ECE470 LAB-1 REPORT Deniz Akyildiz (1003102224)

EXPERIMENT

4.1 Definition of robot structure

Using the values of DH table from the preparation, the robot structure was created. The code for the model and the DH table are shown in Figures 1 and 2.

```
%% 4.1 Definition of robot structure

% DH Table from preparation report.
% All units in cm.
DH = [0, 76, 0, pi/2; ...
      0, -23.65, 43.24, 0; ...
      0, 0, 0, pi/2; ...
      0, 43.18, 0, -pi/2; ...
      0, 0, 0, pi/2; ...
      0, 20, 0, 0];

% Create the robot model.
myrobot = mypuma560(DH);
```

Figure 1: DH Table

```
function myrobot = mypuma560(DH)
    % Function to create a robot model from given DH table.
    L(1) = Link([DH(1,:)], 'standard');
    L(2) = Link([DH(2,:)], 'standard');
    L(3) = Link([DH(3,:)], 'standard');
    L(4) = Link([DH(4,:)], 'standard');
    L(5) = Link([DH(5,:)], 'standard');
    L(6) = Link([DH(6,:)], 'standard');
    myrobot = SerialLink(L, 'name', 'myrobot');
end
```

Figure 2: Robot model using the toolbox.

4.2 Plot a sample joint space trajectory

In this part, we plot the robot motion where the robot follows the trajectory associated with the given joint angles. Figure 3 shows the code, and Figure 4 shows the resulting plot.

```
%% 4.2 Plot a sample joint space trajectory
% Initialize the given joint angles.
theta_1 = linspace(0, pi, 200);
theta_2 = linspace(0, pi/2, 200);
theta_3 = linspace(0, pi, 200);
theta_4 = linspace(pi/4, 3*pi/4, 200);
theta_5 = linspace(-pi/3, pi/3, 200);
theta_6 = linspace(0, 2*pi, 200);
% Create q matrix that consists of joint angles.
q = [theta_1.' theta_2.' theta_3.' ...
     theta_4.' theta_5.' theta_6.'];
% Plot the trajectory.
plot(myrobot, q);
```

Figure 3: Plotting the sample trajectory

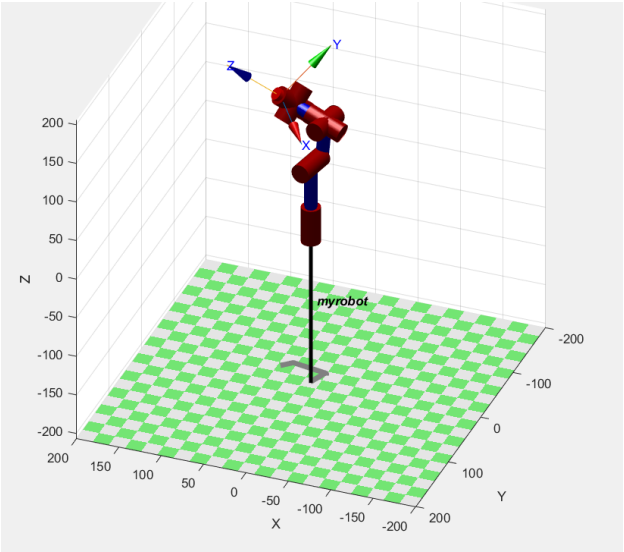

Figure 4: Robot plot following sample joint trajectory

4.3 Forward Kinematics

Following the steps given in section 2.1 of the handout and using the equations from preparation, the forward kinematics was calculated. Function **forward.m** returns the homogeneous transformation matrix H for finding the end effector position and orientation. Figure 5 shows the code for calculating forward kinematics. Figure 6 verifies the result by comparing the trajectory of end effector and the robot motion. Here the red line is the expected trajectory.

```
function H = forward(joint, myrobot)
    % Calculates forward kinematics given joint angles and robot model.
    H = eye(4);

    % Loop to create individiual matrices from ith to (i-1)th reference
    % frames, and multiply them to get the forward kinematics matrix.
    for i = 1:6
        theta_i = joint(i);
        alpha_i = myrobot.alpha(i);
        a_i = myrobot.a(i);
        d_i = myrobot.d(i);
        H_i = [cos(theta_i) -sin(theta_i)*cos(alpha_i) ...
               sin(theta_i)*sin(alpha_i) a_i*cos(theta_i); ...
               sin(theta_i) cos(theta_i)*cos(alpha_i) ...
               -cos(theta_i)*sin(alpha_i) a_i*sin(theta_i); ...
               0 sin(alpha_i) cos(alpha_i) d_i; ...
               0 0 0 1];
        H = H * H_i;
end
```

Figure 5: Forward kinematics calculation

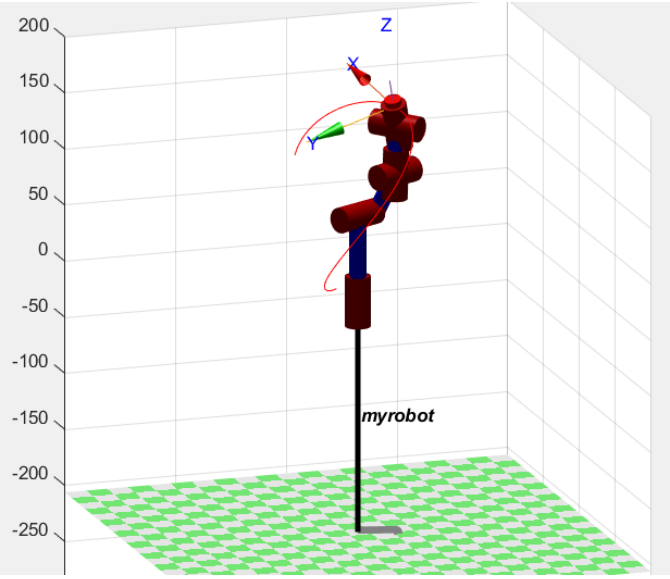

Figure 6: Verification of forward kinematics

4.4 Inverse Kinematics

Following the steps given in section 2.2 of the handout and using the equations from preparation, the inverse kinematics was calculated. Function **inverse.m** returns vector q, which includes the joint angles. Due to the length of the inverse.m function, a screenshot is not included here, however, it could be found in the submitted code. Figure 7 shows the numerical verification for inverse kinematics, as shown in the handout. Similar to previous section, Figure 8 shows the trajectory and robot movement to verify inverse kinematics.

```
%% 4.4 Inverse Kinematics

% Verify inverse kinematics equations, compare results with the q values
% given in section 4.4.
H = [cos(pi/4) -sin(pi/4) 0 20; ...
    sin(pi/4) cos(pi/4) 0 23; ...
    0 0 1 15; 0 0 0 1];
disp('Verify inverse kinematics:')
q = inverse(H,myrobot)
```

```
nmand Window
w to MATLAB? See resources for Getting Started.
Verify inverse kinematics:

q =

   -0.0331   -1.0663    1.0275    3.1416    3.1028    0.8185
```

Figure 7: Verification of inverse kinematics numerically

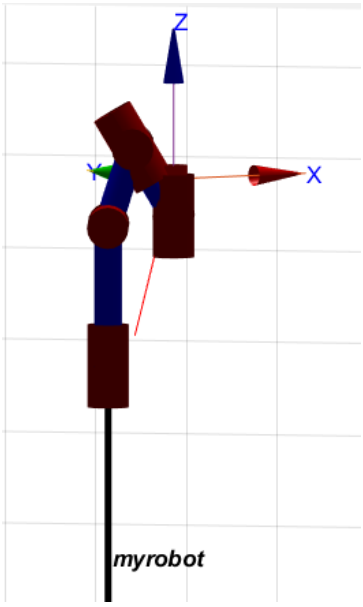

Figure 8: Verification of inverse kinematics