QArm Lab Procedure
Tool Manipulation

Setup

1. It is recommended that you run this lab individually.
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 6 – Tool Manipulation.

Differential Kinematics

1. The Welding_traj_gen.slx in this lab is incomplete, but is a copy of the Welding.slx, model from lab 5. Use your completed model from the previous lab instead.
2. Prior to running the model, open each 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): 500Hz
3. Build and deploy the model using the Monitor & Tune action button under the Hardware Tab of your model. Monitor the time taken to go through the 4 sides of the rectangular welding trajectory. Is the time provided the same? If so, is the rate of motion for the end-effector uniform?
4. Close the model. Open Jacobian.slx. This model asks you to develop the manipulator's Jacobian matrix which relates the manipulator's joint speeds to the task speeds of its end-effector.

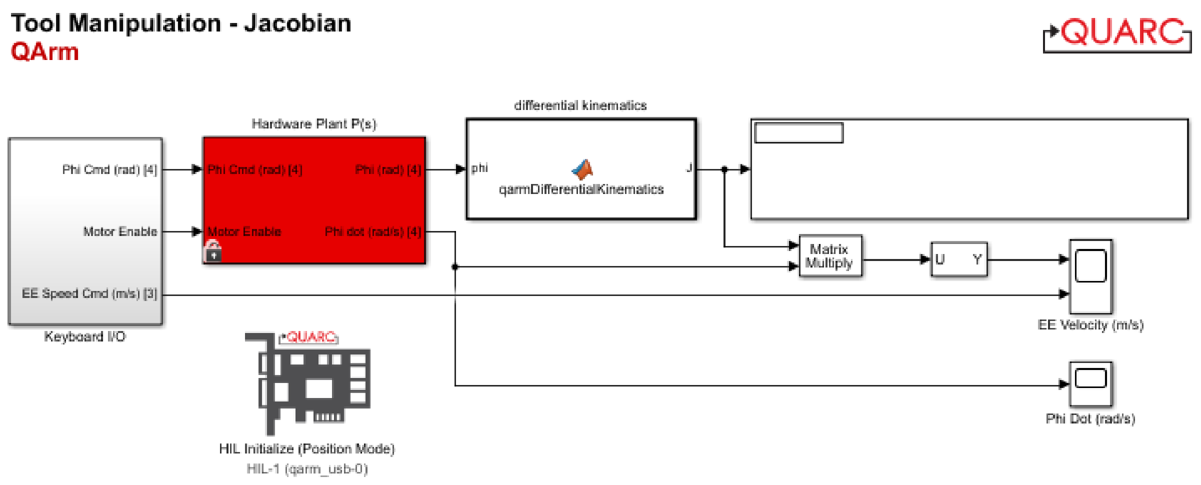


Figure 1 - Jacobian Simulink Model

5. Double click on the embedded MATLAB function qarmDifferentialKinematics. Lines 42 to 57 represent the 16 elements of the Jacobian matrix. Note that the Jacobian is provided to you in the Tool Manipulation Concept Review. Complete this function and save it.

6. Build and deploy the model using the Monitor & Tune action button under the Hardware Tab of your model and use the 4 arrow keys to control the x-y movement of the end-effector. Use Q and A keys to control the z direction movement of it. The display block shows the Jacobian matrix that your function calculates. The matrix multiplication block multiplies the Jacobian and the joint velocities **phi_dot** *(rad/s)*, to give you the End effector velocities. Monitor the End effector velocities versus your commands from the keyboard via the provided scopes. Verify that your Jacobian is implemented correctly.

State Machine

7. Open Welding_tool_man.slx. This model asks you to complete the embedded MATLAB function stateMachine and qarmDifferentialKinematics. This model will repeat the welding task using commands that represent uniform end-effector motion via the Jacobian.

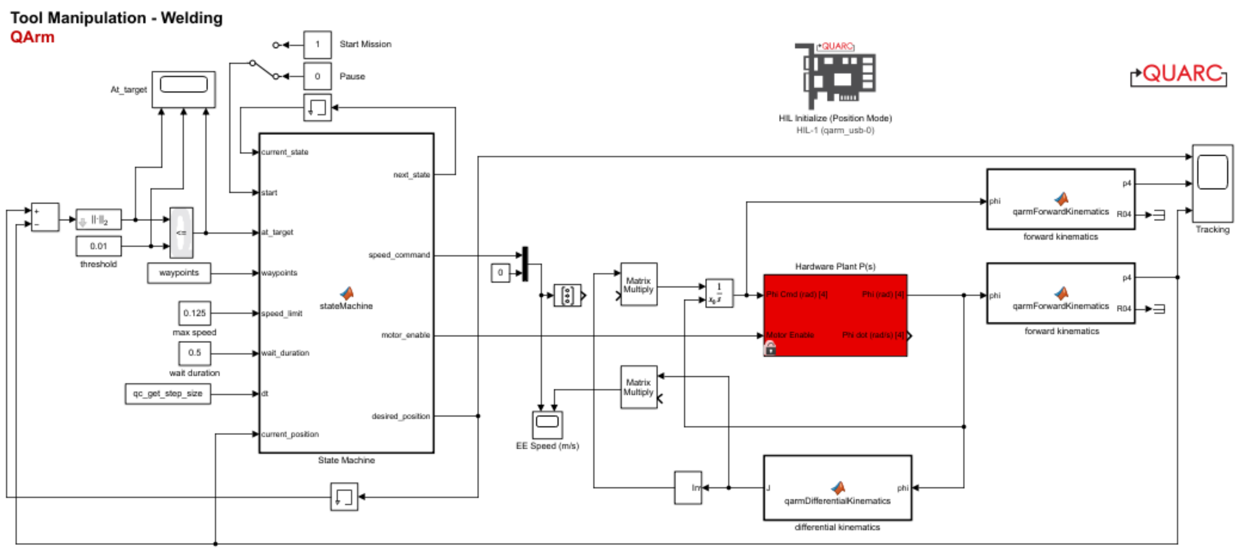


Figure 2 – Simulink model for Welding via tool manipulation

8. Replace the qarmDifferentialKinematics function with the completed version from the Jacobian.slx model. Double click on stateMachine, where the Default Case is completed for you (lines 90 to 97). Each case includes a **State Transition** section where you decide what the next state must be. This is followed by a **State Action** section that determines the output values.

9. Analyze the cases one at a time, while checking the state machine diagram Figure 1 in the Tool Manipulation Concept Review. If start is true, you want to transition to State 2, and if not, you want to remain in state 1. Complete lines 22 and 24 to reflect this transition.

10. When the state reaches case 2, there are three possible states. Based on the logic, complete lines 36, 38 and 40 to reflect this. Regardless of the next state logic, in this state, you want to move towards the target. Set the **speed_command** to move the end-effector towards the next waypoint using **motion_direction** at a constant rate of **max_speed**.

11. When the state reaches case 3, there are also three possible states. Based on the logic, complete lines 51 to 67 to complete the **State Transition** section. In this state you do not want the manipulator to move. Set the speed command in the **State Action** accordingly.

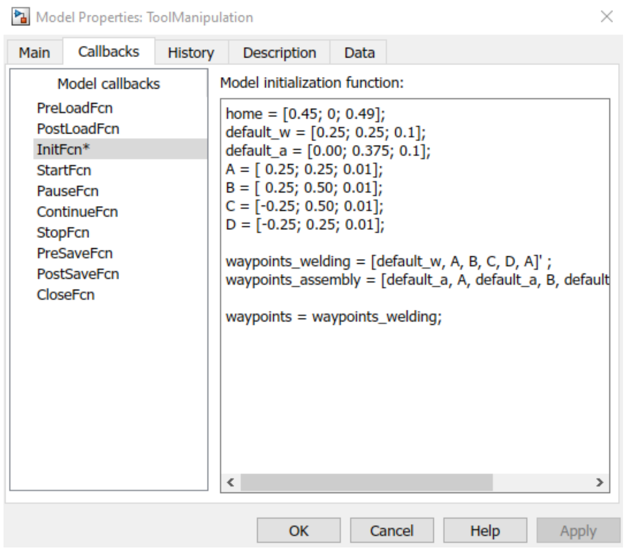
12. Similarly, complete lines 78, 81 and 85 for case 4.

13. Back at the root level, finish the unconnected blocks to complete the matrix multiplication.

14. Build and deploy the model using the Monitor & Tune action button under the Hardware Tab of your model and verify that the welding operation now takes place at a uniform rate regardless of distance between waypoints.

15. Vary the **max speed** variable in the Simulink model to speed up or slow down the algorithm. If you speed up the rate that the manipulator moves, you may have to relax the **threshold** for calculating whether you are close to the waypoint. Why is this?

16. Right click on the blank space in the Simulink model and select **Model Properties**. Click on **Callbacks** panel and select **InitFcn**. Change **waypoints** parameter to **waypoints_assembly**. Verify the rate and the trajectory is correct. You will see that the arm will perform like in an assembly line which will pick a part in a fixed position and assemble it to many places.



```
home = [0.45; 0; 0.49];
default_w = [0.25; 0.25; 0.1];
default_a = [0.00; 0.375; 0.1];
A = [ 0.25; 0.25; 0.01];
B = [ 0.25; 0.50; 0.01];
C = [-0.25; 0.50; 0.01];
D = [-0.25; 0.25; 0.01];

waypoints_welding = [default_w, A, B, C, D, A]' ;
waypoints_assembly = [default_a, A, default_a, B, default

waypoints = waypoints_welding;
```

Figure 3 Model Properties Panel

17. In your Simulink model, set the 'wait duration' constant to the amount of time the manipulator should wait at each location. The arm will now wait at each waypoint, which reflects the time needed to carry out any assembly related operations.

18. Stop the model and close all experiments.

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