Observer-based Feedback
MAE 433 Spring 2012
Lab 8

Prof. Rowley, Prof. Littman
AIs: Brandt Belson, Jonathan Tu

Princeton University

May 1 - May 4, 2012

1 Overview

This lab addresses the control of an inverted pendulum balanced on a rotary beam. The state of this system has four elements, corresponding to the rotary beam angle, the pendulum angle, and their rates of change. However, the only sensors available to us are two encoders which measure the angular positions. In order to apply full-state feedback laws, we implement an observer that reconstructs the full state from these two measurements. We then use observer-based feedback to control the pendulum.

2 Goals

Our hands-on goals for today are to:

- Design an observer to track the state variables of the pendulum system, using
  - Pole placement
  - Linear Quadratic Estimation (LQE)
- Use Bode plots to design an LQR controller for full-state feedback
- Use Bode and Nyquist plots to design observer-based controllers (LQR + observer)
- Implement observer-based control in experiment and verify its effectiveness

3 Pendulum down

First, we focus on observer design alone. That is, we do not apply any control to the system. As such, we must consider the pendulum in its down position. (The pendulum-up system is unstable

and requires control to stay near its equilibrium, where the linearization is valid.) The dynamics of this system are given by

$$\frac{d}{dt}\begin{bmatrix}\alpha\\ \theta\\ \dot{\alpha}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & -\frac{mgr}{J} & -\frac{K_2}{J} & 0\\ 0 & -\frac{(J+mr^2)g}{Jl} & -\frac{rK_2}{Jl} & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \theta\\ \dot{\alpha}\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{K_1}{J}\\ \frac{rK_1}{Jl}\end{bmatrix} u, \tag{1}$$

where $\mathbf{y} = \begin{bmatrix}\alpha & \theta\end{bmatrix}^T$ is the output vector and $u = V_{\text{in}}$ is the input voltage.

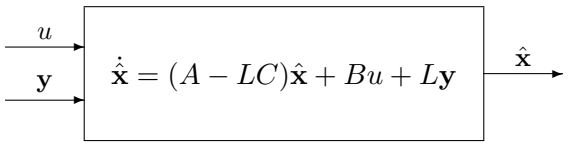
Recall that the state-space realization of an observer is given by

$$\begin{aligned}\dot{\hat{\mathbf{x}}} &= A\hat{\mathbf{x}} + Bu + L(\mathbf{y} - C\hat{\mathbf{x}})\\ \mathbf{y} &= C\mathbf{x},\end{aligned}$$

where $\hat{\mathbf{x}}$ is the state estimate. Using the shorthand notation for state-space systems, we can also write this as

$$G_{\text{obsv}}(s) = \left[\begin{array}{c|c} A - LC & \begin{bmatrix}B & L\end{bmatrix} \\ \hline I_{4\times 4} & \mathbf{0}_{4\times 3}\end{array}\right]. \tag{2}$$

We can also draw this as a block-diagram:



3.1 Pole placement

First, we design observers using pole placement.

1. Choose a set of closed-loop poles for the observer. Then compute the corresponding gain matrix L using the command `L = place(A', C', desired_poles)'`.
   We must use the dual system because the place command assumes we are trying to place the poles of $A - BK$, whereas we are really interested in the form $A - LC$.

2. In Matlab, use the `ss` command to create a state-space system for the observer. Name this system `downobsv`.
   Remember, the dynamics of the observer are given by (2).

3. Create and build the simulink model shown below.
   The black bars are `Mux` *(multiplexer) blocks, found in the* `Signals and Systems` *library. The block in the middle called* `Observer` *is an* `LTI System`*, found in the* `Control System Toolbox` *library. Set its value to be the state-space system* `downobsv`*. The* `Gain` *block provides negative feedback to bring the pendulum to its rest position. For now you can set the gain to 0, allowing you to move the pendulum freely.*

4. Create four scopes, each comparing a state variable to its estimate.
   In other words, plot α and $\hat{\alpha}$ in the same plot, etc. For the derivatives, also plot the signal computed via direct differentiation within the `Rotary pendulum` *block (e.g.,* `Alphadot`*).*

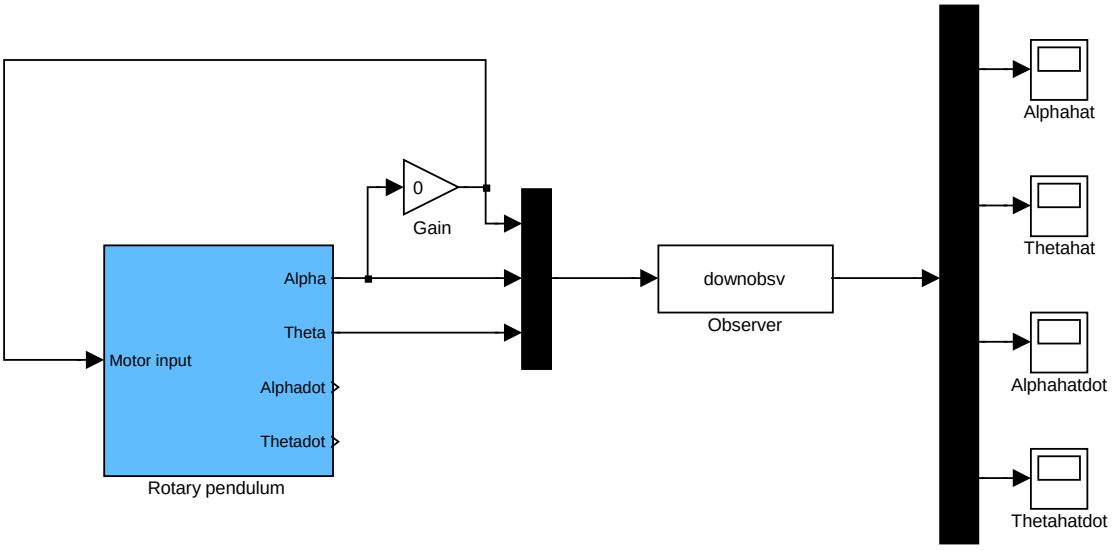


Figure 1: Simulink model for an observer.

5. Run your Simulink model. Move the pendulum and rotary beam around manually. Monitor the scopes to see if your observer tracks the state variables as desired.

6. Change the value of the `Gain` block to some *negative* value. Try swinging the pendulum (remember to use small amplitudes) and see if your observer still tracks the motion.
   Due to the structure of our Simulink model, we must use a negative value to achieve negative feedback.

7. By choosing different sets of closed-loop poles, design observers that exhibit:

   (a) poor tracking (too slow)

   (b) good tracking (fast).

   Check your observer gains L. Are these realistic for implementation in experiment? Think about sensor noise.

8. **Check in with an AI before proceeding.**

3.2 LQE

Next, we design optimal observers using LQE. These observers will balance the effects of input disturbances and sensor noise. We define the LQE weights to be

$$W = 1$$
$$V = \begin{bmatrix} v & 0 \\ 0 & v \end{bmatrix},$$

where v is a design parameter whose value we get to choose. W is related to the magnitude of input disturbances, and V to the magnitude of sensor noise.

1. Choose a weight on the sensor noise, v. Compute the corresponding LQE observer gains using the command `L = lqe(A, B, C, W, V)`.

2. Recompute the LTI system `downobsv` using the new gains L and run your Simulink model.

3. Vary v until you achieve good observer performance. What does the value of v tell you about how good your model is compared to how good your sensor measurements are?
Hint: think about what it means if V is much larger than W, or vice versa.

4. **Check in with an AI before proceeding.**

4 Pendulum up

Next, we use the intuition gained in the previous part of the lab to design observers for the pendulum in the up position, where the dynamics are given by

$$\frac{d}{dt}\begin{bmatrix}\alpha\\ \theta\\ \dot{\alpha}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 0 & -\frac{mgr}{J} & -\frac{K_2}{J} & 0\\ 0 & \frac{(J+mr^2)g}{Jl} & \frac{rK_2}{Jl} & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \theta\\ \dot{\alpha}\\ \dot{\theta}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ \frac{K_1}{J}\\ -\frac{rK_1}{Jl}\end{bmatrix} u. \tag{3}$$

We will combine these observers with a state-feedback controller to stabilize and control the pendulum. The pairing of an observer with a state-feedback controller yields something commonly referred to as a "compensator."

4.1 LQR design

In last week's lab, we designed LQR controllers using intuition. This week, we will do a more systematic design, using Bode plots. Recall that for state-feedback control, the loop gain is given by

$$L_{\mathrm{LQR}}(s) = \left[\begin{array}{c|c} A & B \\ \hline K & 0 \end{array}\right]. \tag{4}$$

1. Consider the Bode magnitude plots given below. Do the labels make sense? Can you explain them?
Hint: think about performance and bandwidth.

2. Choose three sets of LQR weights that you think will yield controllers that are too slow, too fast, and just right. Compute the corresponding controller gains K and plot the resulting Bode magnitude plots. Verify that they are similar to the ones shown in Fig. 4.1.
Use the intuition you gained from the previous step to choose the LQR weights Q and R. If you forget how to do compute the LQR gain K, refer to your Matlab scripts from last week.

3. Implement your LQR controllers in experiment. Do they perform as expected?
You should be able to use your Simulink model from last week's lab.

4. **Check in with an AI before proceeding.**

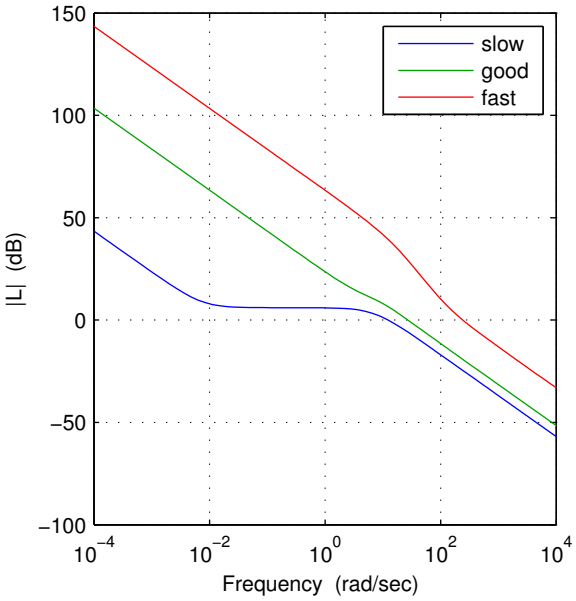


Figure 2: Sample Bode magnitude plots.

4.2 Observer-based feedback

Recall that in practice, we often cannot measure every element of the state. For instance, in this experiment we have two encoders that measure the two angular positions. We cannot directly measure their derivatives. Thus to implement state feedback (nominally $u = -K\mathbf{x}$), we will have to estimate the full state. This will allow us to apply feedback laws like $u = -K\hat{\mathbf{x}}$.

1. What is the problem with simply differentiating the signals for α and θ to get their derivatives? Can you prove this rigorously?
   Hint: think about the transfer function that corresponds to taking a derivative. What does this look like at high frequencies?

2. Do the following, first for an observer designed with pole placement, then one designed using LQE:

   (a) Using your "good" poles/weights from above, design an observer gain L for the *pendulum-up* dynamics.
   Remember to change the A and B matrices!

   (b) Create a state-space system for the compensator, which has the transfer function

   $$G_{\text{upcomp}}(s) = \left[\begin{array}{c|c} A - BK - LC & L \\ \hline K & 0 \end{array}\right].$$

   Name this variable `upcomp`.

   (c) Create a state-space system for the plant, whose transfer function is

   $$P_{\text{up}}(s) = \left[\begin{array}{c|c} A & B \\ \hline C & 0 \end{array}\right].$$

   Name this variable `upplant`.

(d) Multiply the `upplant` and `upcomp` systems together to get the loop gain for the observer-based system, $L_{\text{obs}}(s)$.
Be careful about the order in which you do this multiplication.

(e) Compare the Bode and Nyquist plots of this system to those of the state-feedback loop gain $L_{\text{LQR}}(s)$. Adjust your poles/LQE weights if necessary to get good Bode and Nyquist plots for your observer-based loop gain.
Think about things like performance, bandwidth, and robustness/margins. Remember, our LQR loop gain is optimal for the cost function we chose (defined by Q and R). If our estimator is good, it should achieve similar performance to the LQR state-feedback system.

(f) Create a state-space system for the observer alone, whose transfer function is given in Eq. (2). Name this variable `upobsv`.

(g) Create and build the Simulink model below.
In the `Gain` *block, be sure to select* `Matrix multiplication (K*u)` *from the drop-down menu. Note that here* `u` *here is just what Matlab calls the input to the* `Gain` *block. It is* ***not*** *the motor voltage u. Set the value for the gain to be $-K$, where K is the matrix of LQR gains.*

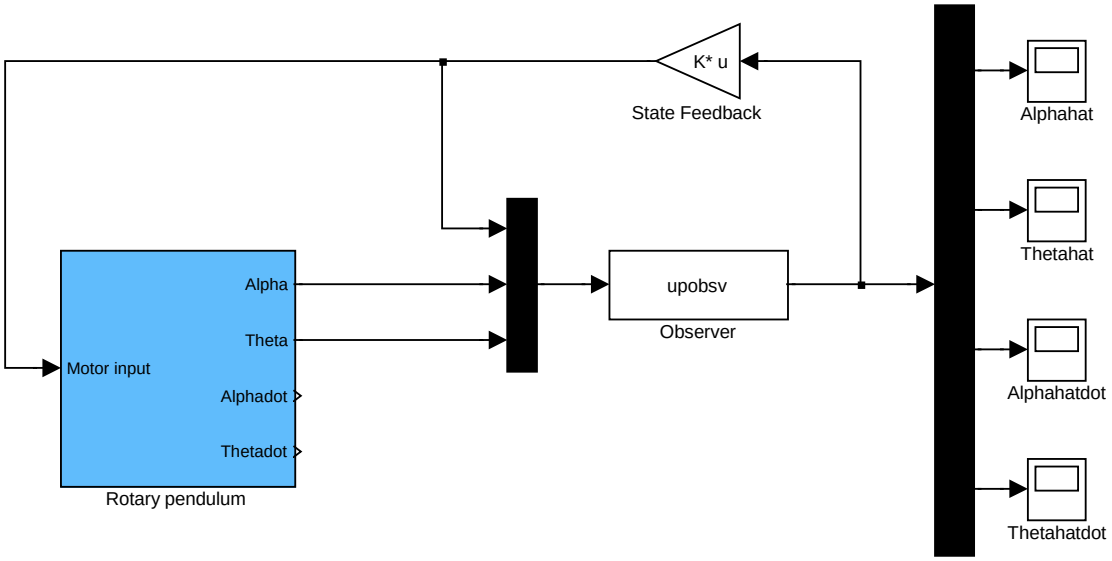


Figure 3: Simulink model for observer-based control

(h) Run the Simulink model. Does the system perform as well as you expect? If not, adjust the poles/LQE weights as necessary to achieve good performance.
If you need to adjust your designs, use the Bode and Nyquist plots of the loop gain to guide you.

(i) As in the previous two labs, modify your model so that you can track a reference value α_{ref}.
For your `Scope` *on* `Alphahat` *to make sense, you will have to apply the same* `alpharef` *offset to the* `Alphahat` *signal, before it reaches the* `Scope`.

(j) Run the following experiment and save the data for your lab report:

i. Set α_{ref} to 10 degrees.

ii. Run your model.

iii. After the rotary arm settles in at its commanded position, tap the pendulum to give it a disturbance.

iv. Let the system come to equilibrium.

v. Save this data.

(k) **Demonstrate the performance of your observer-based controllers to an AI.**

5 Deliverables

The following should be included in your lab report, for both the pole placement-based compensator and the LQE-based compensator:

1. Desired poles/LQE weights used for observer design.
2. Weights used for LQR design.
3. Bode magnitude plot of LQR loop gain and loop gain for observer-based control (together in one plot).
4. Nyquist plot of LQR loop gain and loop gain for observer-based control (together in one plot).
5. For each state variable, a comparison of the measured and estimated state from the observer-based control experiment (four plots total).