QArm Lab Procedure
Trajectory Generation

Setup

1. It is recommended that you run this lab individually.
2. Find an object that is roughly cuboidal in shape, with a length dimension under 5 cm and weight under 100 grams (for example, a piece of Styrofoam). This will be used during the pick and place task.
3. Move the QArm manipulator to the home position and turn ON the unit using the power switch located on the rear side of the base. Once powered, the manipulator should hold this position.
4. Launch MATLAB and browse to the working directory for Lab 5 – Trajectory Generation.

Navigate

1. Open the Simulink model Navigate.slx. You will use this model to implement a simple trajectory generator. the inverse kinematics of the Quanser QArm manipulator.
2. Replace the Inverse Kinematics and Forward Kinematics functions with the completed versions from previous labs.
3. Open the function called cubicSpline. This function contains the incomplete code for the four coefficients given an initial and final setpoint. Complete lines 4, 5, 6 and 7. Note that each of the four coefficients is a 3x1 vector, representing the x, y and z splines. Base your result on equation 6 from the Trajectory Generation Concept Review for zero speeds, that is, $\dot{x}_0 = \dot{x}_f = 0$. Close the function when complete.
4. Open the function timing. Set line 4 to keep the time-constrained between 0 and T. HINT: Use the $\mathbf{mod}$ function in MATLAB. Try $\mathbf{mod}(5,2)$ and $\mathbf{mod}(9,5)$ in the command window.
5. Complete line 7 to set the 4x1 time vector to output $1, t, t^2$ and t^3.
6. Note that the two position waypoints $[0.25\ 0.25\ 0.10]^T$ and $[-0.25\ 0.25\ 0.10]^T$ are similar to the joint space waypoints you used in the Low Level Control lab application. Note that T is set to 5 seconds, and it will take 5 seconds to traverse between the points (10 seconds for a complete cycle).
7. Prior to running the model, open the model's Configuration Parameters and verify that they are configured as follows:
   i. Solver type: Fixed-step
   ii. Solver: ode4 (Runge-Kutta)
   iii. Fixed-step size (fundamental sample time): 500 Hz
8. Ensure that the workspace around the manipulator is clear of obstacles and stay well outside the reach of the manipulator.
9. Build and deploy the model using the Monitor & Tune action. Once started, the model will actuate the manipulator to traverse between the two waypoints. Does the manipulator follow a circular arc between the two points or a straight path? Take notes.

10. Open the Tracking scope. Identify the desired setpoints, the desired task space trajectory being passed to the inverse kinematics module, as well as the measured position out of the forward kinematics module. Comment on the performance and take a screenshot of one complete cycle.

11. Stop the model.

Welding

1. Open the Simulink model Welding.slx. Copy over the cubicSpline function that you completed in the Navigate.slx model and copy over your qarmForwardKinematics and qarmInverseKinematics functions from previous labs. The function waypointNavigator has been completed for you. Comment on what this function is accomplishing.

2. Right-click on a blank space in your model and open the Model Properties. Browse to the **callbacks section**, and then to the **InitFcn** section. Set and apply the, A, B, C, D and default setpoints to the following,

$$A = [0.25 \quad 0.25 \quad 0.1]^T$$

$$B = [0.25 \quad 0.50 \quad 0.1]^T$$

$$C = [-0.25 \quad 0.50 \quad 0.1]^T$$

$$D = [-0.25 \quad 0.25 \quad 0.1]^T$$

$$default = [0.25 \quad 0.25 \quad 0.2]^T$$

3. Set the waypoints to the correct order, that is, default, followed by A, then, B, C, D, back to A. Press **Apply** and **Okay** to close the Model Properties menu.

4. Prior to running the model, open the model's Configuration Parameters and verify that they are configured as expected in the Navigate section.

5. Build and deploy the model using the Monitor & Tune action. Once started, the model will actuate the manipulator to traverse between the waypoints. Does the application weld as expected from the Lead Through laboratory? Take notes on the speed of welding between A to B, and B to C etc.

6. Repeat this experiment with the duration set to 2s. Is the manipulator able to weld correctly? Why? Take notes.

7. Stop the model after a few cycles.

Assembly

1. Open the Simulink model Assembly.slx. Copy over the cubicSpline, qarmForwardKinematics and qarmInverseKinematics functions from the Welding.slx lab. You will notice that the model is the same.

2. Open the Model Properties menu to access the **callbacks** tab and **InitFcn** section. Set the positions A, B, C and D to the same ones from the welding application. Set default to,
$$default = [0.00 \quad 0.375 \quad 0.2]^T$$

3. Set the waypoints to be default, then A, then A again (representing the time it would take for the manipulator to take action at this location), back to default, then B, then B again, then default, C, C, default, D, D. Press **Okay.**

4. Build and deploy the model using the Monitor & Tune action. Once started, the model will actuate the manipulator to traverse between the waypoints. Does the manipulator move as expected?

5. Take notes on any differences and/or any screenshots as you see fit.

6. Stop the model.

Pick and Place

1. Open the Simulink model PickAndPlace.slx. Copy over the cubicSpline, qarmForwardKinematics and qarmInverseKinematics functions from the Welding.slx lab. You will notice that the model is still the same, except for an added function actuateGripper.

2. Open the Model Properties menu to access the **callbacks** tab and **InitFcn** section. Set the **pick** position to a desired **positive x** and **negative y** value (pick x and y values such that the combined magnitude $\sqrt{x^2 + y^2}$ is larger than 0.25 m to avoid proximity and collision with the base) and a z value of 0.03 m (this is to avoid contact with a table).

3. Set the **pick_high** position to be the same as the **pick** position, but with a z component of 0.15 m instead.

4. Set the **place** position to a desired **positive x** value and **positive y** value with a z component of 0.15 m as well.

5. Set the waypoints in sequence to **pick_high**, **pick**, **pick_high**, **home**, **place**, and then **home** again. Why is this particular sequence selected?

6. In the model, open the function actuateGripper. This implements a simple state machine to check if you are close to a desired setpoint and correspondingly open or close the gripper. At line 10, set the threshold to 0.01 m. This implies that this state machine will consider the manipulator 'at' the setpoint if within 1 cm of it. You may want to relax this threshold later if your manipulator fails to identify arrival.

7. At line 13, replace **condition** with code that correctly implements the logic to actuate the gripper to the close position when close to the **pick** setpoint. When this condition is true, you want **gripper_state** at line 14 to be 1.

8. At line 18, replace **condition** with code that correctly implements the logic to actuate the gripper to the open position when close to the **place** setpoint. When this condition is true, you want **gripper_state** at line 19 to be 0.

9. With a completely clear workspace (a dry run without any object), build and deploy the model using the Monitor & Tune action button. Once started, the model will actuate the manipulator to traverse between the waypoints. Does the manipulator move and actuate the gripper as expected?

10. Repeat the previous step by placing the object you selected at the pick position before the manipulator arrives there. Does it pick the object and then drop it off at the place position correctly?

11. Stop the model, and close all models

12. Power OFF the manipulator using the switch at the rear end of the base and bring it back to rest position. Close MATLAB.