24-774: Special Topics in ACSI
Laboratory 1: Controller Design and Implementation

Background: The goal of this lab is to apply controllers designed through state space methods and classical control techniques to a DIDO hardware plant. For some of you this lab will likely be a review, whereas for others it might contain substantial new material. 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 Aero system to understand the encoder / motor parameters and limitations.

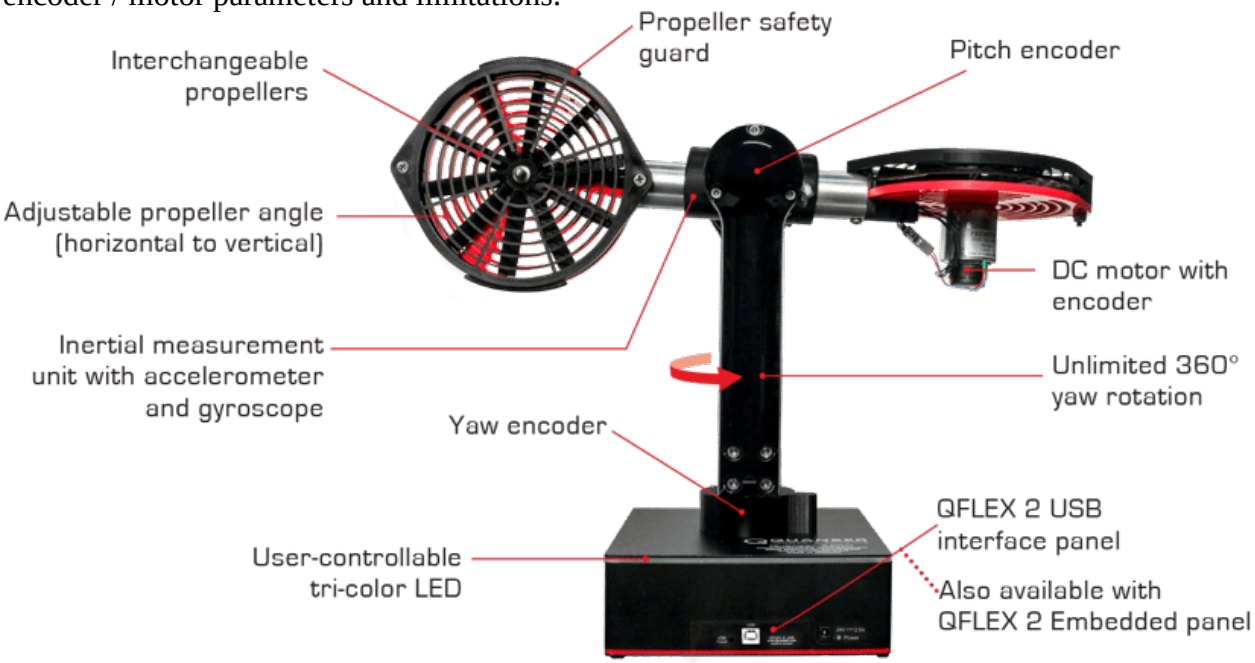

Figure 1: Quanser Aero System. The system has two degrees of freedom: pitch and yaw.

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.

Please use the following to schedule your time in the lab: **https://tinyurl.com/y2kr2es8**. 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. A proposed schedule is included to help guide your work.

The lab computer is a shared resource. **The password is acsimlc2019**. To avoid cluttering the workspace for others, please store all data on a thumb drive or on the cloud and delete from the computer before leaving the lab.

Plant Model and Reference:

The files *quanser_aero_parameters.m* and *quanser_aero_state_space.m* describe the plant model for the system. Based on the Quanser documentation, the nominal plant model is given by

$$\dot{x} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1.7117 & 0 & -0.3249 & 0 \\ 0 & 0 & 0 & -1.0004 \end{bmatrix} x + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0.0503 & 0.0959 \\ -0.1228 & 0.1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} x$$

To test performance you will be following references and monitoring the response of the angles. For simplicity we will consider only square wave references. The yaw reference should have an amplitude of $\pi/4$ radians and a frequency of 0.5 rad/s, whereas the pitch reference should have an amplitude of $\pi/6$ radians and a frequency of 0.4 rad/s.

Procedure

Preliminaries

1. Plot the plant's Bode plot. Comment on any unusual features that you see.
2. Build a Simulink model based on the nominal plant and including a saturation nonlinearity that limits control voltages to ± 25 V. You will use this simulation to test your controllers' performance before you test them on the hardware.

Simulink Control

The Quanser hardware provides a simple interface to Simulink, which is useful for controller prototyping. You should use the Simulink model developed above to test controllers *before* applying them to the hardware system.

For each of the hardware exercises (Simulink and embedded), please capture a short video that demonstrates performance and post to Youtube.

Classical Control

1. Design a diagonal PID feedback controller, i.e., design and implement two SISO PID controllers, one for each of the diagonal plant entries. Adjust your design around the simulation model to anticipate the effects of saturation. Test the performance in hardware using the reference discussed above.

   ***Discussion*:**

   a. Compare the performance between the hardware plant and your Simulink model. NOTE: These will probably not match well.

   b. Diagonal controllers treat coupling between axes as a disturbance. Quantify and discuss the amount of coupling you see between axes. Turn off the reference to each axis one-by-one, and collect data that shows the amount of motion induced in the coupled axis by tracking the primary square wave.

2. Design a DIDO feedback controller based on your model and diagonal controller using dynamic decoupling (see Figure 2). In dynamic decoupling, controllers are designed for a shaped plant $G_s = G_D G_M^{-1} G$ where G_M is the plant model, G_D is the diagonal portion of the plant model, and G_M^{-1} is a proper, stable approximation to the plant model inverse. Assuming that your decoupling controller works properly, your diagonal controller should be effective for the shaped plant. As before, test the performance in hardware using the reference discussed above. NOTE: Your final controller should be $K = G_M^{-1} G_D K_D$ where K_D is the diagonal controller you designed previously.
   Discussion:
   a. Compare the performance between the hardware plant and the diagonal controller applied to a diagonal plant in your Simulink model. The latter is the ideal scenario where your inverse / model are perfect.
   b. Once again quantify and discuss the amount of coupling you see between axes. Did dynamic decoupling improve the MIMO performance?

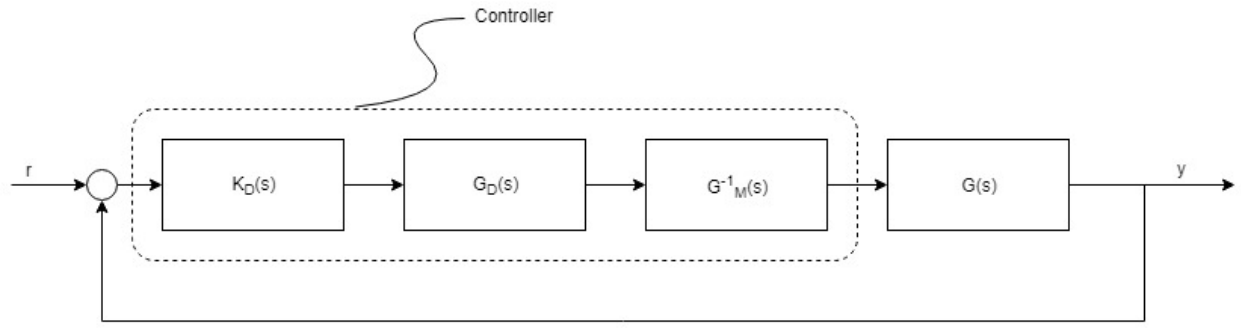

Figure 2: Decoupling controller schematic

State Space Control

State space controller design techniques are ready-made for MIMO plants, so there is no need to attempt a diagonal design. 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.
   Discussion:
   a. Discuss your observer design process. How did you decide on the pole locations or weights for the KF?
   b. Compare your state estimates to the simpler approach of derivative plus LPF. Which works better? Why?
3. Now that we have an estimate of the full state (you can use the Quanser estimates for this), we can design a state feedback controller that tracks the reference. Design a state feedback controller (pole placement or LQR – your choice) to track the reference signals. As before, test the performance in hardware using the reference discussed above.
   Discussion:
   a. Discuss your feedback design process. How did you decide on the pole locations or weights for the LQR?

b. Discuss the performance of your state space controller. How does the performance / cross-coupling compare to the classical decoupling controller?

Embedded Control

Now that you have developed effective controllers, it is time to implement them on embedded hardware for deployment. For simplicity, we will use an Arduino to implement the controls.

1. Create a discrete-time version of the DIDO decoupling controller via emulation and implement the controller on the Arduino. Note that this controller should not be of particularly high order, so simple implementations (e.g. controllable canonical form) will likely be sufficient. Create the reference signal and demonstrate tracking control.
   Discussion:
   a. Discuss the steps of your emulation process and your code implementation on the Arduino.
   b. Test the performance of your controller for various sampling rates. At what sampling rate do you lose stability?
2. Implement a discrete-time version of your state space controller on the Arduino. Once again, track the reference.
   Discussion:
   a. Compare the process for implementing state feedback-based and classically designed controllers in Matlab. Which was easier for you?

Suggested schedule

Date	Deliverable
9/6	Preliminaries / Classical Control
9/13	State Space Control / Arduino code written
9/18	Embedded Control

Tips and Tricks:

- The plant has a massive deadzone nonlinearity. DO NOT expect the hardware results to closely match your simulations. However, the trends between controllers may be comparable.
- While the simulation may appear to be continuous-time, in fact the controller is discretized before implementation. The sampling rate of the model is 1 kHz, and hence the Nyquist frequency is 500 Hz (~3000 rad/s). As a rule of thumb, you should not have controller poles above ~300 rad/s to see good discrete-time reconstruction. This *shouldn't* be an issue for you, but might be if you really try to push performance.
- See the QuickStart instructions to see how to build and run Simulink models on the hardware. I would start by running the example controller given below.
- 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.
- If you are having trouble getting anything to work on the hardware, the following controller should stabilize the system and have decent performance. Hopefully yours performs better!

$$K = \begin{bmatrix} \frac{1711s + 4910}{s + 50} & \frac{-1557s - 5153}{s + 50} \\ \frac{2432s + 7817}{s + 50} & \frac{921.5s + 3308}{s + 50} \end{bmatrix}$$

Reporting

Compile a single PDF lab report for your team following the provided format. Place the PDF file and all associated files (including all Simulink models, Matlab scripts, and Arduino code that you developed) in a .zip file and upload to Canvas.

MEMORANDUM

TO: Dr. Bedillion
FROM: 24-771 Student Names
DATE: August 29, 2019
RE: Laboratory No. X: Laboratory Name

A paragraph or two here should transmit the laboratory report. You should think of this as the report abstract.

This report has been proofread by all members of the group:

Print Name	Signature and Date
Print Name	Signature and Date
Print Name	Signature and Date
Print Name	Signature and Date

Your "Informal Laboratory Report" should start here. It 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.) A brief description of the goals of the lab exercise, and the equipment and procedure used to achieve those goals. The equipment can be specified once per subsection, i.e. describe the Aero system only once, not *n* times.
2.) The details of all calculations involved in generating your results. Be sure to highlight the main results.
3.) 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. Place in-line links to your Youtube videos for video results.
4.) General discussion. What sense do you make of the results? What can you conclude?
5.) Answers to all of 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?
- Work Distribution: To what specific tasks did each team member contribute? Address this for both the laboratory exercises and the report writing.
- 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.
- Appendix: Attach scans of any hand calculations and copy and paste any Matlab / Arduino code. This is simply for ease of grading; you will also be posting the code files in your .zip file.