QArm Lab Procedure
Teach Pendant

Setup

1. Please make sure the QArm is placed on a completely flat surface with no other objects inside this working space.
2. 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.
3. Launch MATLAB and browse to the working directory for Lab 4 – Teach Pendant.

Inverse Kinematics

1. Open the Simulink model InverseKinematics.slx (Figure 1). You will use the model to implement the inverse kinematics of the Quanser QArm manipulator. When implemented correctly, the inverse kinematics calculate four sets of possible, but not always valid, static joint configurations for a given end-effector position.

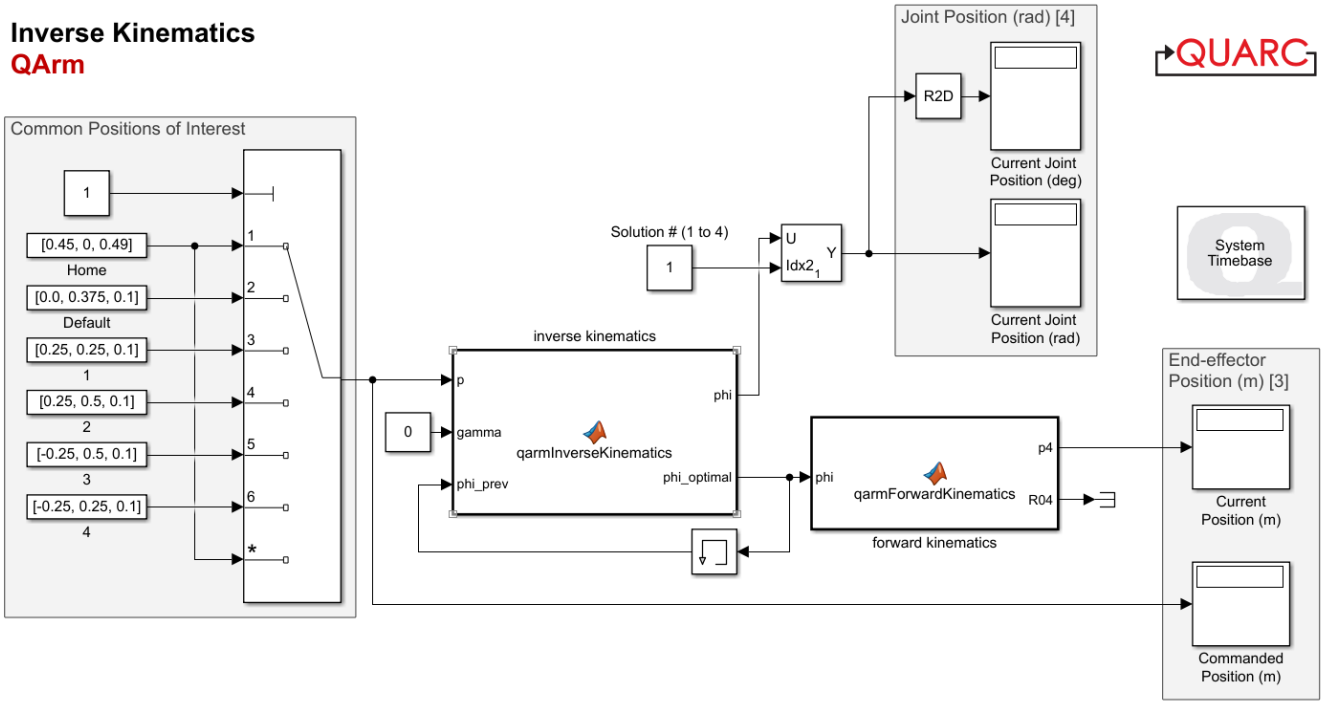

Figure 1: Simulink model that implements the QArm inverse kinematics

2. Replace the forward kinematic MATLAB function in the above model with the function that you developed in the previous lab.
3. Open the function called qarmInverseKinematics. This function contains the incomplete inverse kinematics of the manipulator. Complete lines 27, 30, 31 and 32 based on the constant definitions you used in your completed qarmForwardKinematics function.

4. Complete lines 47 to 52 to correctly define the parameters A, C, H, D1, D2 and F based on the equations provided to you in the Teach Pendant Concept Review. Note that D1 and D2 corresponding to the 2 possible D values.

5. Lines 54 to 57 have been completed for you. These calculate the 2 possible solutions for θ_3 and then duplicate them to prepare for the 4 overall solutions. Complete lines 59 to 63 based on the first solution for θ_3 (using `theta(3,1)` in your code) and the equations provided to you in the Teach Pendant Concept Review. Once complete, repeat this for lines 65-69, 71-75 and 77-81 based on the 2nd, 3rd, and 4th solutions for θ_3. Also note that lines 59 to 69 should use D1, whereas lines 71 to 81 should utilize D2, otherwise your solutions for θ_2 will not be correct.

6. Complete lines 89 to 92 to correctly calculate the four solutions for θ_1.

7. Lastly, complete lines 95 to 98 to correctly map the DH joint angles $\vec{\Theta}$ back to the physical joint angles $\vec{\Phi}$.

8. The function outputs all four inverse kinematics solutions (`phi`) as a 4 by 4 matrix. The function has been pre-coded to also return the optimal solution (**phi_optimal**). Take notes on your interpretation of **phi_optimal**.

9. At the end of the inverse kinematics function, you will find the definition of a smaller sub-function - check_joint_limits. What does it do?

10. Once you have completed the function, run the model using the green Play button  under the Simulation Tab and test your implementation using the displays found under *End-effector Position (m)* to verify that the commanded and current end-effector positions match. To command a series of common end-effector positions, use the constant and multiport switch found under *Common Positions of Interest*. For example, to command the home position of [0.45, 0, 0.49], set the constant to 1. For each commanded position, use the *Solution # (1 to 4)* constant to view each of the four possible solutions in the *Current Joint Position (deg)* display. Remember that due to the manipulator's joint limits, not all calculated solutions are valid. Make a note of any invalid solutions.

11. Stop the model

Teach Pendant (Learn)

1. Open the Simulink model TeachPendant_Learn.slx (Figure 2). You will use this model to "teach" the Quanser QArm a series of waypoints that simulate a robotic assembly process. This model emulates the learning mode of a real teach pendant.

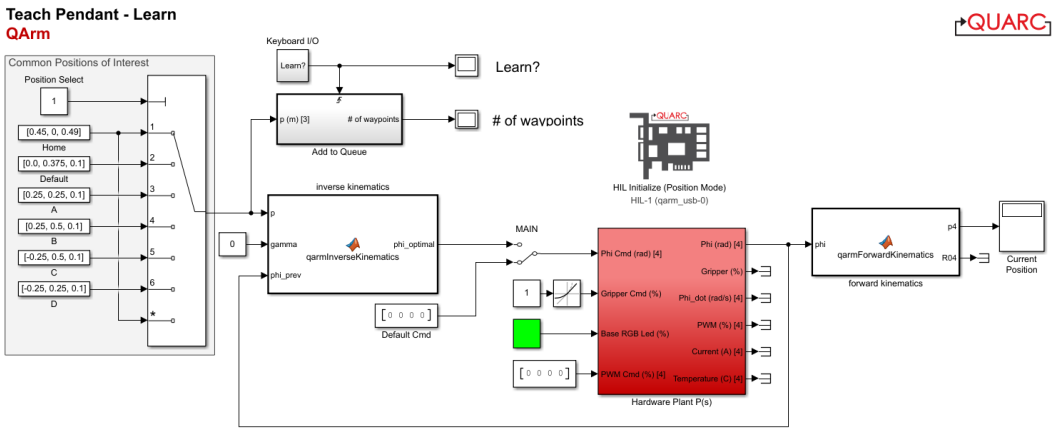

Figure 2: Simulink model *TeachPendant_Learn.slx*

2. Replace the forward and inverse kinematics functions in the above model with the same functions that you developed previously.

3. In the remainder of this lab, you will only use the optimal inverse kinematics solution. Therefore, modify the inverse kinematics function such that it only outputs **phi_optimal**, in other words, remove the **phi** output from the function.

4. Open the Add to Queue subsystem and verify that the logic correctly implements a first-in-first-out (FIFO) queue when the '**Learn?**' signal triggers it. The model will use this subsystem to maintain a queue consisting of the desired waypoints.

5. Prior to running the model, complete the following steps:

    a. Set the Position Select constant to 1.

    b. Set the manual switch labelled MAIN to output *phi_optimal*.

    c. Under *Common Positions of Interest*, set the Default position to [0, 0.375, 0.2] and Home position to [0.45, 0, 0.49].

    d. Set positions A through D to the values shown below:

        i. A = [0.25, 0.25, 0.1]

        ii. B = [0.25, 0.5, 0.1]

        iii. C = [-0.25, 0.5, 0.1]

        iv. D = [-0.25, 0.25, 0.1]

    e. 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

6. Builds and deploys the model using the  Monitor & Tune action. Once started, the model will command the home position to the manipulator.

7. Using the Position Select constant, navigate through all the waypoints, and validate that the manipulator's end-effector moves to each of the desired positions.

8. Set Position Select back to 1 and ensure the # of waypoint display is showing a value of 0. If not, stop and re-start the model. This should clear the queue of points that may have been unintentionally added when tapping the 'Space' key.

9. With # of waypoint display showing a value of 0, set Position Select to 2 (*Default* position). Hit the 'Space' key on your PC to add this position to the queue. When successfully added, the # of waypoint display should increment by 1. Move the manipulator to waypoints A through D while ensuring that you move back to Default between each position, as well as at the end. Once the process is completed, the following 9 positions will be added to the queue: Default, A, Default, B, Default, C, Default, D, Default.

10. At any point, if you need to restart the process, stop and re-run the model to clear the queue.

11. Once you have successfully added all the required waypoints, stop the model and close it.

Teach Pendant (Follow)

1. Open the Simulink model TeachPendant_Follow.slx (Figure 3). You will use this model to command the manipulator to follow the waypoints that were learned in the previous section.

2. Replace the forward and inverse kinematics functions in the above model with the same functions that you developed previously.

3. Open the Playback subsystem. Complete the Waypoint index based on timer section such that the model indexes each of the waypoints stored in the queue every 2 seconds. In other words, when the model is executed, the *Playback* subsystem will output a new waypoint every 2 seconds.

4. Builds and deploys the model using the Monitor & Tune action. Once started, the model will command the home position to the manipulator.

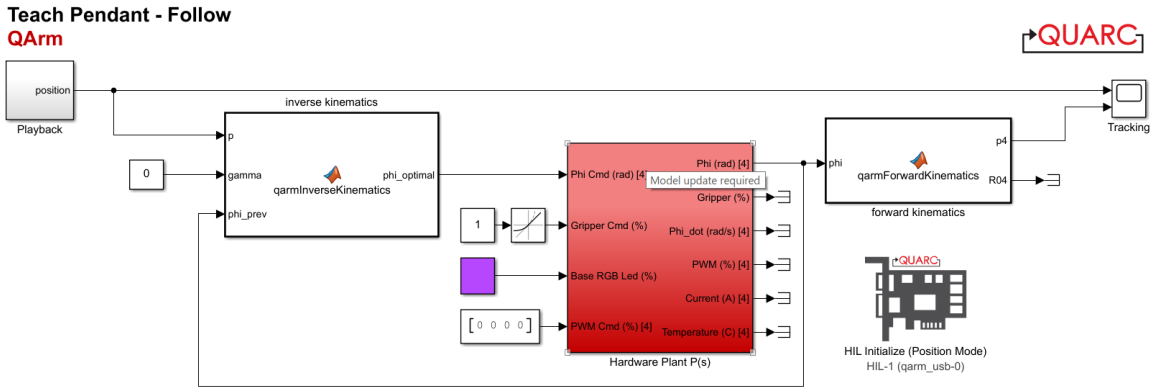

Figure 3: Simulink model *TeachPendant_Follow.slx*

5. Use the Tracking scope to monitor the desired and actual end-effector positions as the manipulator follows the learned waypoints. Sample results are shown in Figure 4.

6. Stop the model and turn off the arm and bring it back to rest position.

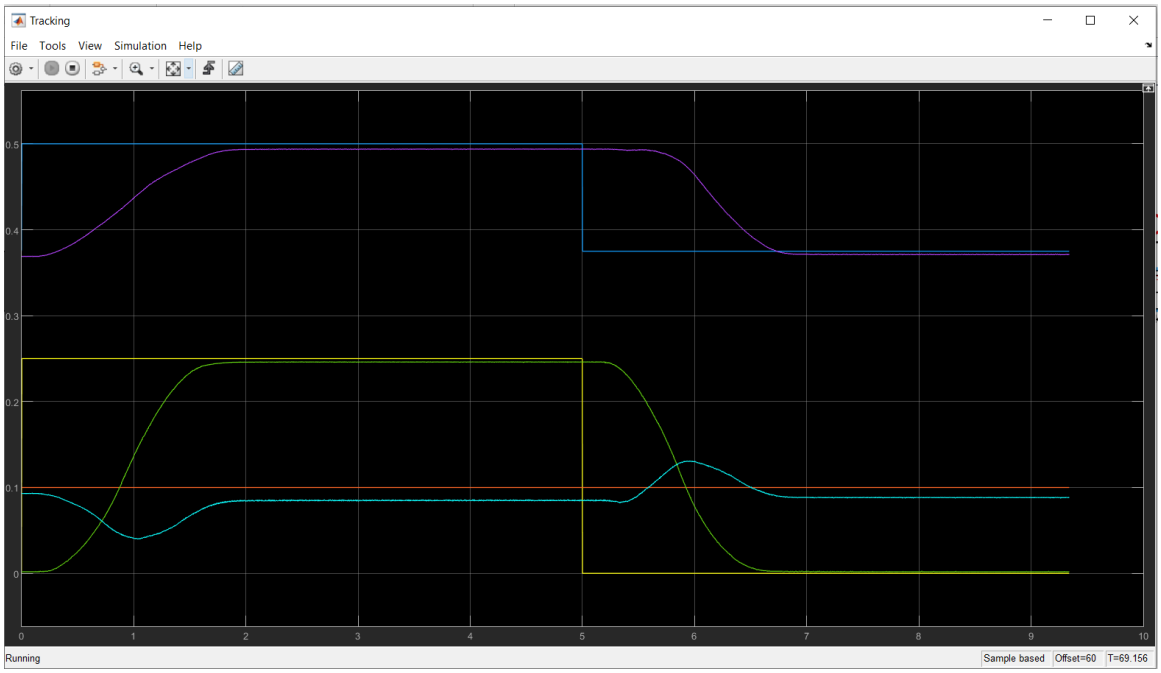

Figure 4: Sample result showing commanded and actual end-effector positions