MEGN544 - Dr. Petruska - Project Part 2

Student: Thong Quoc (Bill) Huynh

October 21, 2023

With the random theta trajectory initially set in the file SimRun-ThetaSpace.m, we verify that the robot arm moves around in the figure (plotArm). From the file SimRun-ThetaSpace.m, we see that the initial theta-configuration for the simulated arm is as below.

$$\begin{bmatrix}\theta_1\\ \theta_2\\ \theta_3\\ \theta_4\\ \theta_5\\ \theta_6\end{bmatrix} = \begin{bmatrix}0\\ 1.5708\\ 0\\ 0\\ 0\\ 0\end{bmatrix}$$

The initial theta values can be displayed in the MATLAB Command Window by typing

```
theta_actual.signals.values(1,:)
```

Looking at the Theta Controller, we see that the port 1 on the top left corner is the Theta Target, which connects to theta-des in the Trajectory Generator. Inside the Trajectory Generator, line 26 shows that theta-des is from the trajectory variable. The trajectory variable shows up in the section commented as "Define Theta space Trajectory" inside SimRun-ThetaSpace.m.

We start with a single point for the theta target. We set it so that at the end of 2 seconds, all theta's should reach $pi/8$ with the following MATLAB assignment for the trajectory variable:

```
trajectory = [2, ones(1,6)*pi/8];
```

Then we continue with 2 points with a gap, the first point being all theta's at $pi/8$, and the second point being all theta's at $pi/4$:

```
trajectory = [2, ones(1,6)*pi/8;
              4, ones(1,6)*pi/4];
```

For the case with CSM trajectory, there are 26 points given in the CSM trajectory. To make sure the robot arm will start and stop with zero velocity at the first and the last point, we will duplicate the first point of CSM and duplicate the last point of CSM (same point at 2 consecutive time stamps). This will make 28 points. Together with the initial theta configuration point of the robot arm, there will be a total of 29 points, which means 28 gaps of 2 seconds each in between. Therefore, we will set the simTime to be 56 seconds to simulate the robot arm going through the CSM trajectory. In the Theta space trajectory, we only define the points starting from the end of 2 seconds and leave out the initial configuration, meaning there will be 28 rows in the trajectory variable.

We use abbInvKine() function to find the theta's (1 to 6) for each point in the CSM trajectory. The desired pose (T-des) includes the desired orientation (rotation R) and the desired position. The position can be found directly from each 3D point in the points3D.mat file, which can be transformed from points2D.mat using the transformation calculations used in Project Part 1. The orientation at each point can also be found using the same calculations in Project Part 1. For the last point in the CSM trajectory, there is no next point for the x-axis of the orientation to point to. Therefore, the orientation at the last point in the CSM trajectory stays the same as the immediately previous orientation. The reachable variable returned in the abbInvKine() function is not utilized in this part of the project because even if a pose is not reachable, the abbInvKine() function already accounts for it by using the real

part in the complex theta solution, which puts the manipulator in a position as close to the unreachable position as possible.

Plots of desired theta trajectory, actual theta trajectory, and theta error as a function of time for each case (1 point, 2 points, and CSM points) are included below. The videos of the robot arm motion for each case are included in the submission zipped folder.

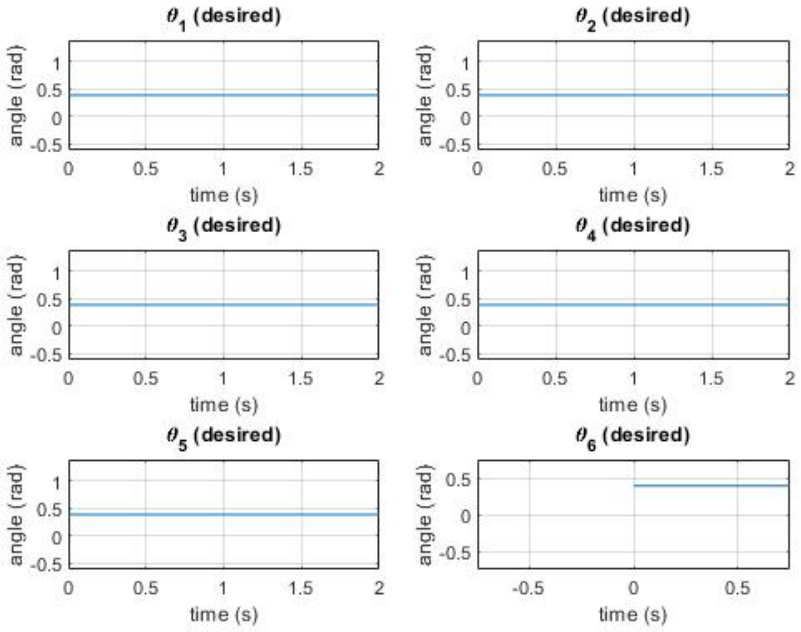

Figure 1: Desired theta trajectory for the case with a single target point.

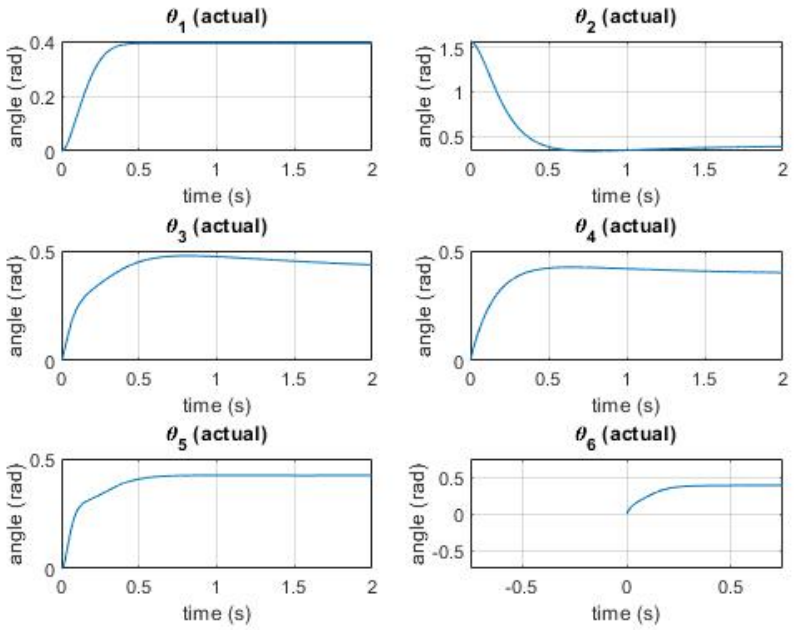

Figure 2: Actual theta trajectory for the case with a single target point.

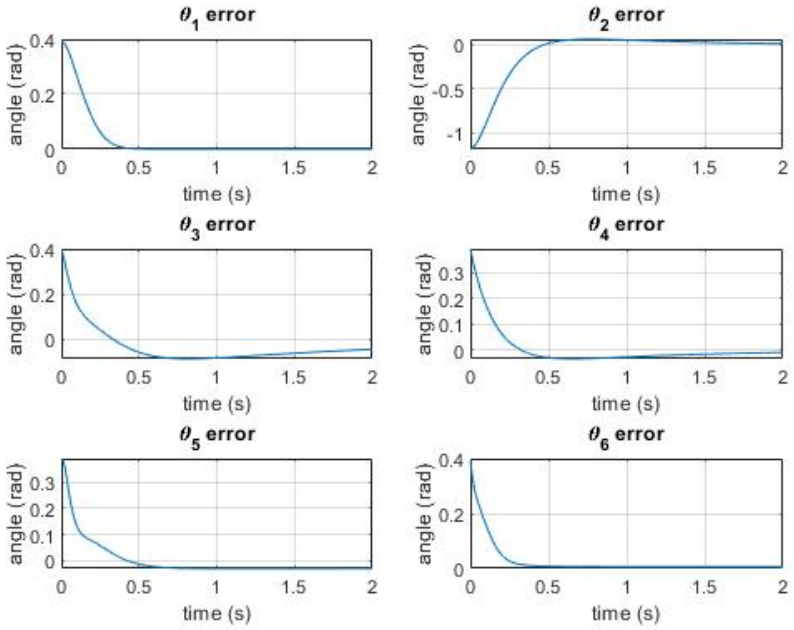

Figure 3: Theta error for the case with a single target point.

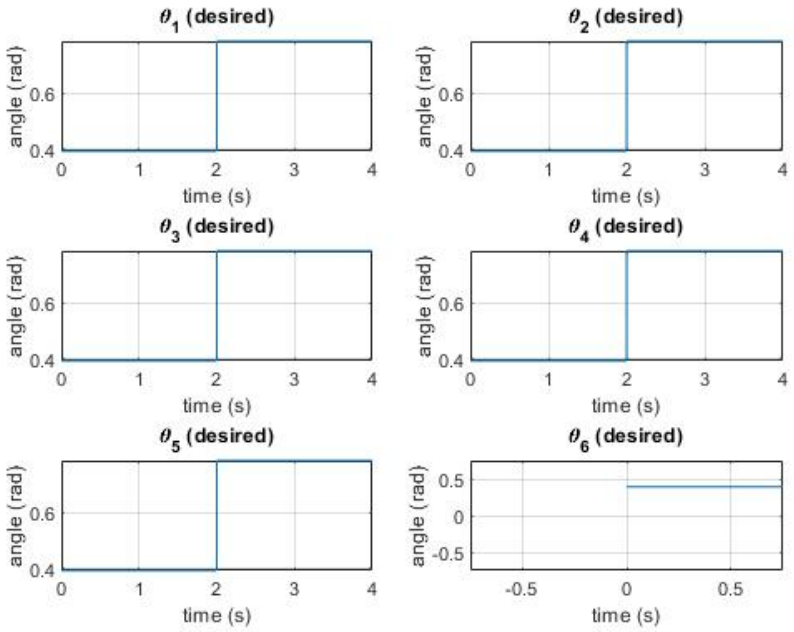

Figure 4: Desired theta trajectory for the case with two target points.

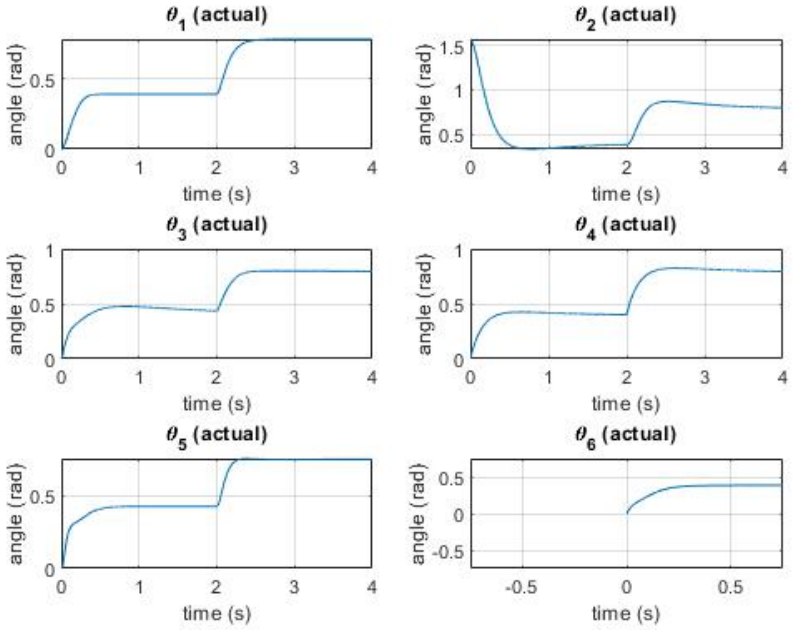

Figure 5: Actual theta trajectory for the case with two target points.

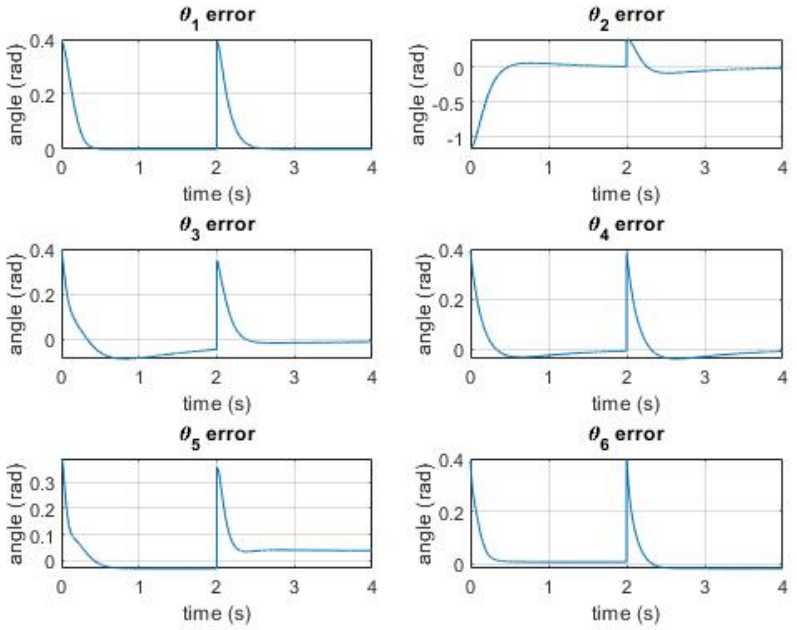

Figure 6: Theta error for the case with two target points.

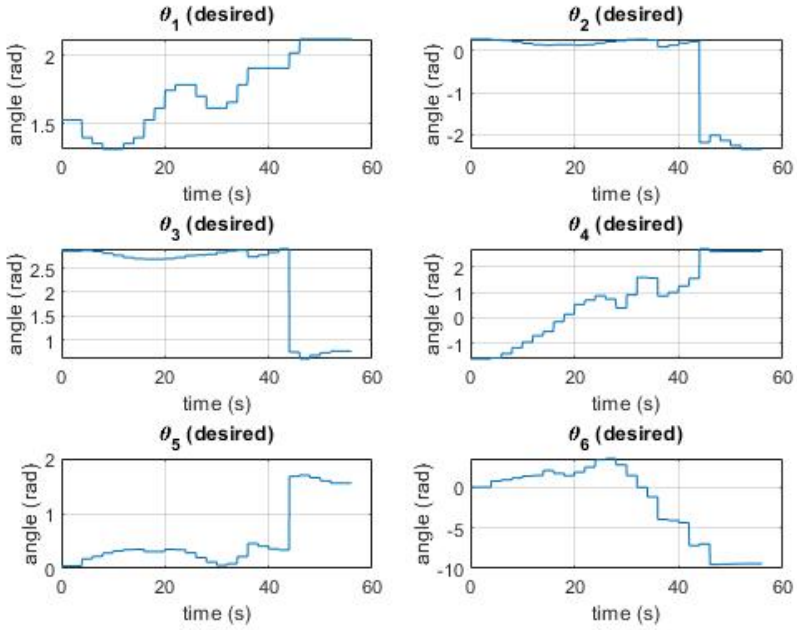

Figure 7: Desired theta trajectory for the case with CSM points.

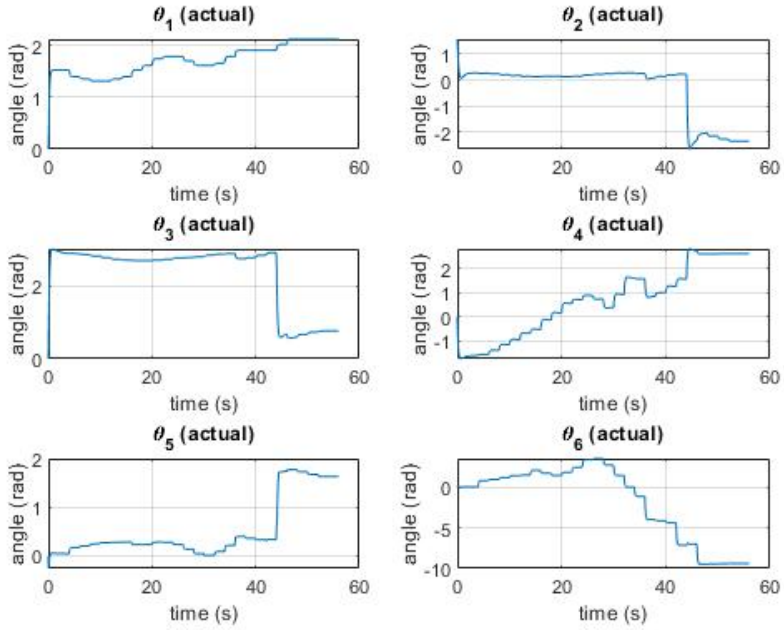

Figure 8: Actual theta trajectory for the case with CSM points.

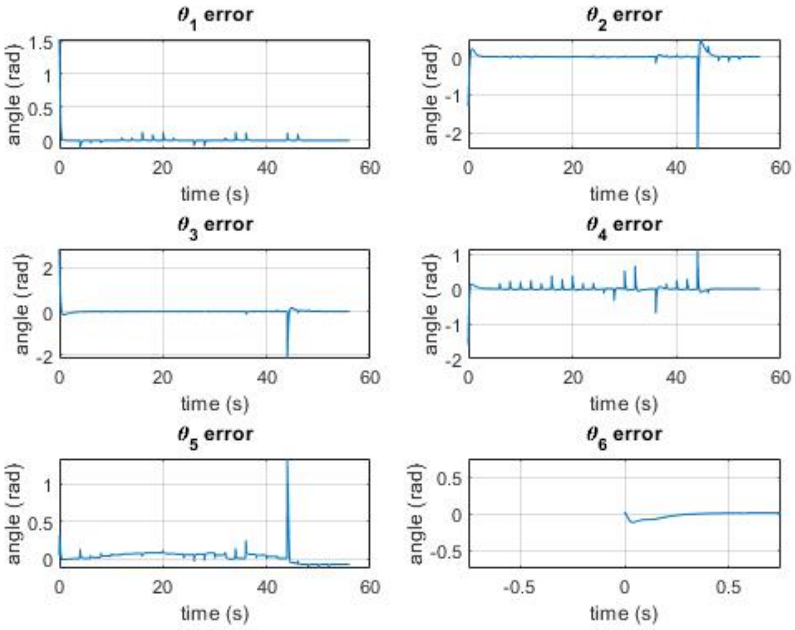

Figure 9: Theta error for the case with CSM points.

With a single target point and two target points, the errors between desired and actual theta trajectories show curves converging from the initial theta configurations to the desired theta configurations. The desired thetas are simply constant over each time period and they locally look similar to a step function if there is a gap between two successive targets.

For the CSM trajectory, considering the overall shapes of the curves, the actual theta trajectory curves look roughly similar to the desired theta trajectory curves. In the theta error plots, the starting period shows a quick convergence. However, there is a large error at approximately 45 seconds seen in the theta error plots for $\theta_2,\ \theta_3,\ \theta_4,\ and\ \theta_5$. This may be due to the joint limits of the robot arm, requiring it to change orientation drastically just to move to a neighboring point of the end manipulator. It could also be due to the inaccuracy of constant acceleration interpolation. In the arm motion video for the CSM trajectory, we see the arm perform a big rotation movement while attempting to draw the M letter.

Solely reducing the proportional gain K_p, which basically relates to the ratio of output response to the value of errors, down to 50% of the original values, we observe through the theta error plots a slower convergence at the beginning. The error curves also become noisier (spikes along the error curves showing larger magnitudes).

Solely increasing the proportional gain K_p up to 150% of the original values, we observe a quicker convergence at the beginning. The error magnitudes do not show significant changes.

Solely reducing the derivative gain K_d, which basically relates to the rate of change of the errors, down to 50% of the original values, we observe a quicker convergence at the beginning. There are not significant changes to the error magnitudes.

Solely increasing the derivative gain K_d up to 150% of the original values, we observe a slower convergence at the beginning and more noise (especially for θ_4).

Solely reducing the integral gain K_i, which basically relates to the total accumulated errors, down to 50% of the original values, we observe a quicker convergence at the beginning without significant changes to the error magnitudes.

Solely increasing the integral gain K_i up to 150% of the original values, we observe a quicker convergence at the beginning with slightly higher error magnitudes.

If we decrease all 3 types of gains in the amounts above simultaneously, we do not see any significant changes.

If we increase all 3 types of gains in the amounts above simultaneously, we only observe changes in the magnitudes of the errors of θ_4.

I believe the implemented theta-space controller uses constant acceleration interpolation because the ABB geometry load section in the sim-run script checks for the file constAccelInterp.m. I do not think the theta-space controller in this part of the project works well enough to trace CSM, because the actual CSM trajectory still shows considerable errors (wiggles and overshoots) towards the M letter. The robot arm having to perform a big re-orientation to continue tracing the M letter is not a good sign.