QArm Lab Procedure
Singularity Avoidance

Setup

1. Launch Quanser Interactive Labs and load the QArm Workspace.
2. Launch MATLAB and browse to the working directory for Lab 8 – Singularity Avoidance.

Trajectory Navigation

You will begin this lab by commanding the end-effector to follow a linear trajectory based on cubic splines. The goal is to compare the behavior of the manipulator as it navigates trajectories in the vicinity of a singularity point.

1. Open the Simulink model Navigate.slx (Figure 1).
2. Replace the cubic spline, forward kinematic and inverse kinematics MATLAB functions in the Simulink model with the same functions that you developed in the previous labs.

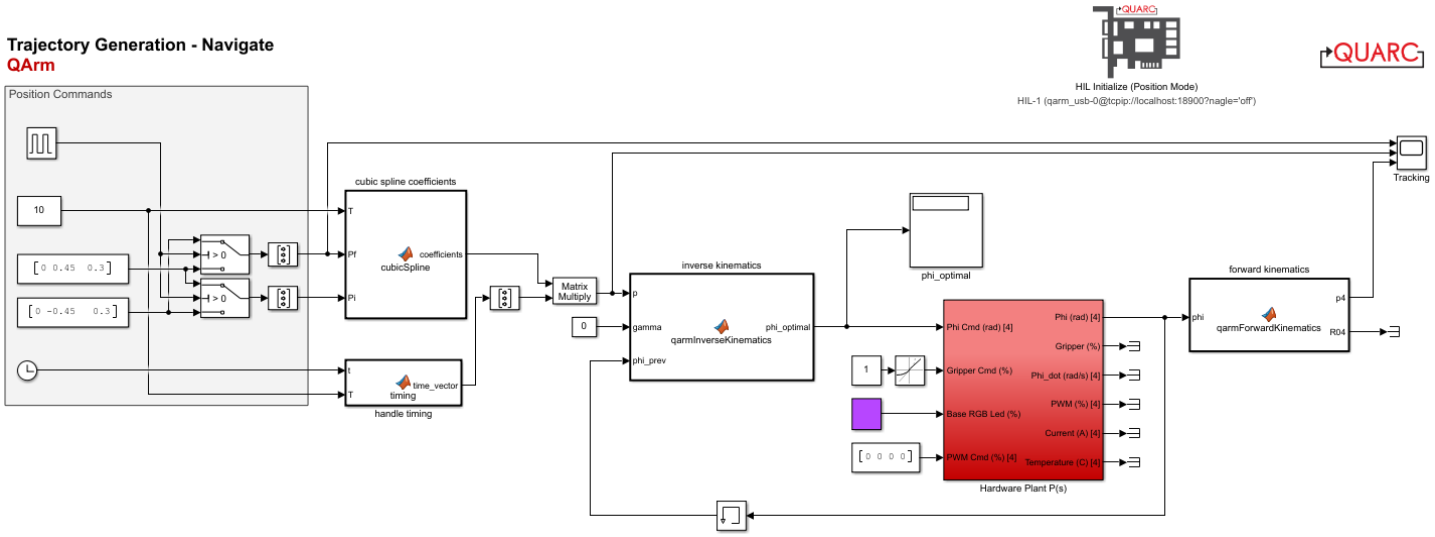

Figure 1 - Simulink model that generates and commands a trajectory based on cubic splines.

3. Run the model using the green Play button  under the Simulation Tab of your model.
4. The model generates and commands a cubic spline trajectory between setpoints [0.1, 0.45, 0.3] and [0.1, -0.45, 0.3]. Since the coordinates of the two endpoints have the same X and Z values but opposing Y values, the end-effector will follow a trajectory parallel to the Y axis.
5. Open the scope labeled Tracking. Sample results are shown in Figure 2. The scope displays the desired, commanded, and measured X, Y, Z values all in meters. The results indicate that when a change in the Y axis setpoint occurs, the end-effector initially accelerates along the Y axis as it starts to follow the spline, then travels at a constant velocity, before decelerating as it reaches the setpoint. Was the manipulator successfully able to traverse the entire trajectory?

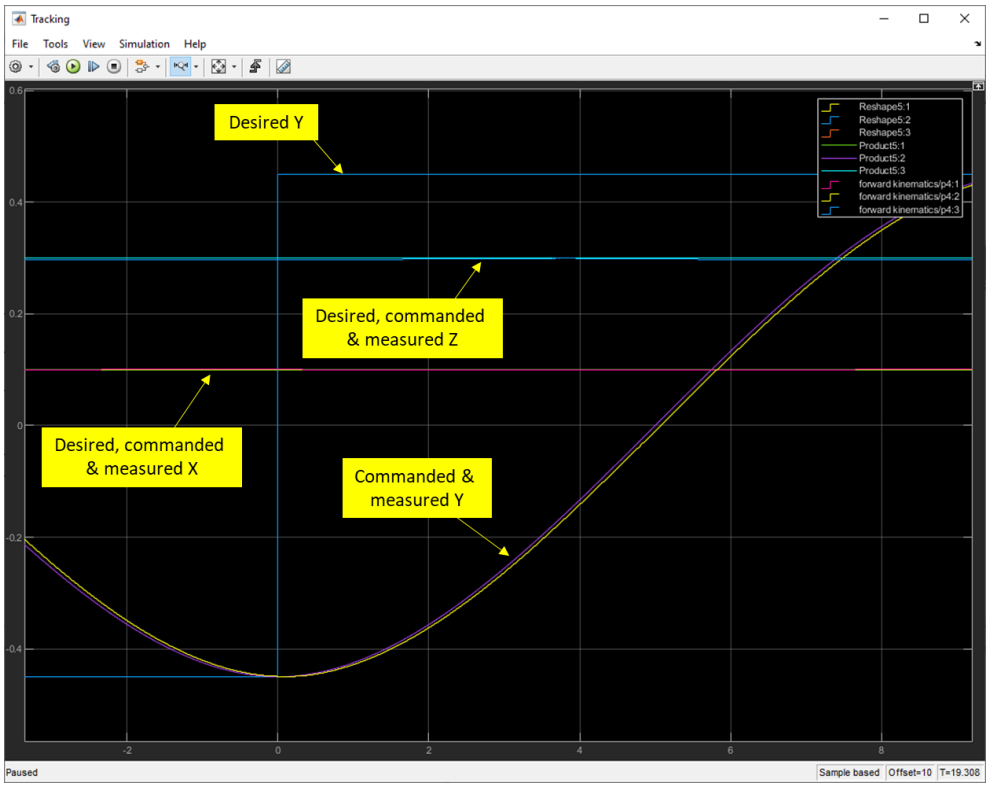

Figure 2 - Sample results showing the manipulator following a spline

6. Stop the model, and reduce the X values of the desired end-point positions from 0.1 m to 0.05 m. Deploy the model again and take screenshots of the tracking scope through one full cycle of motion between the two points. Does the X and Y tracking performance degrade? Why? Add a scope to the **Phi_dot (rad/s) [4]** output of the **Hardware Plant P(s)** and monitor the speed of the base joint as you decrease the X values from 0.1, to 0.5.

7. Finally, set the X values of the desired end-point positions to 0 m. Observe the manipular's behaviour as the commanded trajectory falls on the Y axis, causing the end-effector to pass through a singularity. Take necessary screenshots of your results and think about what prevents the manipulator to successfully track the commanded trajectory (Hint: examine the optimal phi values calculated by the inverse kinematic block).

8. Stop and close the model.

Singularity Avoidance

1. Open the Simulink model SingularityAvoidance.slx (Figure 3).

2. Replace the cubic spline, forward kinematic and inverse kinematics, and differential kinematic MATLAB functions in the Simulink model with the same functions that you developed in the previous labs.

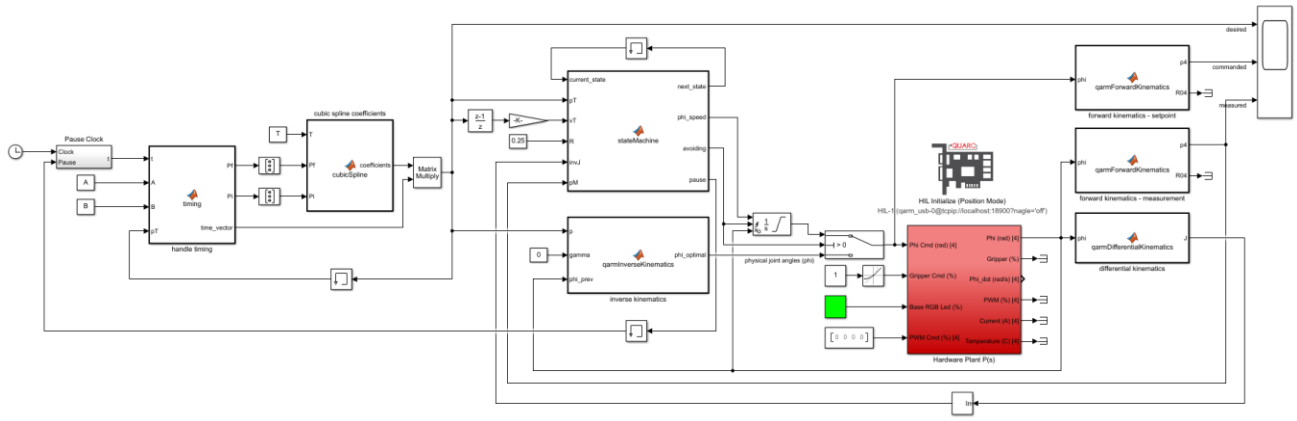

Figure 3: Simulink model that implements a singularity avoidance algorithm using a finite state machine

3. Open the stateMachine MATLAB function. Using the Concept Review as a guide, complete the script to implement the finite state machine shown in Figure 3. You are required to complete the State Transition and State Action sections for states 1, 2, 3, and 4, namely assuming the appropriate values to the following variables: `next_state`, `phi_speed`, and `avoiding`. Recall that the state machine will determine if the target position (pT) falls within the singularity avoidance region; if it does an alternative trajectory along the surface of the singularity avoidance cylinder is commanded to the end-effector otherwise the original cubic spline trajectory (P) is commanded. In the function, by setting the variable `avoiding` to 1, you will force the end effector to follow the perimeter of the avoidance region, otherwise setting it to 0 commands the linear trajectory.

4. Once you have completed the state machine, open Model Properties and select the Callbacks tab. Under InitFcn, set A to $[0;\ 0.45;\ 0.3]$ and B to $[\ 0;\ -0.45;\ 0.3]$. Comment all other A and B values using the % symbol. This will command a spline trajectory with endpoints A and B along the Y axis.

5. Back in the Simulink model, set the radius of the singularity avoidance cylinder by setting the constant wired to the input port of the stateMachine script labeled R to a value of 0.25 m.

6. Run the model using the green Play button  under the Simulation Tab of your model. Why does the manipulator seem to jump discontinuously when exiting the avoidance volume? Stop the model, open the state machine and uncomment lines 51 to 53, which pause the linear trajectory generator if it has arrived at the exit point while the physical manipulator hasn't. Deploy the model and comment on the difference.

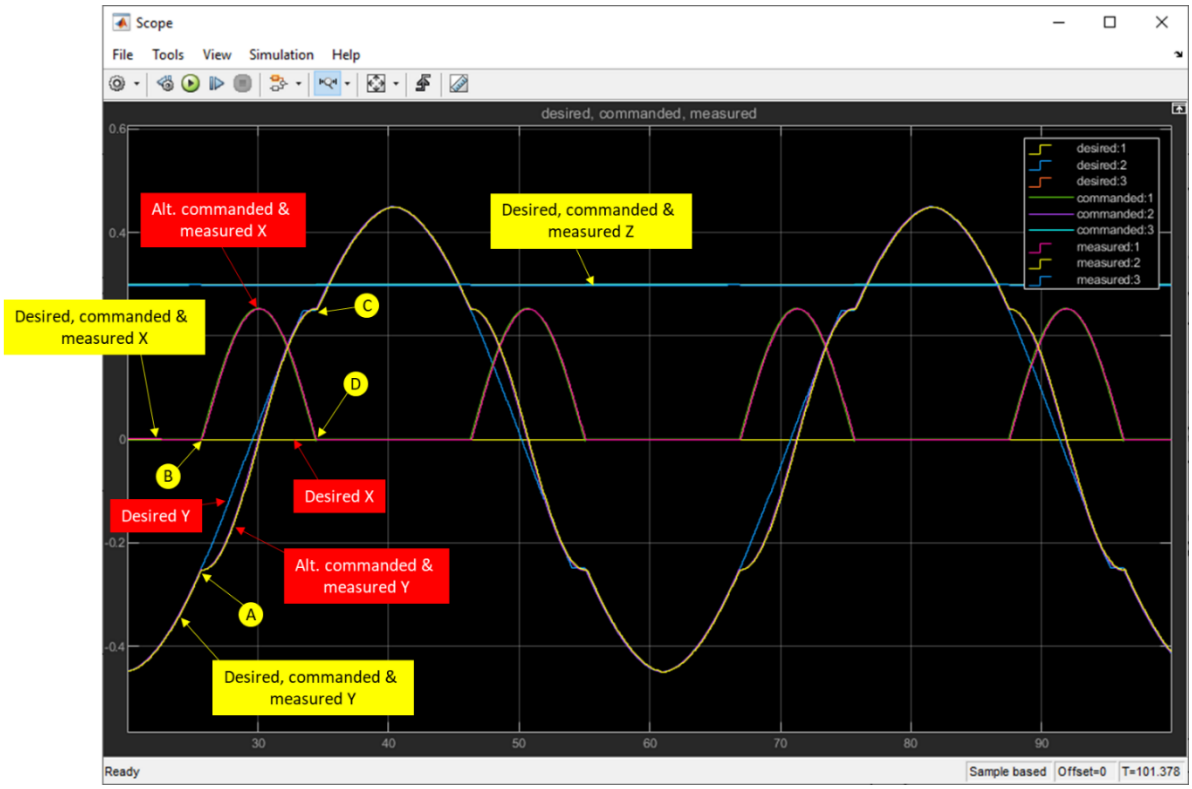

Figure 4: Sample results showing the desired, commanded, and measured trajectories

7. When the model runs, the end-effector will initially move towards the first point. Then at each sample time the state machine will determine if the end-effector should follow the original commanded spline trajectory towards the other endpoint or follow the alternative trajectory along the surface of the singularity avoidance cylinder. Sample results are shown in Figure 2.4. Take a moment to observe the desired, commanded, and measured trajectories along the X, Y, and Z axes. The results indicate acceptable tracking along the Z axis. Tracking along the X and Y axes are also acceptable up to the t = 25 s mark, at which point the end-effector approaches the singularity avoidance region. Here, the state machine switches state and commands the singularity avoidance trajectory instead. Label B indicates the moment at which the end-effector's X coordinate starts tracking the singularity avoidance cylinder with a radius of 0.25 m. Label D indicates the moment when the end-effector reaches the end of the singularity avoidance cylinder and continues to track the originally commanded cubic spline along the X axis. Labels A and C on in Figure 2.4 represent the same as B and D, respectively, for the Y trajectory.

8. Stop the model.

9. Once again open Model Properties and select the Callbacks tab. Under InitFcn select the second set of A and B points and repeat the previous step. What is different between this set of points and the previous? Make note of your observations and take necessary screenshots of your results.

10. Repeat the previous step with the third set of A and B points. What is different between this set of points and the first? Make note of your observations and take necessary screenshots of your results.

11. Repeat the previous step with the last set of A and B points. What is different between this set of points and the first? Make note of your observations and take necessary screenshots of your results.

12. Stop the model.