Lab Procedure

LQR Balance Control Virtual

Introduction

Ensure the following:

1. You have reviewed the Application Guide – LQR Balance Control
2. Make sure you have Quanser Interactive Labs open in the Qube 3 – Pendulum → Pendulum Workspace.
3. Launch MATLAB and browse to the working directory that includes the Simulink models for this lab.

The **Hardware Interfacing** and **Filtering** labs explained the basic blocks to read and write from the Qube-Servo 3. For simplicity, all labs forward will use a Qube-Servo 3 block that sets up the system beforehand and outputs the available information from the Qube.

In this lab you will use the qs3_lqr_ctrl.slx file to complete the model as shown in Figure 1. This model will balance the Qube-Servo 3 rotary pendulum system using a generated control gain K.

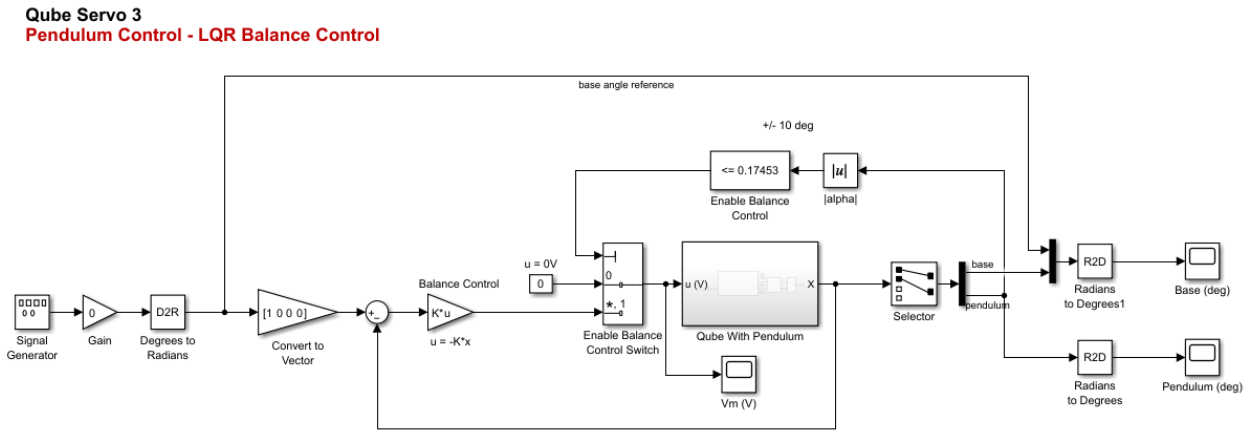

Figure 1: LQR pendulum balance control model

The LQR theory has been packaged into the MATLAB *Control System Toolbox*. Given the state-space matrices A and B that model the system and weighting matrices Q and R, the LQR function computes the feedback control gains automatically.

In this lab, the effect of changing the weighting matrix, Q, on the cost function, J, will be explored. The other weighting matrix, R, will be held at 1.

Exploring LQR Control Design

1. In the pendulum state space modeling lab, you completed the *rotpen_ABCD_eqns_ip.m* file. Use that file and replace the *rotpen_ABCD_eqns_ip.m* file that is empty in this folder.

2. In MATLAB, run the ***setup_qube3_rotpen.m*** script, this will load the Qube-Servo 3 rotary pendulum state-space representation matrices A, B, C, and D as well as the *qube_3_rotpen_param.m* that includes all the fixed values of the Qube system.

   The A and B matrices should be displayed in the Command Window:

   ```
   >> setup_qube3_lqr
   A =
         0         0    1.0000         0
         0         0         0    1.0000
         0   55.1525   -4.5471   -0.1816
         0  168.5810   -4.4942   -0.5551
   B =
         0
         0
    20.6755
    20.4351
   ```

3. Define the matrices Q and R as follows in the MATLAB Workspace:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad and \quad R = 1$$

   Then generate the gain K using the `lqr` command. Take a screenshot of the control gain generated.

4. Change the LQR weighting matrix to:

$$Q = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

   Take a screenshot of the new control gain generated.

5. Explore the effects on the control gain, K, of changing each of the diagonal elements in Q to 5, while keeping all other diagonal elements at 1. For example, the next Q to try is:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Write down the effects of changing each of the diagonal elements in the weighting matrix Q.

6. Use the *eig* command to find the open-loop poles of the system. Take a screenshot of the results from the command window and note how the poles affect the system.

Constructing the Model

When performing this experiment, make sure the rotary arm is centered about the $0°$ marking on the Qube-Servo 3, you can turn the resting position of the pendulum by turning the plug into the Encoder 1 plug. Make sure it is connected all the way in. Also make sure the pendulum arm is completely at rest. If the encoders are not initialized at the correct zero positions, it could influence the system response.

7. Open the Simulink model qs3_lqr_ctrl.slx. We will try to construct the model in Figure 1.
8. Add a `Signal Generator` with an amplitude of 1 and frequency of 0.125 Hz.
9. Add a `Gain` (of 0) and a `Degrees to Radians` block as the desired input to the system as shown in Figure 1.
10. Add a `Gain` block of $[1 \quad 0 \quad 0 \quad 0]$ to convert the square wave to a reference state.
11. Add the logic to make sure the controller is only enabled when the pendulum is $\pm 10°$ ($\pm .175\ rad$) from the upright position. Add an `Abs` and `Compare to Constant` blocks.
12. Make sure that inside the `Qube With Pendulum` subsystem, the `Counts to Angles` subsystem matches the one you created in the Pendulum Interfacing lab to convert encoder counts to radians and use the inverted pendulum angle (using `mod`).
13. Make sure that the subsystem `State X` outputs your desired state space variables of $x = [\theta \quad \alpha \quad \dot{\theta} \quad \dot{\alpha}]^T$. Use a `Mux` block so that the output is a single signal.
14. The `Transfer Fcn` blocks act as a derivative and low-pass filter to obtain the velocity of the rotary arm and pendulum. Note how we don't use the tachometer for speed in this case. The tachometer is not the best when movements are very small. Using the derivative of the position is the better solution in this case.
15. Add `Radians to Degrees` block to make sure the existing scopes `Base` and `Pendulum` have the information in degrees instead of radians.
16. Connect your system so it now looks like the model in Figure 1.
17. If your model was open before you follow the instructions in the previous section, the `Gain` block that uses K will have a red outline. If this is the case, `Ctrl+D` on the model, it should grab the variable from your workspace and not show a problem anymore.
18. Load the control gain, K, from Step 3 in the previous section into the MATLAB workspace.

Testing the Model

19. Run the QUARC controller using the **Run** button on the **Simulation** tab.
20. Bring the pendulum up using the lift pendulum button until the controller engages.
21. Once the pendulum is balanced, set the **Gain** to 30 to make the rotary arm angle go between $\pm 30°$. The scopes should be similar to Figure 2 below.

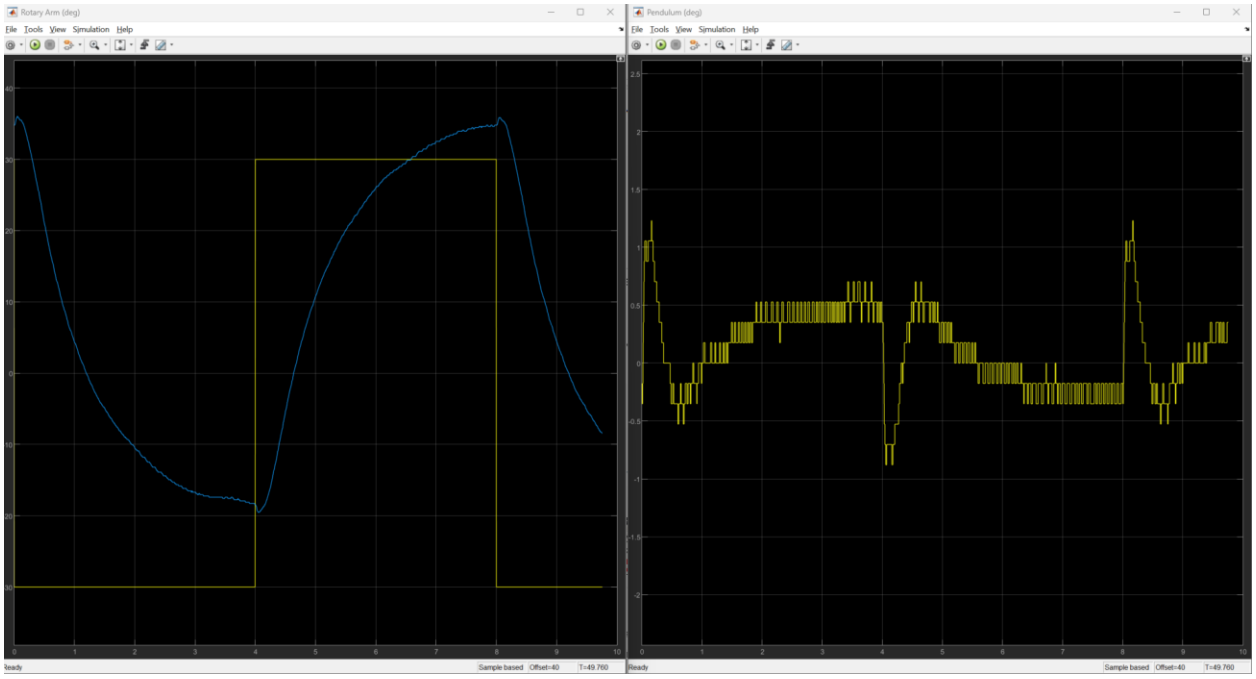

Figure 2: Sample response of balancing the pendulum between rotary arm angles.

22. In MATLAB generate the control gain for $Q = diag([5 \quad 1 \quad 1 \quad 1])$. The *diag* command specifies the diagonal elements in a square matrix.

```
>> Q=diag([5, 1, 1, 1])

>> K=lqr(A,B,Q,R)
```

23. To apply the newly designed gains to the model while it is still running, go to the Simulink model and from the **Modeling** tab, select *Update Diagram* or press **CTRL+D**.
24. Take a screenshot of scopes demonstrating the new behavior.
25. Tune the diagonal elements of the Q matrix so that the criteria below are met. Write down the procedure for adjusting the weightings in Q.
    a. Pendulum angle overshoot is less than $\pm 3°$ ($\alpha < |3°|$).
    b. Overshoot of the base angle is less than 14%.

26. The measurement tool can be used to measure the performance of the controller as shown in Figure 3 below:

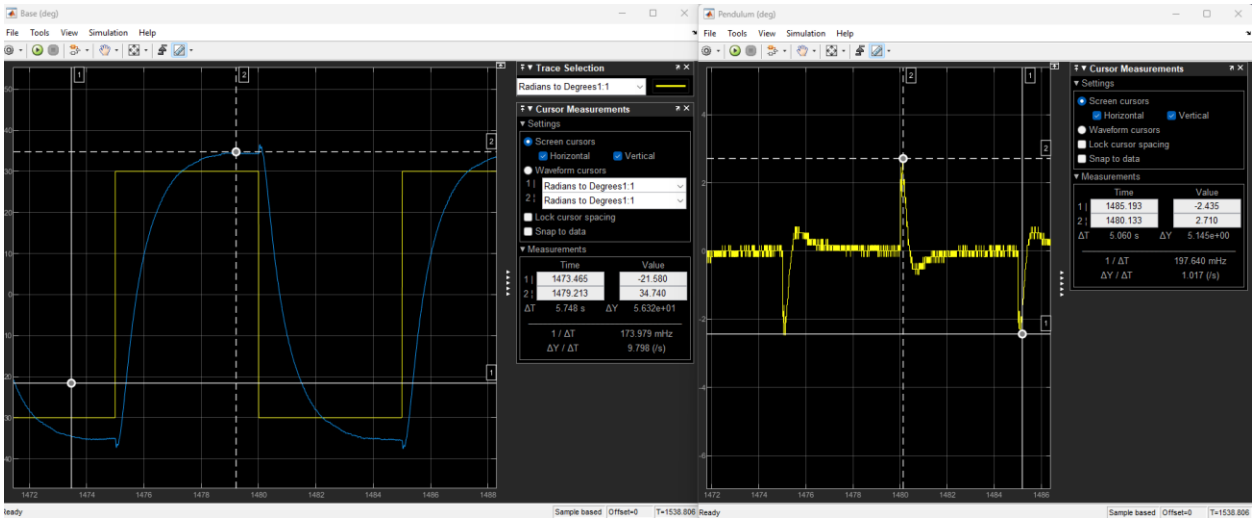

Figure 3: Sample response of an LQR controller meeting the design requirements.

27. Take a screenshot of the scopes demonstrating the desired behavior. Write down the final Q matrix used and resulting control gains, K.

28. Stop and close your model. Ensure you save a copy of the files for review later.

29. Close Quanser Interactive Labs.