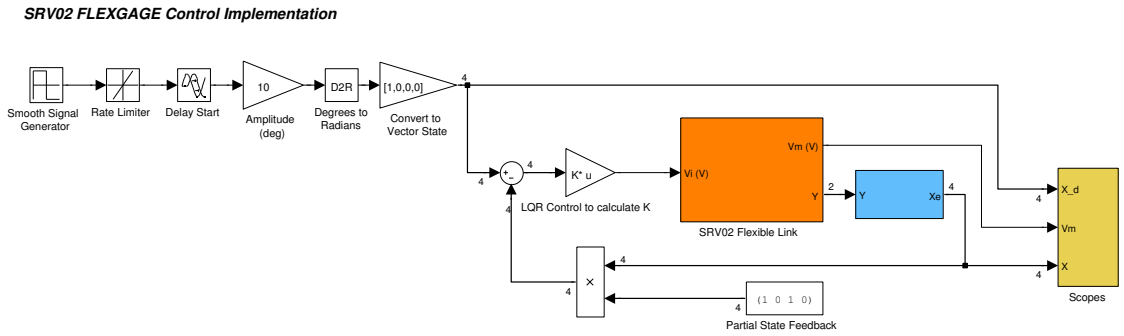


Figure 5.3: A Simulink model of the rotary flexible beam that utilizes partial state feedback and the linear quadratic regulator to design a feedback gain, $K$, that minimizes the deflections of the flexible beam while the rotor base tracks a square wave trajectory.

Q5: Similar to the last experiment, use the LQR technique and the same weighting matrix $Q$ when computing your feedback gain, $K$. Plot the system responses to a square wave function with amplitude 30 and with frequency of 0.33 Hz. Compare the optimality of full state feedback and the partial state feedback closed-loop systems. Describe at least two previously-learned techniques that you may employ (and describe how) to obtain a better state feedback design, leading to superior system behaviour.

## 5.5 Designing an Observer & Using Observer Estimate as Feedback for LQ Problem

Q6: Is it possible to construct an observer for the rotary flexible link system, for the scenario where the strain gauge, which reads the flexible beam's deflection angle, is damaged?

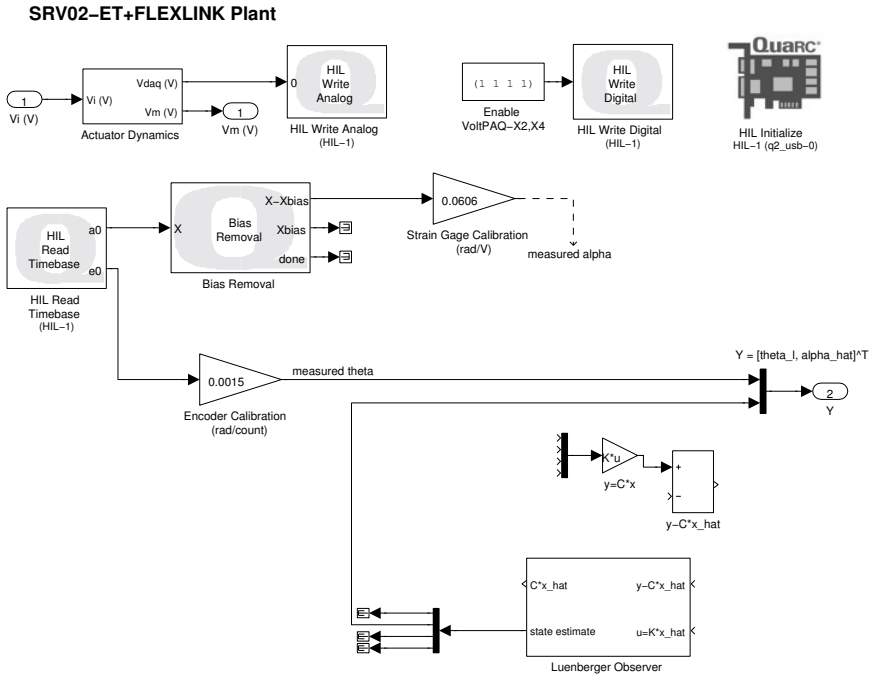


Figure 5.4: A Simulink subsystem block showing the rotary pendulum system and a Luenberger observer. These systems are not connected, however one can connect these in such a way to compare the measured and estimated pendulum angle, $\alpha$ and $\hat{\alpha}$, as well as use the pendulum angle estimate as LQR feedback. Note that this model will not run without modification.

Q7: Open the Simulink model **lqr_observer_model** with the Luenberger observer contained in the rotary pendulum's subsystem block, as shown in Figure 5.4. Follow the procedure from Section 4.4 of Lab 3 to build a Luenberger state observer to estimate state $\alpha$. You will need to tune your observer gain, $L$, similarly to how you did so in Lab 3 so that the observer dynamics are faster than the system dynamics. However, now your feedback gain $K$ is being produced by the linear quadratic regulator, so tuning $Q$ will alter $K$ and thus the observer gain $L$ may have to be tuned concurrently. **Make sure** to change your output matrix, $C$, such that $\alpha$ is not an output. Use the estimate of the beam's deflection angle, $\hat{\alpha}$, to calculate an estimate for the angular velocity of the beam via the high-gain observer subsystem. Plot the responses of the rotor base angle and the flexible beam deflection angle due to a square wave input of amplitude 30 and frequency 0.33Hz. On the flexible beam deflection angle plot, also plot your state estimate (you'll need to use a To Workspace block for this). What are your gains $L$ and $K$ and what is your weighting matrix $Q$? What can you conclude about the optimality (i.e., rotor base trajectory tracking and beam's deflections) of this control system when comparing it to the partial state feedback scenario and the full state feedback scenario?

## 5.6 Deliverables

To recap the lab objectives, this is what you need to include in your lab report to obtain full marks:

1. prelab question (i) and (ii);
2. the answers to questions Q3, Q4, Q5 and Q7 (include plots where requested).

# Appendix A
# Matlab

The students of this course should be familiar with the basic ideas of computer programming and Matlab from first year courses.

## A.1 Defining variables

- `x=3`
  Defines the variable `x` to be the constant 3.

- `x=(1,2;3,4;5,6)`
  Defines the variable `x` to be the $3 \times 2$ matrix

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

  Elements of a row are delimited by commas (or spaces) and each row is delimited by a semicolon.

## A.2 General Commands

- `dir`
  Displays a list of files in the current directory.

- `open lab_1.mdl`
  Opens the specific file in the argument. We will be dealing mostly with `*.mdl` and `*.m` files.

- `who`
  Displays a list of variables in the memory of Matlab.

- `simulink`
  Opens the `Simulink Browser Library` window. Using the GUI control, you can drag and drop the blocks to build the Simulink models needed for each lab. Details on using Simulink and WinCon are discussed in Appendix B.

## A.3 Plotting

- `plot (lab_1_Tachometer)`
  Plots the data from the variable `lab_1_Tachometer` in Matlab memory. You can check the list of variables in the Matlab memory by using the `who` command.

- `plot (lab_1_Tachometer,'r:')`
  Plots the result in the memory of Matlab and specifies the colour of the graph to be red and the line style to be dotted. Colour and line format are optional commands and they do not have to be specified for the plot command to produce an output. You can also specify one style parameter without the other. The default colour is blue and the default line style is solid. Table A.1 is a list of colours and line styles that can be used with the plot command.

| Line style/colour | Matlab command |
|---|---|
| solid | '-' |
| dashed | '--' |
| dotted | ':' |
| dash-dot | '-.' |
| blue | 'b' or 'blue' |
| black | 'k' or 'black' |
| cyan | 'c' or 'cyan' |
| green | 'g' or 'green' |
| magenta | 'm' or 'magenta' |
| red | 'r' or 'red' |
| white | 'w' or 'white' |
| yellow | 'y' or 'yellow' |

Table A.1: Colour commands in Matlab

- `title ('Angular Velocity of the Motor')`
  Sets the title of the plot to the text in quotations.

- `xlabel ('Time (s)')`
  Sets the x-axis label of the plot to the text in quotations.

- `ylabel ('rad/sec')`
  Sets the y-axis label of the plot to the text in quotations.

- `hold`
  Hold the current graph in figure and allow the user to plot more than one set of data on the same figure.

- `hold off`
  Release the current graph in figure and allow a new plot to replace the current graph.

## A.4 Control System Toolbox

- `sys = ss(A,b,c,D)`
  Defines the the state-space system from matrices $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$, and $\boldsymbol{D}$. For a model with $n$ states and 1 output,
  - $\boldsymbol{A}$ is an $n \times n$ matrix,
  - $\boldsymbol{b}$ is an $n \times 1$ matrix,
  - $\boldsymbol{c}$ is a $1 \times n$ matrix ($\boldsymbol{c}^t$ in our notation), and
  - $\boldsymbol{D} = [0]$ (always true for this class).
- `h = tf([1 0],[1 2 10])`
  Defines the variable `h` to be the transfer function
  $$\frac{s}{s^2 + 2s + 10}.$$
- `h = zpk([1 0],[-1 -2 -10],[3])`
  Defines the variable `h` to be the transfer function using the location of the zeros, poles, and a multiplicative constant:
  $$\frac{3s(s-1)}{(s+1)(s+2)(s+10)}.$$
- `h = tf(sys)`
  Defines the variable `h` to be the transfer function for a given state-space system.
- `sys = tf2ss[tf]`
  Gives the SISO linear system corresponding to the transfer function `tf`.
- `bode(sys)`
  Produces the Bode plots for the given system.
- `impulse(sys)`
  Produces the impulse response for the given system.
- `nyquist(sys)`
  Produces the Nyquist plot for the given system.
- `margin(sys)`
  Produces the gain and phase margins with associated crossover frequencies.

# Appendix B
# Simulink

Simulink allows simulation of complex control systems using a drag and drop block diagram interface. Simulink is especially useful when used in conjunction with the Real-Time Workshop which allows Simulink diagrams to be converted into C codes which can be run in real-time on a number of so-called targets (the PC being one such target).

## B.1 Starting

- To use Simulink, one must first start Matlab. After starting Matlab, you would type `simulink` in the Matlab command prompt to get the `Simulink Library Browser` window.
- Selecting `File`→`New`→`Model` (or `Ctrl+N`) while in the `Simulink Library Browser` will give you a blank model window into which you can drag-and-drop system blocks from the `Simulink Library Browser` to build a Simulink model. These models can be saved as `*.mdl` files for future simulations and editing.

## B.2 Building a Simulink model

- To connect the output of block A to the input of block B, simply left-click the output port of block A and drag the line that would appear into the input port of the block B.
- In order to connect the output of a block into the inputs of multiple blocks at the same time, you can right-click on an existing connection to get another line and drag that connection into the input of another block.
- When a block is dragged into the model window, it will be given a generic name. For example, when the `scope` block is dragged into a model, it would simply be labelled as "scope" and if it was the second `scope` block to be dragged into the model, it would be labelled as "scope2". It is a good practice to rename these blocks and give them more appropriate labels. These names are usually suggested in the diagram of the models in each lab. To rename a block, simply click on the existing name once and edit.

## B.3 Simulations

- One could view and edit the simulation parameters by clicking on

`Simulation→Simulation Parameters`

(or `Ctrl+E`) while editing the model. For the purpose of this course, there are only three things that you have to worry about:

1. Start and stop time: The default start time is 0 and the default stop time is 10. Usually, there is no reason to change the default start time, but you might find it useful to extend the stop time so that you could observe the simulation for a longer period of time.
2. Solver Method: You must have this set to a fixed-step when implementing real-time controllers. The default solver is a discrete method. You need to change it from the default method to `ode4`, which is an implementation of the Runge-Kutta method.
3. Step size: The default setting of 0.001 corresponds to 1000 Hz sampling frequency. This is the fastest rate at which the system can sample.

## B.4 Plotting

- The outputs of a simulation can be captured by using the `To Workspace` block. The data would be recorded as a variable in memory of Matlab. The default name of the output is `simout`, but you should change the name of this output just as you would give appropriate label to the block itself. One could plot the simulation output by using plot command discussed in Appendix A.

### B.4.1 Saving data via the "To Workspace" block

This is the preferred method of saving data to your Matlab workspace. The `To Workspace` block can be found at

`Simulink→Sinks`

Drag this block into your workspace and connect it to the variable you wish to save. Double click on the block to configure it. Choose a good variable name and in the `save format` drop down menu select `Structure with time`. After the simulation the data will be automatically saved to your Matlab workspace and you can plot it with the command

```
plot(varname.time,varname.signals.values)
```

replacing "`varname`" with the variable name you chose when configuring the `To Workspace` block.

### B.4.2 Saving data from a scope

You can also save scope data, but the scope seems to have short-term memory loss which makes it one of the most useless blocks in the simulink library. Nevertheless if you need to save the data from a scope, follow these instructions.

1. Double click on the scope you wish to save the data from.
2. Click on the Parameters Icon (in the top left of the scope dialog box).
3. Data History Tab
    - Uncheck box `Limit data points`
    - Check box `Save data to workspace`
    - Choose a variable name
    - Select `Array` from the Format drop down menu.
4. Click `Apply` and `OK` to exit.

After the simulation has been performed, the data will appear as an array in your Matlab workspace. You can plot the data with the command

```
plot(ScopeData1(:,1), ScopeData1(:,2))
```

where `ScopeData1` is the variable name that you set in the above steps.

# Appendix C
# Lab Equipment

In the lab, there are several computers equipped with data acquisition systems running Windows 7 with Matlab. The hardware equipment and some software tools are manufactured by Quanser Consulting, a Canadian company developing real-time control systems for education and research. This document introduces some of the hardware equipment to be used in the labs. Familiarity with this document is needed to perform the labs.

## C.1 Hardware devices

The lab hardware consists of three components:

1. data acquisition system;
2. power module;
3. servomotors.

## C.2 Data Acquisition Board

In order for the computer to run a controller, analog-to-digital (A/D) and digital-to-analog (D/A) conversions are necessary. These are done using the data acquisition and control board (DACB), which inputs the measured signal(s) to the computer and outputs control action to the actuator in the control loop. The DACB in this lab consists of a single board: the Q2-USB, which is made by Quanser Consulting. Figure C.1 shows a photo of the Q2-USB card. This data acquisition board is an external board connected to the computer through a USB port.

Figure C.1 also shows the proper configuration of the data acquisition board. The Encoder is the 5 pin Din and is plugged in to channel 0 in the Encoders portion of the board. The tachometer (S3 on the Quanser) is the red RCA plug and is plugged into channel 0 is the ADC portion of the board. Finally the analog output is the solo black RCA plug and is plugged in to channel 0 of the DAC portion of the board.

## C.3 Universal power module

The universal power module (UPM-2405), which is shown in Figure C.2, is a linear power operations amplifier. The Q2-USB data acquisition board cannot deliver enough power to the actuators used in this lab; therefore, a signal buffer is needed. The UPM-2405 is used

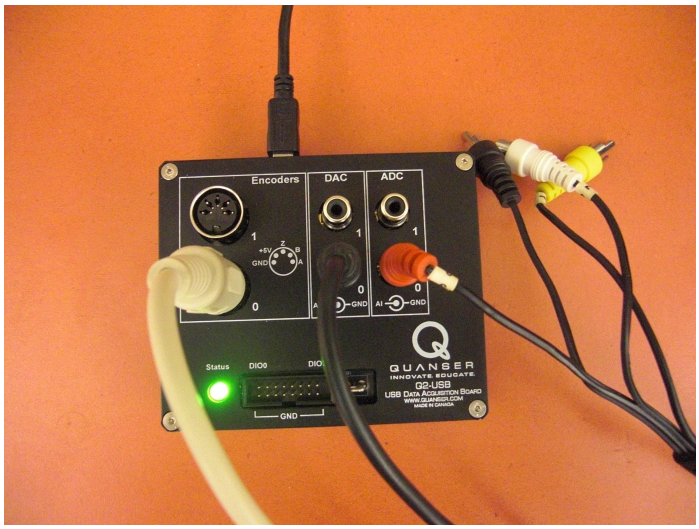


Figure C.1: Q2-USB data acquisition board

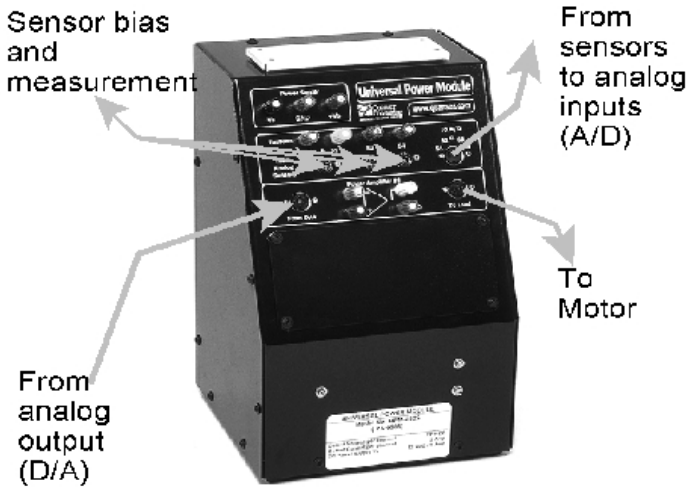


Figure C.2: Universal power module

as our signal buffer since it can deliver up to 5A to an actuator in a non-inverting, unity gain configuration.

The following connections can be made to/from the UPM (see the labels on the UPM).

- From analog sensors: there are four (S1-S4 inputs which can be connected from analog sensors (and then subsequently to the computer); the cable used is a 6-pin mini-din/6-pin mini-din cable (light tan colour), which is now referred to as the analog sensor cable.
- To A/D: the four analog sensor signals (S1-S4) can then be connected to the Q2-USB terminal board for A/D conversion into the computer; the cable used is a 5-pin din-stereo/4RCA cable (black colour), which is now referred to as the A/D cables.
- From D/A:/@ this is where you input the D/A signal from the Q2-USB terminal board to the UPM; the cable used is a 5-pin din-mono/RCA cable (black colour), which is now referred to as the D/A cable.
- To load: here you connect the amplified D/A signal to an actuator (e.g., servomotor); the cable used is a 7-pin din/4-pin din cable (black colour), which is now referred to as the load cable. Note there are two types of load cables one with unity gain and another with a cable gain of 5. Make sure you use the right one.
- Others: A few other connections are possible for convenience: e.g., a DC power supply on the top left provides 12 volts; the signal s from analog sensors S1-S4 can be easily monitored by connecting to a scope to the banana plug terminals.

## C.4 DC servomotor

The Quanser DC servomotor (SRV02) is shown in Figure C.3. A 3W motor is mounted in a solid aluminium frame and drives a built-in Swiss-made 14.1:1 gearbox whose output drives an external gear, which is attached to an independent output shaft that rotates in an aluminium ball-bearing block. The output shaft is equipped with an encoder. The external gear on the output shaft drives an anti-backlash gear connected to a precision potentiometer for measuring the output angle. The external gear ratio can be changed from 1:1 to 5:1 using different gears. Two inertial loads are supplied with the system in order to examine the effect of changing inertia on motor performance. Several connections are available for the servomotor. The input voltage connects to the UPM using the load cable. The potentiometer and tachometer ports connect to the UPM-2405 using sensor cables and are used to measure angular position and angular velocity respectively. Additionally, the shaft encoder port connects to the terminal board using an encoder cable and is used to measure angular position. The calibration factors listed in Table C.1 are needed in order to use the sensors in units of degrees or radians.


Figure C.3: DC servomotor (SRV02)

| Connection | Conversion (Rad) | Conversion (Deg) |
|---|---|---|
| Encoder | $-\frac{2\pi}{4096}$ | $-\frac{360}{4096}$ |
| Tachometer | $\frac{100\pi}{63}\frac{\text{rad}}{\text{sec}}$ | $\frac{18000}{63}\frac{\text{deg}}{\text{sec}}$ |
| Potentiometer | $\frac{1}{4096}$ | $\frac{180}{4096\pi}$ |

Table C.1: Conversion factors