Imperial College London

COURSEWORK

IMPERIAL COLLEGE LONDON

DEPARTMENT OF SURGERY AND CANCER

Medical Robotics and Instrumentation

Author:
María de la Paz Cardona Sánchez (CID: 01410045)

Date: January 19, 2022

Task 1

The first step in the robot simulation is to derive the DH parameters using the modified DH convention. These parameters, shown in Table 1, are used to define the properties of a *Link* class object for each of the joints in the robot manipulator. The six individual link objects are used to define a *SerialLink* robot class that represents the whole robot. Zero configuration of the robot is displayed using the *plot* method of the Serial-link class, specifying a vector of zeros as the joints angles (θ_{1-6}) to show the zero configuration. The end-effector is defined using the *tool* property and setting it to *transl*(L_6,0,0). Graphical representation of the robot in its zero configuration when using the *teach* method instead of *plot* is shown in Figure 1. The workspace is the region of operation of the end-effector, whose volume is limited by the length of its links and the articular limits of the joints, which ranged from -π to π for every joint. In this case, the robot can reach most of the positions in a radius less than or equal to 2.7 ($\sum_{i=2}^{6} L_i$) as shown in Figure 2. This figure was generated with a modified version of the code proposed in (1), which takes ~1 min to run.

Frame i	a_{i-1}	α_{i-1}	d_i	θ_i	offset
1	0	0	L_1	θ_1	0
2	L_2	$-\pi/2$	0	θ_2	$-\pi/2$
3	L_3	0	0	θ_3	0
4	0	$-\pi/2$	L_4	θ_4	0
5	0	$\pi/2$	0	θ_5	0
6	-L_5	$-\pi/2$	0	θ_6	π

Table 1: DH table

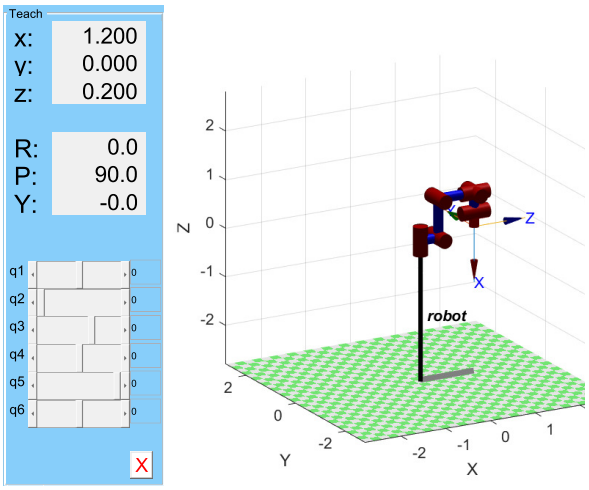

Figure 1: Robot's zero configuration.

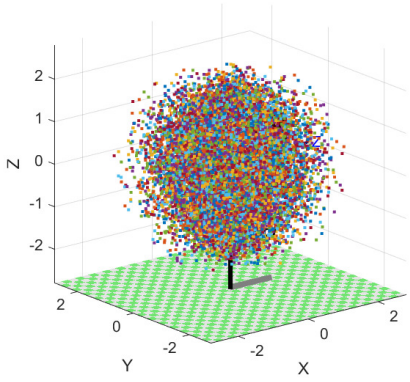

Figure 2: Robot's workspace.

To follow the linear trajectory along a sphere defined in *pos_static_trajectory*, a homogeneous transformation matrix is calculated using *transl* to represent translation of the end-effector to the desired pose. The result is a homogeneous transform (4x4x29 matrix), representing the transform for each step in the trajectory. This transform is used as an input to the function *ikine* to find the joints coordinates through inverse kinematics at each step of the trajectory, giving a 6x29 matrix that represents the coordinates of the 6 joints at the 29 different steps. When inputting the joints coordinates to the *plot* function, it animates the robot moving along the trajectory.

To calculate the end-effector position error, first, we use forward kinematics, with *fkine*, to find the end-effector pose from the joints coordinates matrix obtained before. Then, the translation element of the homogeneous transform obtained from the forward kinematics is substracted from the coordinates in *pos_static_trajectory*. The result gives an error value in the x, y, and z coordinates for each of the steps (3x29 matrix), as shown in Figure 3. The low error shown in this figure, in the order of 10^{-12}, proves that the robot follows the trajectory in an acceptable way.

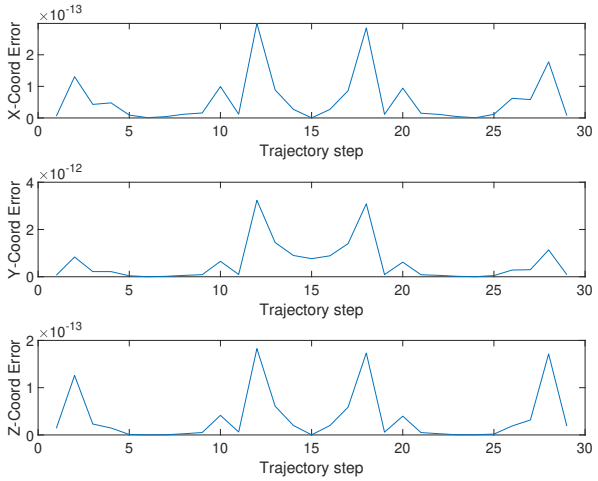

Figure 3: End-effector position error.

Task 2

In the second task, the end-effector is kept perpendicular to the sphere surface at all points along the trajectory. To achieve this, an orthonormal rotation matrix (3x3x29) of each of the angles in *alpha* about the vector *vector* is defined using *angvec2r*. This rotation matrix and the translation vector of *pos_static_trajectory* are used to define a new homogeneous transformation matrix (4x4x29) using *rt2tr*. Again, the joints coordinates at each step of the trajectory (6x29) are found using inverse kinematics through *ikine* and the animation is displayed with the *plot* function.

The error in the end-effector pose is calculated in a similar process to Task 1. First, forward kinematics are applied to calculate the end-effector pose. Then, the translation element of the homogeneous transform obtained from the forward kinematics is subtracted from the coordinates in *pos_static_trajectory*, giving the x, y, and z coordinate errors shown in Figure 4. To find the orientation error, the rotation matrix is extracted from the transformation matrix obtained from *fkine* using *tr2rt*. This rotation matrix is used to find the Euler angles using *rotm2eul*. The calculated angles are subtracted from the given angles, obtained from multiplying the *alpha* and *vector* vectors. The result is shown in Figure 5. These figures show how the pose error is in the order of 10^{-11}, proving the robot adequately follows the trajectory.

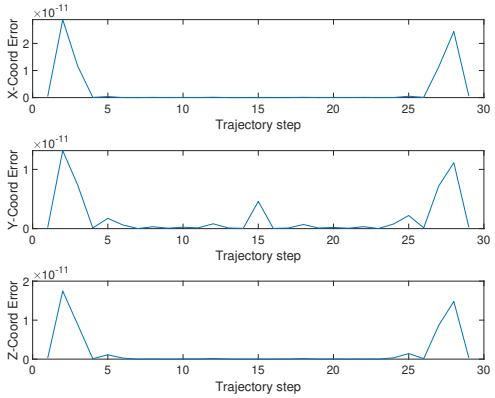

Figure 4: End-effector position error.

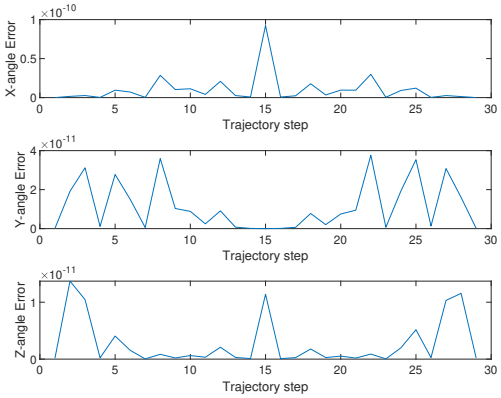

Figure 5: End-effector orientation error.

Task 3

The first part of Task 3 repeats the inverse kinematics process of Task 1 but adds joints velocities as an input to *ikine*. The first step is to calculate the homogeneous transformation using *transl* function and *pos_beat_trajectory* as an input. Then, the Jacobian is calculated using the homogeneous transform as an input to *jacobe*. The Jacobian's inverse is multiplied by the desired 0.1 m/s end-effector velocity to obtain a vector containing the velocity for each joint. As in previous tasks, joints coordinates together with the caculated velocity vector are inputted to *ikine* and the result is displayed using *plot*. The end-effector position error is calculated in the same way as in Task 1 and is shown in Figure 6, which displays a very low error of the order of 10^{-12}. This figure also incorporates the end-effector velocity error. End-effector velocity was calculated using forward kinematics to find the position difference between two consecutive steps and dividing it by the time step. The norm of the velocities in the x, y, and z coordinates was calculated to obtain overall velocity. End-effector velocity starts at being 0 m/s, giving an error of 0.1, but after the first step it reaches a constant value of 0.1 m/s, giving the zero error shown in the bottom plot of Figure 6. The low error displayed in Figure 6 proves the robot follows the trajectory in an suitable way with the desired constant velocity.

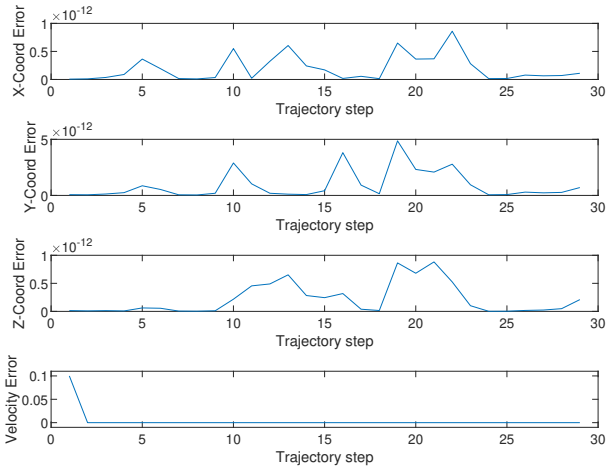

Figure 6: End-effector position and velocity error.

The second part of Task 3 repeats the inverse kinematics process of Task 2 and incorporates joints velocities the same way the first part of Task 3 does. The end-effector position, orientation, and velocity errors are calculated in the same way as in previous tasks. Figures 7 and 8 show a low value of the order of 10^{-11}, proving the robot satisfactorily follows the trajectory with the desired constant velocity.

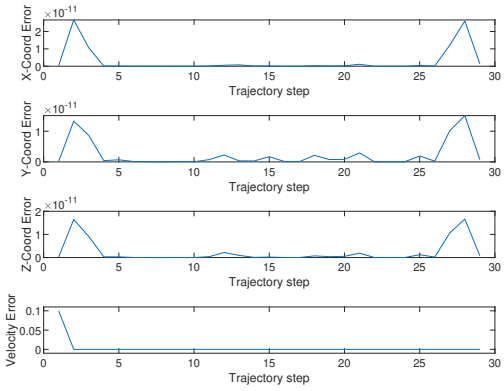

Figure 7: End-effector position and velocity error.

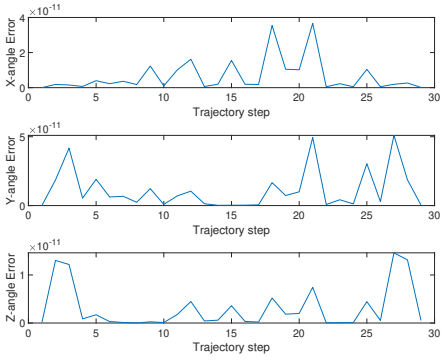

Figure 8: End-effector orientation error.

References

[1] A Reza. Development of Direct Kinematics and Workspace Representation for Smokie Robot Manipulator & the Barret WAM. 2017 5th International Conference on Robotics and Mechatronics (ICROM). `https://arxiv.org/ftp/arxiv/papers/1707/1707.04820.pdf`. pages 2