Quanser Lab:
System ID and Controller Design

Introduction

Please read this **entire document** before starting with any experiments.

The objective of this design exercise is to reinforce the concepts you have learned in lectures by completing the end-to-end design of a controller on real hardware. This will include:

- Identifying a transfer function model from measurements taken from a real system.
- Designing a controller using the theory you have learned in SS&C.
- Validating the controller in both simulation and on hardware.

The experimental parts of the exercise will be conducted on a Quanser 3DOF helicopter, as illustrated in Figure 1 (a). This hardware was designed as a bench-top representation of a tandem-rotor helicopter (such as the Boeing HC-1B Chinook illustrated in Figure 1 (b)), although I also like to think of it as 'half a quadcopter', and provides a safe and versatile environment for learning the practical aspects of control system design. Multiple previous undergraduate and PhD research projects have been conducted using these systems.

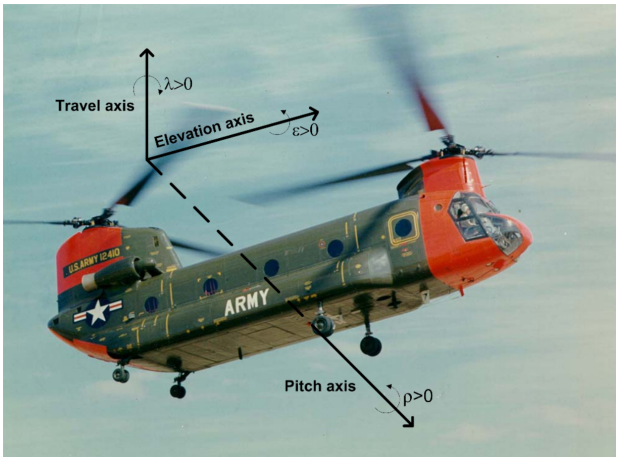


Figure 1: (a) Quanser 3DOF helicopter (b) Boeing HC-1B Chinook

The overall aim of the design exercise is to develop a stabilising controller for the **elevation axis** of the system, as shown in Figure 2.

The design exercise is **formative**: it is intended as a learning exercise and you will not be directly graded on your performance. It is, however, still a *expected* part of the course, and the summative assessment may build on the work you complete.

The task should take you no more than a few hours in total to complete, and you may start immediately.

Images in Figure 1 taken from `quanser.com/products/3-dof-helicopter/` and `chinook-helicopter.com/history/aircraft/A_Models/61-02410/61-02410_a.jpg` on 8/03/22

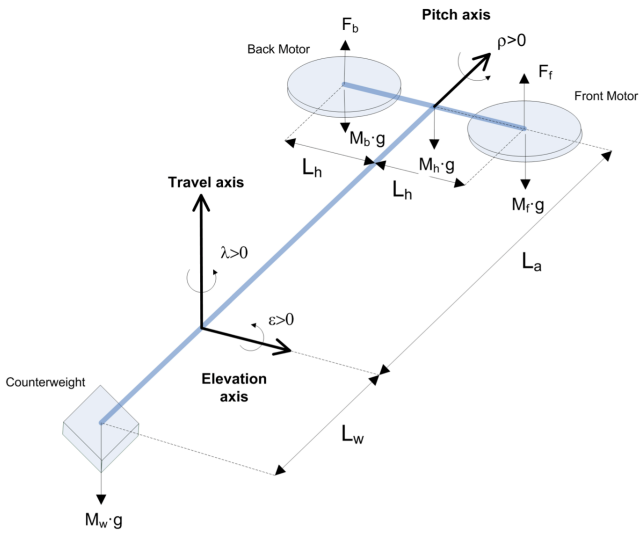


Figure 2: Quanser free-body diagram

Support

The experimental work is to be completed **unsupervised** - all of the information that you require to complete the task is included in this document. Support is, however, available if you encounter a problem.

Support will only be provided until the coursework release date. There are two routes to access support:

- **Technical problems in the lab**

  The lab is managed by the technical services team, and if you have any issues with getting the hardware working please email the tech hub at `engf-tech-hub@bristol.ac.uk`, who will respond ASAP. Please be clear in your email about the problem you are having, including

  – The specific Quanser station you are using.

  – Detailed description of the problem you are having.

  – A copy/paste of any error message that MATLAB prints in the command window.

  I cannot stress this enough: if any of the equipment breaks while you are using it, or if you observe any broken equipment in the lab, **please send an email to to technical services so that it can be sorted as soon as possible**. You will **not** be penalised for accidents that occur during the experiments, and we would like to know if there are any problems as soon as they happen so that we can fix them and maintain lab access.

- **Procedural problems.**

  If you have a problem following the procedure (e.g. if you believe that there is an error in this handout, or a bug in the provided Simulink files) then please put a post on the Teams channel for this lab as soon as possible. As this is a formative exercise you are welcome to also ask questions about exercise more generally, but I will try to 'steer' you in a way that helps you to work the problems out for yourself as opposed to simply telling you how to complete things. Remember to check discussions before posting to see if anyone has

previously had the same problem and if a solution has already been provided*.

Lab Overview

The Lab

The two experimental parts of this exercise require the use of the Quanser 3DOF Heli hardware in **Queen's Building M.003**, beneath QB cafe (entrance requires your university card). There are four Quanser stations in this lab.

Figure 3: Quanser stations in Queen's Building 0.80 (since moved to M.003)

Lab Access

The lab is accessible Monday to Friday, from 9am to 5pm. **Students are not to access the lab outside of these times.** You will need to attend the lab **twice** (once to model the system, and once to implement your controller). You will need to do the first lab **before** the controller design, so you are encouraged to complete it as soon as possible. Each lab should take a maximum of one hour.

You are welcome to use the Labs as much as you need, but please only book one session at a time, and don't block book. The dynamics of each Quanser is slightly different, so **it is recommended that you book the same station for both the first and second experiments**.

There are enough slots for everyone to complete the lab in pairs, but not if you all leave things to the last minute. Please complete the lab as soon as possible.

The Stations

As shown in Figure 4, each of the four stations is equipped with

*I recommend turning on notifications for relevant channels.

1. A desktop computer
2. A monitor
3. A keyboard & mouse
4. A joystick
5. An amplifier
6. A data acquisition board
7. A Quanser helicopter

If any of the above is missing when you arrive at the station (or you observe that it's missing from a neighbouring station), please contact technical services at `engf-tech-hub@bristol.ac.uk` immediately.

Figure 4: Quanser station B.

The Design Problem

The overall workflow for the design exercise is

1. Experiment 1: System Identification
2. Controller Design
   (a) Heuristic PID Tuning
   (b) Pole Placement
   (c) Analytical Validation
3. Experiment 2: Experimental Validation

In the following, the sections that require lab access are highlighted in **blue**. As you need to complete the first experiment before doing the analytical work, it is suggested that you **complete it as soon as possible**.

You are welcome to record your solutions in whichever format suits you best as you complete the exercise. I would, however, suggest using a MATLAB 'live script', and dedicating a single cell to solving each part.

Experiment 1: Step Response

The first part of the exercise is an experiment where you will measure the elevation step response of the Quanser. This will then be used to identify a transfer function model.

A note on safety: You are encouraged to experiment with the hardware, but please please be conscious of both your own safety and the fragility of the Quanser helicopters. Keep you fingers away from the fans, and avoid large step changes to the input variables. If you are not careful, it is easy to crash the helicopter into the 'floor', which is likely to cause significant damage and will delay both your own progress with the exercise and that of your peers. Always use the specified initial control variables, and when powering the device down **always** have a hand ready to catch the helicopter as it descends. This is demonstrated in an accompanying video.

If you need to stop the hardware in an emergency, just turn the amplifier off.

Pre-experiment actions: There is currently an issue with the MATLAB installation on the workstations, and if you immediately open MATLAB you will likely observe several errors. When you arrive at the workstation you will first need to copy the `pathdef.m` file from the supplied files to the folder

`C:\Users\yourUserID\OneDrive - University of Bristol\Documents\MATLAB`

If this folder does not exist, you will first need to create it. Note that you will also need to remove this file again when you use MATLAB on other University computers.

When opening MATLAB/Simulink files, make sure to first open MATLAB and then open files from the MATLAB window. If you try to open MATLAB/Simulink files without MATLAB already open, it will possibly cause errors.

Experimental Procedure

The objective of the experiment is to record the open-loop **step response** of the Quanser's **elevation axis**. The experimental procedure is as follows:

1. The amplifier should be off when you arrive at the workstation. If it is off, turn it on using the switch illustrated in Figure 5; if it is on, turn it off and then on again. You should be able to clearly hear the amplifier's fan when it is on.
2. Open MATLAB, and then open the `m_part1.slx` Simulink file (I have found that Quanser C asks if you would like to open a project or model; **choose model**). The Simulation block diagram contains two parameters that can be adjusted by the user: the values in *Yaw Demand* and *Elevation Input*, highlighted in pink. The *Yaw Demand* can be used to tune the yaw position of the Quanser, but should be left constant during the experiments; the Elevation Input is the open-loop input that you will be designing a controller for. Leave all other parameters and blocks/linkages as they are supplied to you throughout the experiment.
3. Build the model with `Hardware` $\rightarrow$ `Build, Deploy & Start` $\rightarrow$ `Build`
4. Make sure that the control variables are set at *Yaw Demand* $= 0$ and *Elevation Input* $= 0$.
5. Start the experiment. The encoder for the elevation angle is zeroed at its position when the Simulink file starts, so you need to hold the Quanser in the horizontal position before you start it (this is demonstrated in the accompanying video). The stationary point of each Quanser is slightly different, so it may take a few attempts to start each one in the correct position. You can start the Simulink file with `Hardware` $\rightarrow$ `Monitor & Tune`. MATLAB will start recording both the input and the output of the system from the moment the Simulink file starts running.
6. If necessary, modify the *Yaw Demand* until the Quanser is in front of you. This input is measured in degrees, and you should only apply changes of $\pm 90^\circ$ at a time (more than this shouldn't be necessary anyway).
7. Change the *Elevation Input* to 2 and observe the response. The states of the system can be observed using the preprogrammed scopes.
    - Keep an eye on the hardware and be ready to catch it if you accidentally introduce a very large demand input.
8. The Simulink file will automatically terminate at 100s. When it does, the fans will automatically cut out, and you will need to be ready to catch it to stop it crashing into the ground. You can also terminate the program early with `Hardware` $\rightarrow$ `Stop`. When the program terminates, the recordings of the *Elevation Input* and *Elevation Output* will automatically be transferred to the workspace, and can then be saved with `s_save.m`. Take a look at the measurements you have taken using `s_plot.m`, and determine whether they are suitable for fitting a transfer function model's step response - if not, delete your data, run the commands `clear` and `clc`, and then return to step 4 and repeat the experiment. Make

sure you have saved suitable data **before** leaving the lab, as you might have to wait a while at busy times to be able to re-complete the experiment.

9. When you are finished, turn off the amplifier and **log off** of the computer (don't turn it off).

Figure 5: Amplifier power switch.

System Identification

Note that this analysis doesn't need to be completed in the lab.

1. Using the data you measured in the experiment, fit both 1st and 2nd order transfer function models data and determine the standard parameters for each. The transfer functions can be fitted both analytically and through trial-and-error (you may find the `tfest` function helpful).
2. Compare the simulated step response of the estimated models to the experimental data (a single plot showing all three is recommended).
3. Determine the poles of the estimated transfer functions, and validate that they match the simulated step response.
4. What are the limitations of identifying the system this way? Can you think of a better way of identifying the system's dynamics?

Control Theory

PID Investigation

The analysis in this section is to be performed in simulation (using e.g. MATLAB) with the transfer function you identified in the previous section that best fit the data (it should be the second order one - if your first order one produces a better fit you've messed up somewhere...).

1. Implement a proportional negative-feedback controller (in either Simulink or MATLAB). Using step responses, show the effect of the proportional term by varying the controller gain. Compare the findings with those that are predicted by the root locus method.
2. Add a derivative gain to the controller, and investigate the effect of the derivative term on the step response.
3. Remove the derivative gain, add an integral gain, and show the effect of the integral term on the step response.
4. Describe in your own words the contribution of the P, I, and D terms.
5. Reinstate the derivative gain, and manually tune the gains to achieve a step response that you would consider 'satisfactory'.
   (a) Why do you consider the tuned performance to be satisfactory?
   (b) What characteristics of the real system did you consider when you were tuning the controller?
6. Implement the controller
$$C(s) = \frac{25s^2 + 200s + 1}{s}$$
   (a) Is the predicted performance good?
   (b) Is it realistic? What elements of the real system are ignored in your simulated model, and how might the performance change if you were to implement this controller on a real system?

Controller Design

1. Using the theory you have learned in lectures, **design** a controller for your 'best' transfer function model to meet the minimum step response performance objectives shown in Table 1. You can use any controller structure that has been covered in the course, but it's suggested that you start with a proportional controller then add extra terms to address its limitations.
2. What limitations have you found when designing a controller using the methods covered in the lectures?

Table 1: Minimum closed-loop design requirements.

Performance Measure	Value
Overshoot	10 %
Settling time	5 s
Steady-state error	0 %

Analytical Validation

1. For the block diagram illustrated in Figure 6,
   (a) Generate the Bode plot of the open-loop transfer function $C(s)G(s)$.

(b) Making reference to the sensitivity and complementary sensitivity functions, use the Bode plot of $C(s)G(s)$ to infer the output of the system $y(t)$ for the following inputs

i. $r(t) = 0$, $d(t) = \cos(t)$, $n(t) = 0$.

ii. $r(t) = 0$, $d(t) = 0$, $n(t) = \cos(10,0000t)$.

(c) How could your controller be modified to improve it's robustness to these inputs?

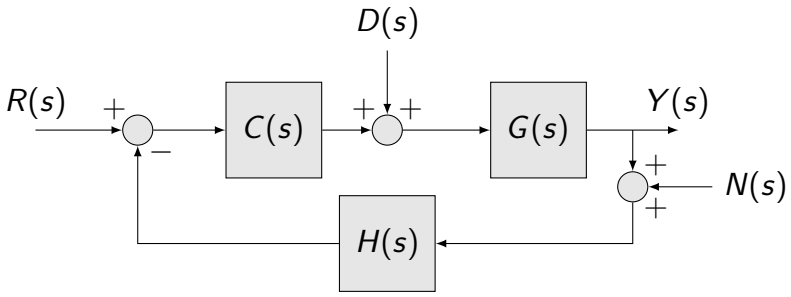


Figure 6: Closed-loop block diagram, where $G(s)$ is the transfer function you identified in Part 1, $C(s)$ is the controller you **designed** in Part 2, and $H(s) = 1$.

2. Assume that the process in Part 1 has been completed by your peers on two of the other Quanser stations, in which they have identified the following two transfer functions for the Quanser helicopter:

$$G_1(s) = \frac{5.089}{s^2 + 0.05821s + 1.131} \quad \text{and} \quad G_2(s) = \frac{12.11}{s^2 + 0.2013s + 3.289}.$$

(a) Determine the step response of these two transfer functions in closed-loop using the controller you designed in Part 2 and compare with the closed-loop step response obtained using your identified model.

(b) For these transfer functions, does your controller still meet the design requirements in Table 1? Discuss why/why not, making reference to the Bode plot of $C(s)G(s)$ and the sensitivity function.

(c) How could you modify the controller to improve it's robustness to parameter variations?

Experiment 2: Experimental Validation

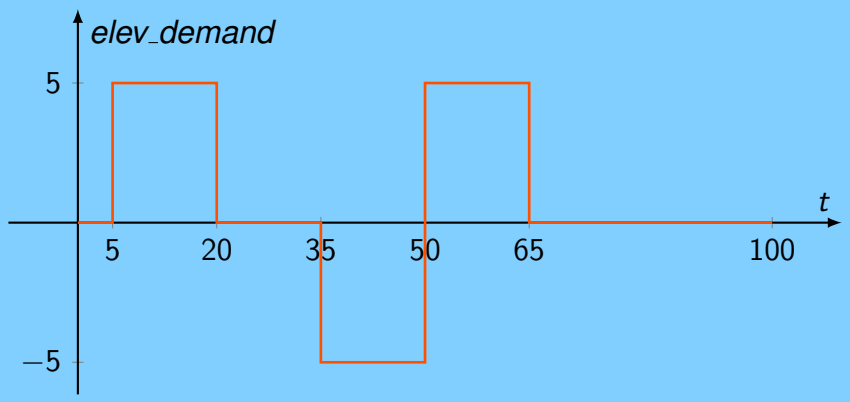
The final part of the exercise is a an experiment where you will implement your controller in closed-loop. Remember to complete the same 'pre-experiment actions' specified in Part 1 before attempting to use the hardware, and remember to book the same Quanser station for the second experiment as the first (each of the Quanser's have slightly different dynamics).

Experimental Procedure

1. Turn the amplifier on (or off and then on, if it is already on when you arrive in the lab).
2. Open MATLAB, and then open the `m_part3.slx` Simulink file. The block diagram contains a *Controller* block where you are to implement your controller to control the output *elev_output*

to meet the demand *elev_demand* using the input *elev_input*. Leave all other parameters and block/linkages as they are supplied to you throughout the experiment.

3. Build the model with `Hardware` → `Build, Deploy & Start` → `Build`.
4. Hold the Quanser in a horizontal position (as you did in Part 1), and start the Simulink file with `Hardware` → `Monitor & Tune` (again remember that you will need to catch the Quanser when the Simulink file terminates at 100s, or if you prematurely stop it). The Simulink file will apply to following demand input to your controller:



Record the response of the controlled system for this demand. It might take a few attempts to get good data (check that you can use it to respond to the following points). The data recorded by Simulink can be saved with `s_save3.m` and then plotted with `s_plot3.m`.

5. If the performance of your controller is poor, manually re-tune it until you get a satisfactory response, and then re-record the data.
   - If your controller contains a derivative term, you will need to put a low-pass filter before the differentiator block (otherwise the differentiator will amplify any noise or step signals in the feedback look). This is automatically implemented in the Simulink PID block. When using a low-pass pre-filter you will need to choose the cut-off frequency - this can either be done analytically using the methods you have learned in lectures, or heuristically (the cut-off frequency needs to be high enough to not interact with any of the system or controller dynamics).
6. Also try implementing a PID block with the parameters - this should work if all else fails!

$$k_i = 15.3, \quad k_p = 23.8, \quad k_d = 8.5, \quad N = 100.$$

7. When you are finished, turn off the amplifier and **log off** of the computer (don't turn it off).

Validation Procedure

1. Plot a figure that shows:
   (a) The demand.
   (b) The predicted output of the transfer function model in closed-loop with the controller

you used in the experiment.

(c) The output of the system you measured in the experiment.

2. How does the performance of the real system deviate from that predicted by the transfer function model? Which elements of the Quanser's dynamics are not modelled by the transfer function? How can they be used to explain any observed discrepancy in the results?