QArm Lab Procedure
Low Level Control

Setup

1. Launch Quanser Interactive Labs and load the QArm Workspace.
2. Launch MATLAB and browse to the working directory for Lab 1 – Low Level Control.

Exploration

Part 1

1. Open the Simulink model LowLevelControlPWMMode.slx (Figure 1). For this section of the lab you will use the switches and gains in the Exploration Box to select the setpoints for the base, shoulder, elbow and wrist motors. With varying setpoints, you will tune the proportional, integral, and derivative (PID) terms of the joint controllers to examine their effect on performance characteristics mentioned in the Concept Review.

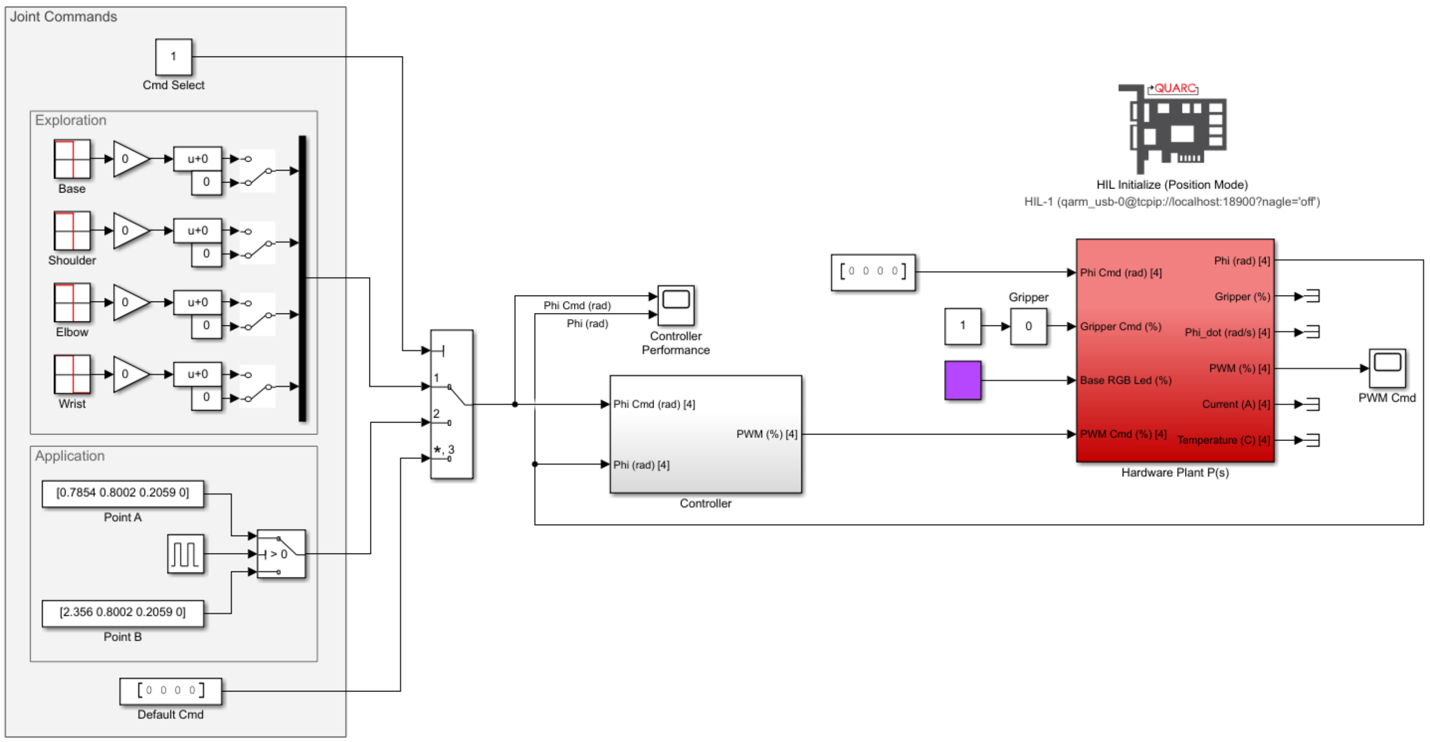

Figure 1: Simulink model that controls the QArm in PWM Mode.

2. Before making any changes to the model, open the model's Configuration Parameters and verify that they are configured as follows:
   a. Solver type: Fixed-step
   b. Solver: ode4 (Runge-Kutta)
   c. Fixed-step size (fundamental sample time): 500 Hz

3. Now that Simulink is configured correctly, you will need to build a proportional controller, so that you have a closed-loop feedback system. Open the Controller subsystem and connect the blocks according to Figure 2.2, ignoring the integrator, derivative and 3 input sum blocks. Also, ensure that the Cmd Select constant at the root level of your Simulink model is set to 1.

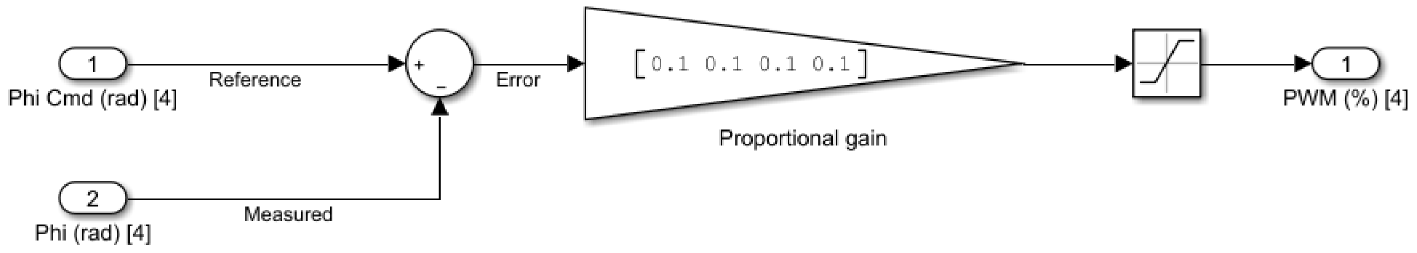

Figure 2.2 Proportional Controller

4. Run the model using the green Play button under the Simulation Tab of your model. Once started, the model will command 0 rad angles to all four of the manipulator's joints.

5. Keep the proportional gain matrix K_p at $[0.1\ 0.1\ 0.1\ 0.1]$ (it is a 1x4, one gain for each joint). To apply a pulse signal to a joint you will need to toggle one of the switches in the Exploration Box to be 'ON'. Start by toggling the Base joint. Set the gain block after the Base signal generator to 1 and leave the bias block set to zero for now. The pulse input applied to the arm oscillates between $\frac{\pi}{4}$ and $-\frac{\pi}{4}$ ($+45$ and -45 degrees). You can use the Controller Performance scope to monitor the arm's response, and you will also be able to see the physical arm responds. You can turn **off** all plotted lines **apart from** *'Phi Cmd (rad) 1'* and *'Phi (rad) 1'* by clicking each coloured line and you should see the response shown in Figure 2.3.

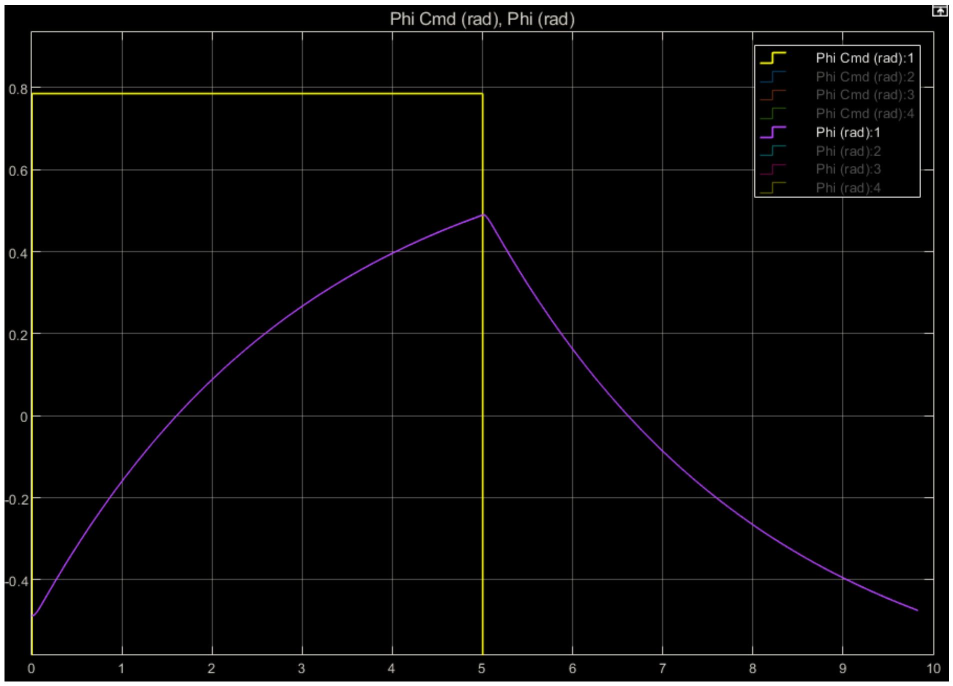

Figure 2.3 Plot of QArm response to pulse input with proportional gain of [0.1 0.1 0.1 0.1]

6. While the model is running, change K_p to $[\mathbf{0.2}\ 0.1\ 0.1\ 0.1]$ and observe how the response of QArm changes. What is the rise time? Does the arm have enough time to reach the desired position before the next command? Make a note of the performance changes that you notice. Now set K_p to

$[\mathbf{0.5}\ 0.1\ 0.1\ 0.1]$ and answer the same questions as before. Take note of a steady-state error with this higher gain if it is present.

7. How do you think the arm's response will change if you set K_p to $[\mathbf{1}\ 0.1\ 0.1\ 0.1]$? What should the controller's output be for a $\frac{\pi}{2}$ error? Take notes. Change K_p to $[\mathbf{1}\ 0.1\ 0.1\ 0.1]$ and open both the Controller Performance scope and the PWM CMD scope side by side. Were your assumptions from the previous step correct? Take note of the rise time and steady-state error. Zoom into the Controller Performance plot, is there any steady-state error? How does changing K_p affect the controller output? Could you characterise the relationship with an equation?

8. Now change the first value of the K_p to **2**, **5** and then **10**, does the controller's performance improve? If so, by how much? Answer the same questions as before based on the results from the plot and the performance of the arm. Ensure you have a good understanding of the effect that increasing the K_p of the Base joint controller has on the response of the arm. Note that the PWM CMD plot shows a response that is bounded between -1 and 1. Why is this? Can you identify the Simulink block in the controller that is responsible for this effect?

9. Use a similar method to the one outlined in the steps above to make a PD controller by duplicating the proportional gain block inside the Controller subsystem (you will also need to include the derivative block after the gain, and add its output to that of the Proportional gain using a sum block). Start with a derivative gain K_d of $[0\ 0\ 0\ 0]$ and **increase** the first term until any overshoot disappears and your rise time is less than 1.5s.

10. It is unlikely that you will need an integral gain as there is very little steady-state error to account for, but you can create a PID controller if you wish and see how the performance changes, if at all.

11. Now that you have at least a PD controller for the Base joint, you will need to select gains for the other three joints. It is best to do this sequentially, i.e. examining the pulse response of the shoulder, by disabling the switch on the Base pulse generator and then enabling the one on the shoulder. Repeat this for the elbow, and finally the wrist. Due to the natural damping effects, the derivative gains for all the joints should be low ($0 < Kd < 0.5$) and *remarkably similar.* The correct proportional gains are also within 0.5 decimal points of each other.

12. Ensure that the biases for the base, shoulder, elbow, and wrist in the exploration section are set to 0. Enable all the switches and set the gains on the signal generators to 1 to enable them. All the joints should move together. Take a screenshot of the manipulator's response for your records.

Part 2

1. To implement the three scenarios discussed in the Concept Review you will need to use the bias blocks to set a midpoint for the arm to oscillate about, as well as the gain blocks to enable the oscillations with a value of 1. Use Table 2.1 below to recreate each scenario with the arm. For each configuration make a note of the overshoot using the Controller Performance scope and take a screenshot of the general shape. The gain and bias for the Base joint and Wrist should be 0.

Table 1 Parameters for scenarios from Concept Review

Scenario	Shoulder joint	Elbow joint
1	Gain = 1, Bias = 0	Gain = 0, Bias = $-\pi/2$
2	Gain = 0.5, Bias = $\pi/6$	Gain = 0, Bias = $-\pi/2$
3	Gain = 0.75, Bias = 0	Gain = 0, Bias = 0

Reflection

1. Once all your controllers are tuned, set the CMD Select constant to 2 for this section. Your model will now be taking input from the blocks in the Application Box.
2. The end-effector moves between two unique positions that are defined by the joint angles labelled as 'Point A' and 'Point B'. However, if you examine these two joint commands, you will see that only the angle for the Base joint is changing.
3. Examine the path that the end effector takes between the two points. If the end-effector were a pen drawing on a whiteboard, what shape would it draw? Is there a way to get the arm to move between the same two points but following a linear path? What else would have to change for that to be possible?
4. Take a screenshot of both the Controller Performance and PWM CMD scopes for one full cycle of motion. Keep these for your records. We will explore these in the Navigate section of the Trajectory Generation lab.
5. Stop the model and close it.

Comparison

1. Open the Simulink model titled LowLevelControlPositionMode.slx.
2. Before making any changes to the model, open the model's Configuration Parameters and verify that they are configured as follows:
   a. Solver type: Fixed-step
   b. Solver: ode4 (Runge-Kutta)
   c. Fixed-step size (fundamental sample time): 500 Hz
3. Run the model using the green Play button ▶ under the Simulation Tab of your model. Once started, the model will command 0 rad angles to all four of the manipulator's joints.
4. Set the CMD Select constant to 2 and run the model. Compare the performance of the controller that you designed in the PWM mode example versus the controller used in the Position mode model. Take note of all the important performance characteristics.
5. If no further experiment is required, set CMD Select to 3 and ensure the default command values are [0 0 0 0].
6. Stop the model and close Quanser Interactive Labs.