24-774: Special Topics in ACSI

Laboratory 2: State Space Control

Background: The purpose of this lab is to practice hardware implementation of state space controllers. You will be provided with pre-made Quanser labs for your reference along with sample code for the Arduino – if you are having trouble, I would recommend going through some of the Quanser labs for guidance. Before performing the tasks here, please read the user manual for the Qube Servo 2 system to understand the encoder / motor parameters and limitations.

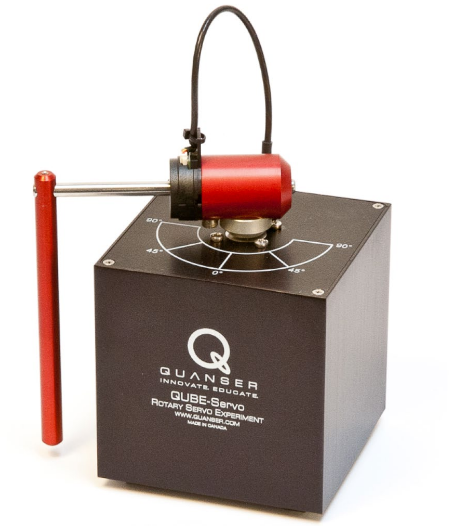

Figure 1: Quanser Qube Servo System. This is a classical single input, dual output control problem.

It is expected that you will work through this lab with some degree of parallelization, i.e., not everyone needs to be involved in every task. However, you must ensure that the work load is balanced, and that every team member understands the content that was generated by others. The report should also be written in parallel, but each team member is responsible for proof reading the final report.

The hardware systems are located in ANS 106; you should have badge access through your IDs. Please use this spreadsheet to schedule lab access https://docs.google.com/spreadsheets/d/1piBprRu7EQnM1eGTL_MRaxKbrmVcofhJ0Ud9JEeo0t4/edit?usp=sharing, adding columns for dates as needed. To ensure fair access, each team will have a maximum of 4 hours / day in the lab; as a courtesy to others, please do not reserve times until you know you will use them. Your success will critically depend on starting early and putting in a consistent effort, rather than waiting until the last days before the due date.

The computer password is acsi21.

Background

Pendulum dynamics

The dynamics of the Furata pendulum are well known – a good summary is here: https://en.wikipedia.org/wiki/Furuta_pendulum.

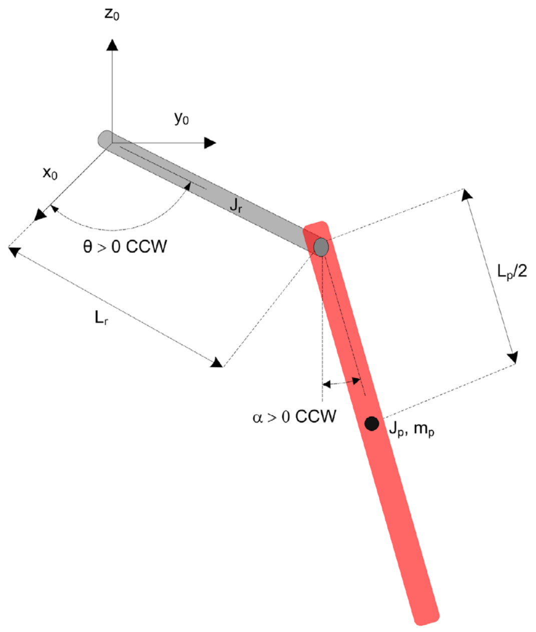

Figure 2: Geometry setup for system dynamics. The pendulum angle is zero when the pendulum is straight down.

The equations for our system are (with signs corrected)

$$\left(m_pL_r^2 + \frac{1}{4}m_pL_p^2 + \frac{1}{4}m_pL_p^2\cos^2\alpha + J_r\right)\ddot{\theta} + \frac{1}{2}m_pL_pL_r\cos\alpha\,\ddot{\alpha} + \frac{1}{2}m_pL_p^2\sin\alpha\cos\alpha\dot{\theta}\dot{\alpha}$$
$$-\frac{1}{2}m_pL_pL_r\sin\alpha\,\dot{\alpha}^2 = \tau - D_r\dot{\theta}$$

$$\frac{1}{2}m_pL_pL_r\cos(\alpha)\ddot{\theta} + \left(J_p + \frac{1}{4}m_pL_p^2\right)\ddot{\alpha} - \frac{1}{4}m_pL_p^2\cos(\alpha)\sin(\alpha)\dot{\theta}^2 + \frac{1}{2}m_pL_pg\sin(\alpha) = -D_p\dot{\alpha}$$

$$\tau = \frac{k_m \left(V_m - k_m \dot{\theta} \right)}{R_m}$$

where the parameters are given by

Variable	Value (in consistent SI units)
R_m	8.4
k_m	0.042
M_r	0.095
L_r	0.085
J_r	5.72e-5
D_r	0.0015
M_p	0.024
L_p	0.129
J_p	3.33e-5
D_p	0.0005

State Estimator Design

State space controller design techniques are ready-made for MIMO plants using state feedback. However, in this system we can only measure the angles and not their rates, so we will need an observer to extract the velocity.

1. Quanser's approach to state estimation is differentiation of position measurements followed by a low-pass filter (LPF). Design an observer (Luenberger observer or Kalman filter (KF) – your choice) to estimate the rotation rates. Implement your observer with hardware in the loop using the provided Simulink model.

   Discussion:

   a. Discuss your observer design process. How did you decide on the pole locations or weights for the KF? Note that it is probably easiest just to estimate all states rather than using a reduced order observer.
   b. Compare your state estimates to the simpler approach of derivative plus LPF as you move the hardware system around. For the LPF approach, we can hit the measured angles by a transfer function of the form $D(S) = \frac{S}{\frac{S}{\tau}+1}$ where τ sets the

low pass filter frequency. In their work Quanser uses $\tau = 50$, but you are welcome to play with that parameter. Is the simpler approach sufficient?

2. Implement your state observer in Arduino code. NOTE: An observer is just a state space system! You can leverage your code from Lab 1 to implement this. You do not need to set motorVoltage and drive the motor for this task. For logging the data, you can modify the Display and buildString function in QUBEServo.cpp, or directly use the Serial.print function of the Arduino library.

   Discussion:

   a. Plot the state measurements (4 states) from the Arduino as you move the pendulum around. You time histories of the states you can approximate the derivatives in post processing. Do your state estimates you found with your observer match the post-processed derivatives of the measured angles closely?

<u>In summary:</u>

- *Outside the lab*
  - Design your observer and implement on your Simulink model.
  - Design your derivative + LPF approach and test on your Simulink model.
  - Write your Arduino code (can't test outside, but you can at least compile)
- *In the lab*
  - Implement your observer / LPF approach with the hardware in Simulink & collect data
  - Test your Arduino code and collect data

LQR Controller Design for Inverse Pendulum Control

For this 4-state system LQR is a very effective control design method. How you do the state estimation for this part is entirely up to you, but should probably come from one of the methods above.

1. Using the <u>continuous time</u> state space model, design an LQR controller that stabilizes the system in the **upright position** while tracking setpoint changes in motor angle. Implement the controller for the nonlinear Simulink model (no hardware) and plot the response starting at an initial condition of $\alpha = \pi/6$ for a zero reference in motor angle.

   Discussion:

   a. Describe / justify your design approach. How did you choose Q and R?
   b. Using your simulation, estimate the range of pendulum angles for which the controller stabilizes the nonlinear system.

2. Implement your controller on the hardware using Simulink. The controller should be enabled when the pendulum is sufficiently close to upright ("sufficiently" based on your analysis above). You might find the “triggered subsystem” block to useful in implementing your controller only when the pendulum is near the upright position. Demonstrate the performance by tracking square waves in motor angle of magnitude 10 degrees at frequency 0.2 Hz.

Discussion:

a. Plot the errors in tracking the reference waveform. Compare this performance with your simulation (does not need to be run in parallel).

3. Implement the LQR controller on the Arduino. Capture videos of the response to square waves and post to Youtube. Tip: the “triggered subsystem” was useful for the Simulink controller implementation. How can you implement the same “trigger” in your Arduino code?

   ***Discussion*:**

   Test the performance of your controller for various sampling rates <u>without redesigning your gain matrix</u>. At what sampling rate do you lose stability? Compare with simulation results, with your Simulink model updated to incorporate embedded limitations (e.g. quantization).

<u>In summary:</u>

- *Outside the lab*
  - All of #1
  - Create Simulink model for #2, including triggering the controller when upright
  - Write your Arduino code (can’t test outside, but you can at least compile)
- *In the lab*
  - Hardware Simulink testing and data collection
  - Arduino code testing and data collection

Tips and Tricks:

- See the QuickStart instructions to see how to build and run Simulink models on the hardware. I would start by running the example model.
- The controller implementation acts like it runs continuous time plants, BUT it actually samples at 500 Hz. Check your controller poles to make sure they’re below ½ that frequency. You will almost certainly be unstable if you run a controller with faster poles.
- Be careful about signs. The nice thing is that the instability will not be catastrophic. If things aren’t working, just flip the controller sign and try again.

Reporting

Compile a single PDF lab report following the guidelines here. Your report should be organized in terms of numbered items in the lab procedure. For each numbered item in the lab procedure you must address the following items at a minimum:

1.) The details of all calculations involved in generating your results. Be sure to highlight the main results.
2.) Presentation of your results in the form of plots and tables. This should include all relevant plots and Simulink models. Do not present plots that use the black background that is the Simulink scope default.
3.) General discussion. What sense do you make of the results? What can you conclude?
4.) Answers to all the discussion questions in the lab procedure.

After completing these tasks for all numbered items in the lab procedure, complete the following sections to finish your report:

- Conclusions: What were the main results? What did you learn (if anything) by completing the lab? What suggestions do you have to make the lab better or more interesting?
- References: Compile all of your references into a single section at the end of the document. I highly recommend the use of a reference manager, e.g. Bibtex, EndNote, etc.
- Team roles: What role did each member of the team play in generating results / writing the report? I strongly encourage your team to rotate the role of the primary report writing throughout the lab assignments.