UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF COMPUTER SCIENCE AND TECHNOLOGY,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELLING AND CONTROL OF MANIPULATORS

Third Assignment

Jacobian Matrices and Inverse Kinematics

Author:

Galvagni Gianluca

Student ID:

s5521188

Professors:

Giovanni Indiveri
Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo

January 13, 2023

Contents

Mathematical expression	Definition	MATLAB expression
$< w >$	World Coordinate Frame	w
${}^a_bR$	Rotation matrix of frame $< b >$ with respect to frame $< a >$	aRb
${}^a_bT$	Transformation matrix of frame $< b >$ with respect to frame $< a >$	aTb

Table 1: Nomenclature Table

1 Assignment description

The third assignment of Modelling and Control of Manipulators focuses on the definition of the Jacobian matrices for a robotic manipulator and the computation of its inverse kinematics.
The third assignment is **mandatory** and consists of three exercises. You are asked to:

- Download the .zip file called MOCOM-LAB3 from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling the predefined files. In particular, you will find two different main files: *"ex1.m"* for the first exercise and *"ex2.m"* for the second and third exercises.
- Write a report motivating your answers, following the predefind format on this document.

1.1 Exercise 1

Given the CAD model of the robotic manipulator from the previous assignment and using the functions already implemented:
Q1.1 Compute the Jacobian matrices for the manipulator for the following joint configurations:

- $\mathbf{q}_1 = [1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3]$
- $\mathbf{q}_2 = [1.3, 0.4, 0.1, 0, 0.5, 1.1, 0]$
- $\mathbf{q}_3 = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$
- $\mathbf{q}_4 = [2, 2, 2, 2, 2, 2, 2]$

1.2 Exercise 2

In the second exercise the model of a Panda robot by Franka Emika is provided. The robot geometry and jacobians can be easily retrieved by calling the following built-in functions: *"getTransform()"* and *"geometricJacobian()"*.
Q2.1 Compute the cartesian error between the robot end-effector frame ${}^b_eT$ and the goal frame ${}^b_{ge}T$.
${}^b_{ge}T$ must be defined knowing that:

- The goal position with respect to the base frame is ${}^bO_g = [0.6, 0.4, 0.4]^T$
- The goal frame is rotated of $\theta = -\pi/4$ around the z-axis of the robot end-effector initial configuration.

Q2.2 Compute the desired angular and linear reference velocities of the end-effector with respect to the base: ${}^b\nu^*_{e/0} = \alpha \cdot \begin{bmatrix} \omega^*_{e/0} \\ \upsilon^*_{e/0} \end{bmatrix}$, such that $\alpha = 0.2$ is the gain.
Q2.3 Compute the desired joint velocities. (Suggested matlab function: *"pinv()"*).
Q2.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"*.

1.3 Exercise 3

Repeat the Exercise 2, by considering a tool frame rigidly attached to the robot end-effector according to the following transformation matrix:

$$eTt = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Q3.1 Compute the cartesian error between the robot tool frame ${}^b_tT$ and the goal frame ${}^b_{gt}T$.
${}^b_{gt}T$ must be defined knowing that:

- The goal position with respect to the base frame is ${}^bO_g = [0.6, 0.4, 0.4]^T$
- The goal frame is rotated of $\theta = -\pi/4$ around the z-axis of the robot tool frame initial configuration.

Q3.2 Compute the angular and linear reference velocities of the tool with respect to the base:
${}^{b}\nu^{*}_{e/0} = \alpha \cdot \begin{bmatrix} \omega^{*}_{e/0} \\ v^{*}_{e/0} \end{bmatrix}$, such that $\alpha = 0.2$ is the gain.
Q3.3 Compute the desired joint velocities. (Suggested matlab function: *"pinv()"*).
Q3.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"*.
Q3.5 Comment the differences with respect to Exercise2.
Q3.6 Test the algorithm for a new tool goal, knowing that the transformation matrix of the goal with respect to the robot base is:

$$bTg = \begin{bmatrix} 0.9986 & -0.0412 & -0.0335 & 0.6 \\ 0.0329 & -0.0163 & 0.9993 & 0.4 \\ -0.0417 & -0.9990 & -0.0149 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2 Exercise 1

The goal of the kinematic analysis of a manipulator is to evaluate how, in a certain configuration q, a given set of velocities in the joint space $\dot{q}$ will effect the total velocity $\underline{\dot{x}}_{e/o}$ of the end-effector. This relation is linear and provided by a configuration dependent matrix called the **basic Jacobian**.

We need to calculate the angular and linear velocities and then we can spilt the result into Jacobian matrix and the velocities ($\dot{q}$).

$$w_{e/o} = \sum_{i=1}^{n} w_{i/i-1} \quad \textit{and} \quad v_{e/o} = \sum_{i=1}^{n} v_{i/i-1} + (w_{i/i-1} \wedge r_{e,i})$$

The **angular velocities** between each couple of intermediate frames $w_{i/i-1}$ is a function of the joint velocity $\dot{q}_i$. In particular, since we are using 1 DOF joints, the axis of rotation of frame $< i >$ with reference to $< i-1 >$ is fixed and we can calculate the angular velocity as:

$$w_{i/i-1} = \begin{cases} k_i \cdot \dot{q}_i & \text{if } i \text{ is a } \textbf{rotational} \text{ joint.} \\ 0 & \text{if } i \text{ is a } \textbf{prismatic} \text{ joint.} \end{cases}$$

The **linear velocities** between each couple of intermediate frames is a function of joint velocity $\dot{q}_i$ and, depending on the type of joint, is given by:

$$v_{i/i-1} = \begin{cases} k_i \cdot \dot{q}_i \wedge r_{e,i} & \text{if } i \text{ is a } \textbf{rotational} \text{ joint.} \\ k_i \cdot \dot{q}_i & \text{if } i \text{ is a } \textbf{prismatic} \text{ joint.} \end{cases}$$

If we joint the previous result togheter we get:

$$\begin{cases} w_{i/i-1} = J_{A1} \cdot \dot{q}_1 + J_{A2} \cdot \dot{q}_2 + ... + J_{An} \cdot \dot{q}_n \\ v_{i/i-1} = J_{L1} \cdot \dot{q}_1 + J_{L2} \cdot \dot{q}_2 + ... + J_{Ln} \cdot \dot{q}_n \end{cases}$$

where J_{Ai} and J_{Li} are two column vectors defined as:

$$J_{Ai} = \begin{cases} k_i & \text{if } i \text{ is } \textbf{rotational}. \\ 0 & \text{if } i \text{ is } \textbf{prismatic}. \end{cases} \quad \textit{and} \quad J_{Li} = \begin{cases} k_i \wedge r_{e,i} & \text{if } i \text{ is } \textbf{rotational}. \\ k_i & \text{if } i \text{ is } \textbf{prismatic}. \end{cases}$$

Moreover, the two equations above can be represented in a compact way by considering the total velocity of the end-effector as an unique 6-dimensional vector:

$$\begin{Vmatrix} w_{e/o} \\ v_{e/o} \end{Vmatrix} = \begin{Vmatrix} J_{A_{e/o}}(q) \\ J_{L_{e/o}}(q) \end{Vmatrix} \cdot \dot{q} \quad \longrightarrow \quad \dot{x}_{e/o} = J_{e/o}(q) \cdot \dot{q}$$

2.1 Q1.1

In this first exercise, I calculated the Jacobian matrices for four configuration q using the CAD model for the last assignment.
I used the functions:

- *BuildTree()* to create the tree of frames from the CAD model.
- *GetDirectGeometry()* and *GetTransformationWrtBase()* to generate the transformation matrix from the tree of frames for each configuration q.
- *GetJacobian()* to compute the end-effector Jacobian matrices.

The coniguration are:

- q_1 = [1.3, 1.3 ,1.3 ,1.3 ,1.3 ,1.3 ,1.3];
- q_2 = [1.3, 0.4, 0.1, 0, 0.5, 1.1, 0];
- q_3 = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3];
- q_4 = [2, 2, 2, 2, 2, 2, 2].

And the corresponding Jacobian matrices are:
All first three values for each column are the $w_{i/i-1}$ along the axis x, y and z. The last three values for each column are the $v_{i/i-1}$ along x, y and z.

```
ans(:,:,1) =

         0   -0.9636   -0.0716   -0.5061   -0.8473   -0.0197    0.2847
         0    0.2675   -0.2578   -0.8231    0.4187    0.5899    0.7773
    1.0000         0    0.9636   -0.2578    0.3267   -0.8072    0.5611
  332.3621   54.3368  267.8935  107.6084    2.9068 -146.6361         0
  156.3765  195.7268  165.2129 -193.3893  -86.9691   33.4724         0
         0  278.4197   64.0885  406.2391  119.0037   28.0437         0


ans(:,:,2) =

         0   -0.9636   -0.2464   -0.9691   -0.2464    0.8541    0.2965
         0    0.2675   -0.8875    0.2287   -0.8875   -0.0089   -0.8132
    1.0000         0    0.3894   -0.0920    0.3894    0.5201   -0.5008
 -172.4173   57.3657 -257.4679   25.8352 -116.4574  -65.3871         0
  -47.1129  206.6372   34.4907  -89.5885    1.2145  -89.0395         0
         0 -153.5314  -84.2931 -495.1143  -70.9140  105.8560         0


ans(:,:,3) =

         0   -0.9636   -0.2662   -0.9614   -0.2024    0.9793   -0.1909
         0    0.2675   -0.9587    0.2566   -0.4150   -0.0780   -0.6643
    1.0000         0    0.0998   -0.0993    0.8870    0.1869    0.7226
  119.9420  -79.1761  295.7497  -96.5365  -44.2772  -10.3742         0
  -10.7688 -285.2003  -79.8556 -475.8164    3.5279  113.7315         0
         0  118.4517   21.5995 -294.5973   -8.4523  101.8161         0


ans(:,:,4) =

         0   -0.9093   -0.1732    0.7225   -0.5366    0.6971   -0.2091
         0   -0.4161    0.3784   -0.5786   -0.8144   -0.2803    0.8009
    1.0000         0    0.9093    0.3784   -0.2209   -0.6599   -0.5611
 -277.6738 -248.1043  -26.8875 -186.4315  -96.9863 -104.9246         0
   85.2324  542.1178  180.7494  -31.6789   38.9941  -80.9536         0
         0 -217.0189  -80.3391  307.5100   91.8052  -76.4613         0
```

Figure 1: Jacobian matrices, for configuration q_1, q_2, q_3 and q_4.

0.9986	0.0335	-0.0412	0.6000
0.0329	-0.9993	-0.0163	0.4000
-0.0417	0.0149	-0.9990	0.4000
0	0	0	1.0000

Figure 2: Goal matrix, with bOge $= [0.6; 0.4; 0.4]$ and it has a rotation along the z axis with $\theta = -\pi/4$.

```
ts = 0.5;
t_start = 0.0;
t_end = 30.0;
t = t_start : ts : t_end;
```

Figure 3: Simulation parameters, in this case t = 61.

3 Exercise 2

In the second exercise we are going to use the model of *Panda* robot by Franka Emika.
The goal of the *inverse kinematic problem* is to compute the joint-space velocities $\dot{q}$ that let us to obtain the desired output velocity in the cartesian space. Since the input and output spaces are in a one-to-one linear relation, given by the Jacobian matrix, we can come up with a solution by inverting such matrix.
Inverse kinematic algorithm:

1. Evaluate the projection if the desired output velocity $\dot{x}^*$ on $Span(J(q,p))$, which corresponds to the best approximation we can get in the current situation (q,p) : $\dot{\hat{x}} \simeq \dot{x}^*$.
2. Invert the one-to-one relation in order to obtain the minimal input component $\dot{\hat{q}}$ that will produce the feasible $\dot{\hat{x}}$.
   $\Rightarrow \quad \dot{\hat{x}} = J(q,p) \cdot \dot{q} \longrightarrow \dot{\hat{q}} = J^{\boxtimes}(q,p) \cdot \dot{x}^*$
3. Finally, any joint-velocity vector obtained by adding to $\dot{q}$ a component belonging to the null input space will save the problem.
   $\Rightarrow \quad \dot{q}$ such that $\dot{q} = J^{\boxtimes}(q,p) \cdot \dot{x}^* + (I - J^{\boxtimes}(q,p)\, J(q,p)) \cdot \dot{z}$; $\forall\, \dot{q} \,\epsilon\, \Re^n$

Usually we choose the *minimal norm solution*, which is the one with $\dot{\bar{q}} = 0$.

For the evaluation of the error committed, we suppose that we want to achieve a certain output velocity $\dot{x}^*$, that however has a non-null component along the unfeasible direction. Obviously $\dot{x}^*$ cannot be reached exactly and we can only approximate it.
In particular the error committed will be:

$$\dot{\bar{x}} = \dot{x}^* - \dot{\hat{x}} = \dot{x}^* - J\,\dot{q} = \dot{x}^* - J\,(\dot{\hat{q}} + \dot{\bar{q}}) = \dot{x}^* - J\,\dot{\hat{q}} - J\,\dot{\bar{q}} = \dot{x}^* - J\,J^*\,\dot{x}^*$$

with $J\,\dot{\bar{q}} = 0$.

If we assume that the goal does not change, we can choose two gains, in this case $\alpha_A = \alpha_L = \alpha = 0.2$ such that the total end-effector velocity results $\dot{x} = \|error_A \cdot \alpha \;\; error_L \cdot \alpha\|$. Then the joint variables $\dot{q}$ to reach the goal will be given by the pseudo-inverse of the overall task Jacobian:

$$\dot{q} = {J_{b,s}}^{\boxtimes} \cdot \dot{x}$$

$\dot{q}$ are the desired joint velocities.

3.1 Q2

I create the rotation matrix to make the translation matrix from the base to the end-effector. For this exercise, I have the initial configuration q $= [0.0167305, -0.762614, -0.0207622, -2.34352, -0.0305686, 1.53975, 0.753872]$ and this end-effector's goal bOge $= [0.6; 0.4; 0.4]$ with a rotation along the z axis with $\theta = -\pi/4$ (Figure 2).
Inside the for loop, it prints 61 frames from the initial configuration to the final configuration, this number depends by the *simulation parameters* defined at the beginning of the exercise (Figure 3).
Every time, I calculated the transformation matrix with the *q*, then the Jacobian matrix. After that I calculate the angular and linear errors and the reference velocities $\dot{q}$ with the Jacobian matrix and the gain (α); how I explained at the start of this chapter.

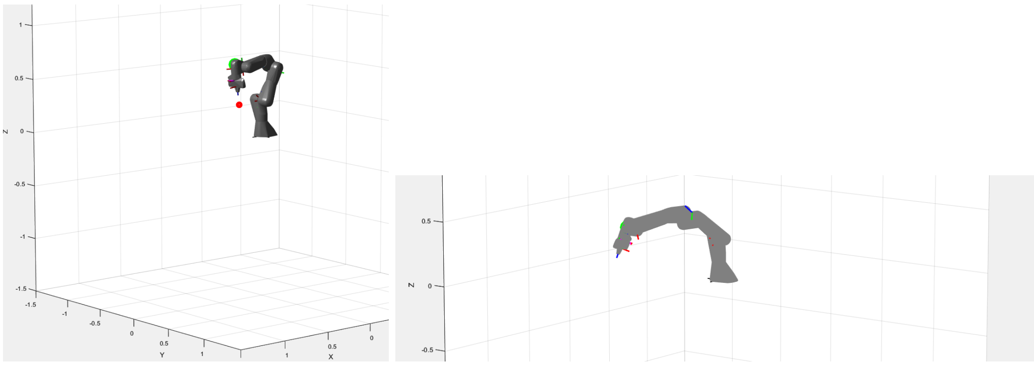

Figure 4: Robot plotting, with the robot 's structure. It starts from initial q to goal q. I added to the plot loop a clean function, for that reason the robot at the end "lose" its 3D configuration.

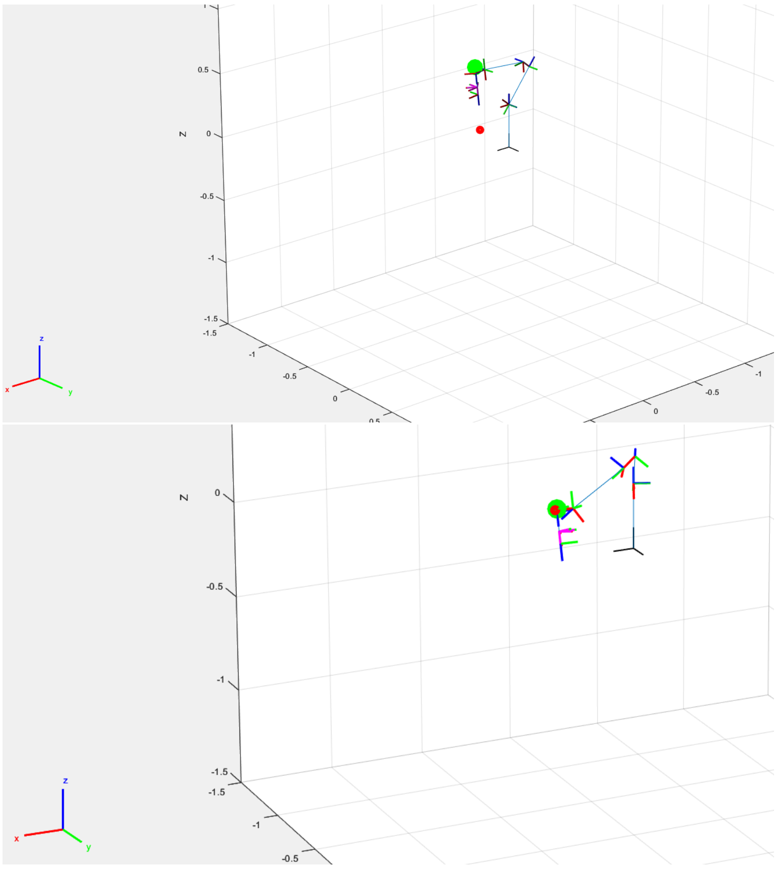

Figure 5: Robot plotting, with only the frames and links. It does the same movement to respect Figure 4, but here you can see the green and red dots (green is end-effector position and red is goal position) are in the same position at the end.

3.2 Q3

In this last point, I change only few things to respect the *Q2 point*. The difference between *Q2* and *Q3* is that in the first the robot has to go on the goal with its end-effector and in the second it has to go on the same goal with a tool. The tool has this characteristic; θ = 0, so its axis is equal to the end-effector'axis and configuration is $[0, 0, 0.2]$, so it is a "stick" along the z axis. To calculate the matrix from the base to the tool I have to implement into the loop cycle the rigid body transformation matrix from e-e frame to rigid-tool frame projected on the base.

$$RBT\_eet = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & C & -B & 1 & 0 & 0 \\ -C & 0 & A & 0 & 1 & 0 \\ B & -A & 0 & 0 & 0 & 1 \end{bmatrix}$$

and $[A, B, C]$ are: $bTe(1:3, 1:3) \cdot eTt(1:3, 4)$.
With *bTe* is rotation matrix from the base to the end-effector and *eTt* is the tools' configuration ($[0, 0, 0.2]$). To put A, B, C in the RBT_eet in order (like you can see over), I used a *Skew* function with a minus before it.
And then I calculate the Jacobian matrix from base to the rigid-tool (bJt).

$$bJt = RBT\_eet \cdot bJe$$

3.3 Q3.6

At that point, I changed the goal with: $bTg = \begin{bmatrix} 0.9986 & -0.0412 & -0.0335 & 0.6 \\ 0.0329 & -0.0163 & 0.9993 & 0.4 \\ -0.0417 & -0.9990 & -0.0149 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and I started the simulation how you can see in the *Figure 8*.

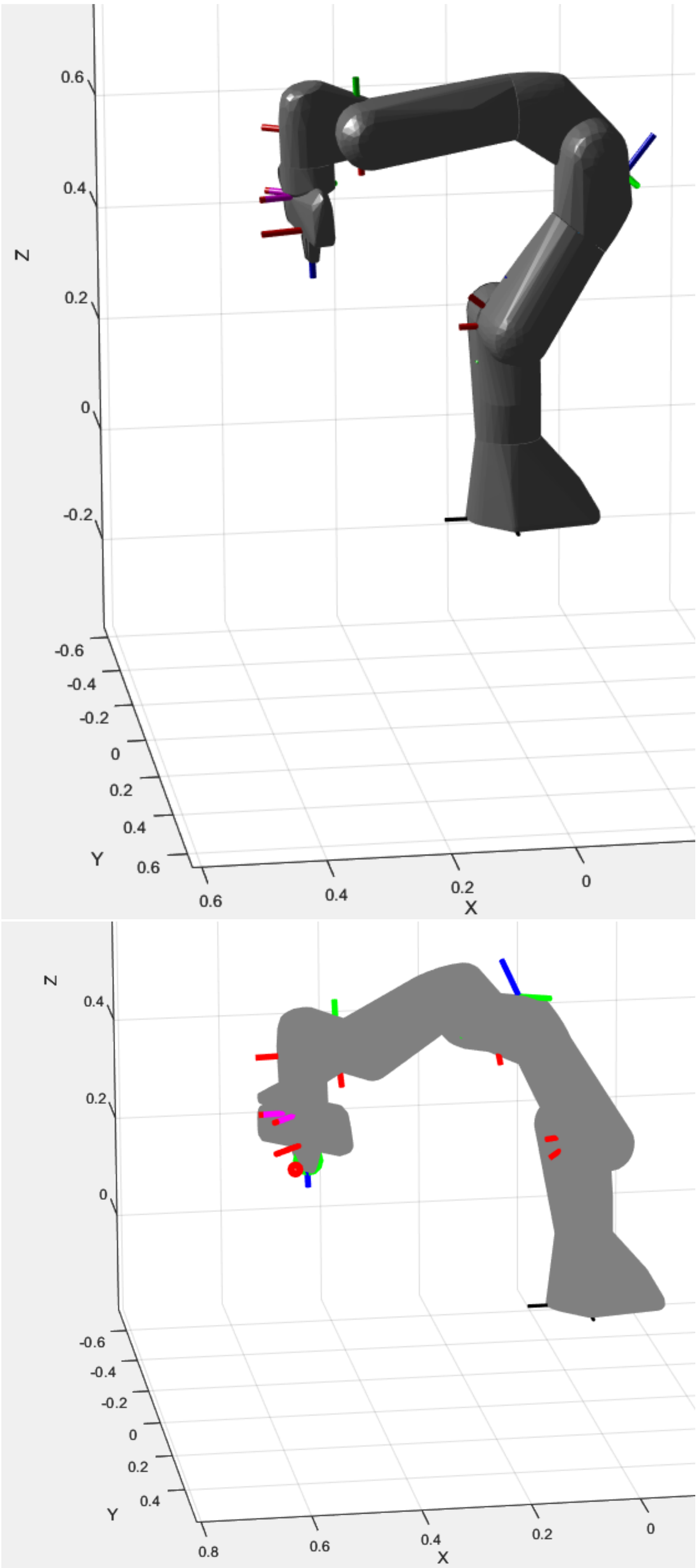

Figure 6: Robot plotting, with the robot's structure. It has the same initial configuration of *Figure 4* but the goal is reached with the tool.

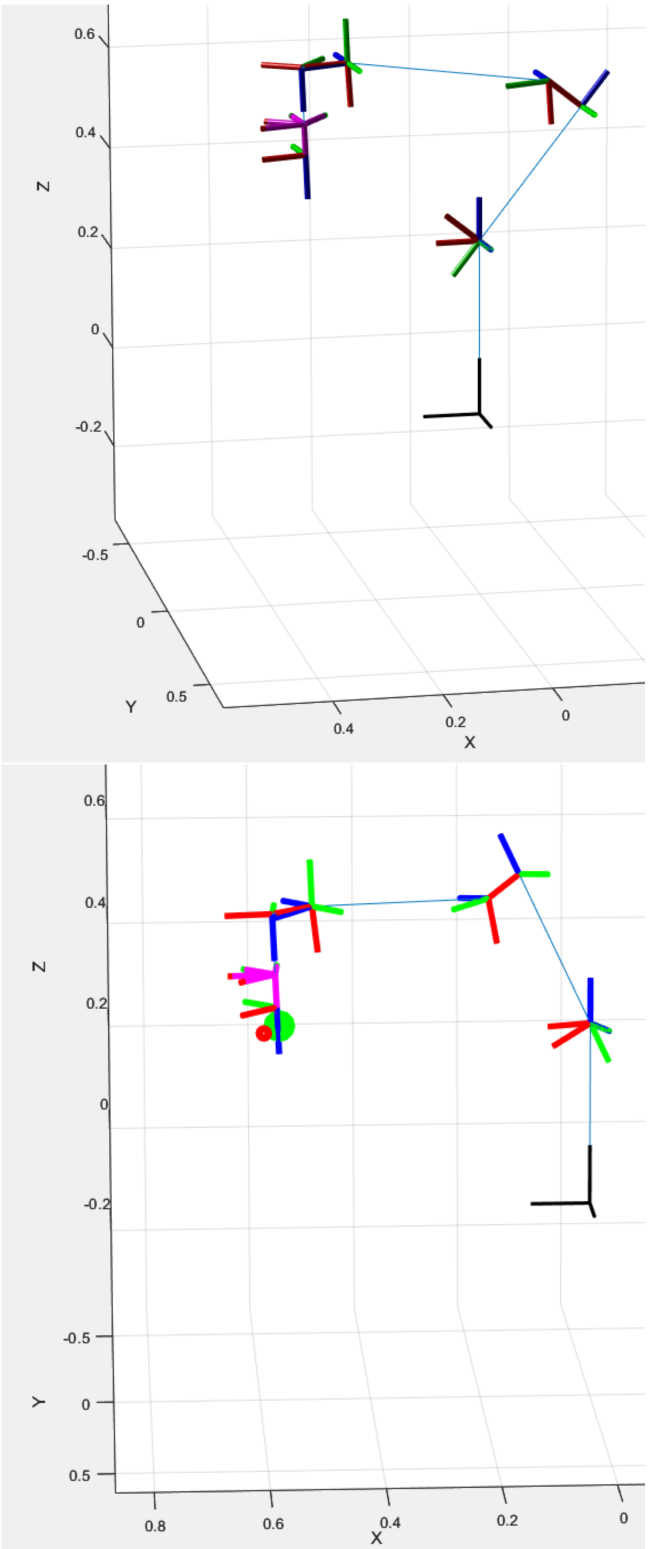

Figure 7: Robot plotting, with only the frames and links. It does the same movement to respect Figure 6, but here you can see the green and red dots (green is end-effector position and red is goal position) are in the same position at the end.

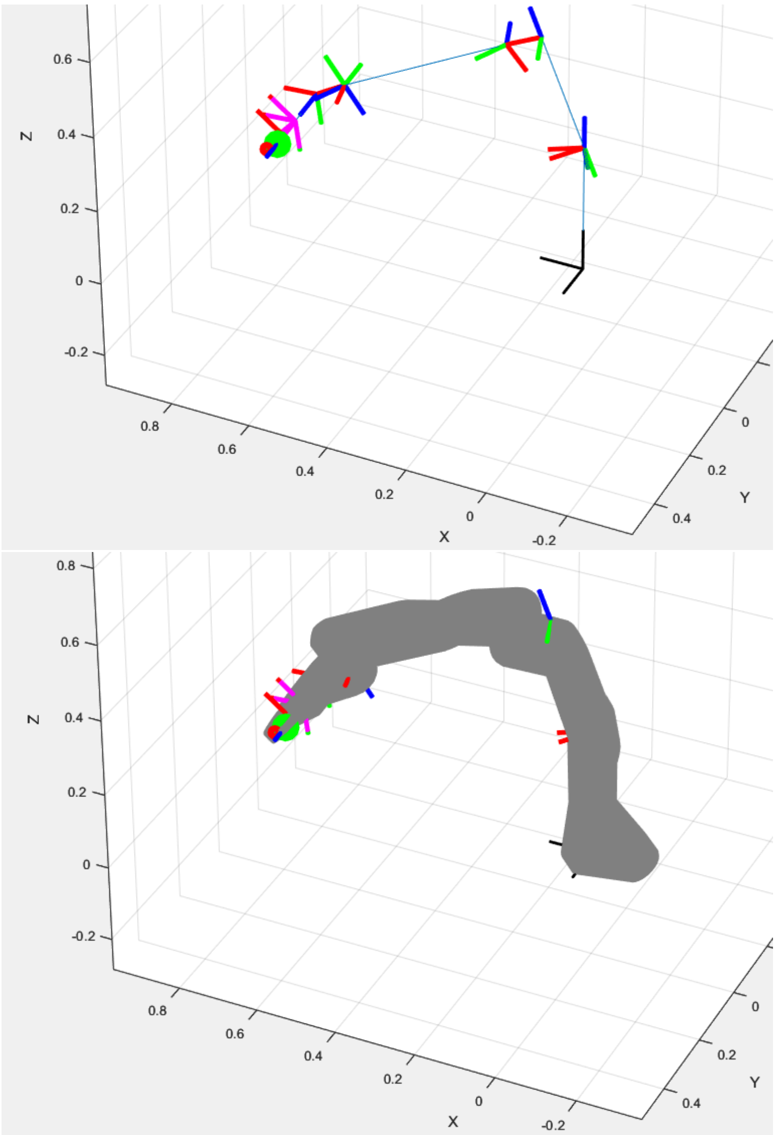

Figure 8: Robot plotting, final configuration Q3.6. I do not print the start configuration because it is the same of *Figure 6 and 7*. It does different movements respect to those this two figures and it finish with a position not perpendicular like they.

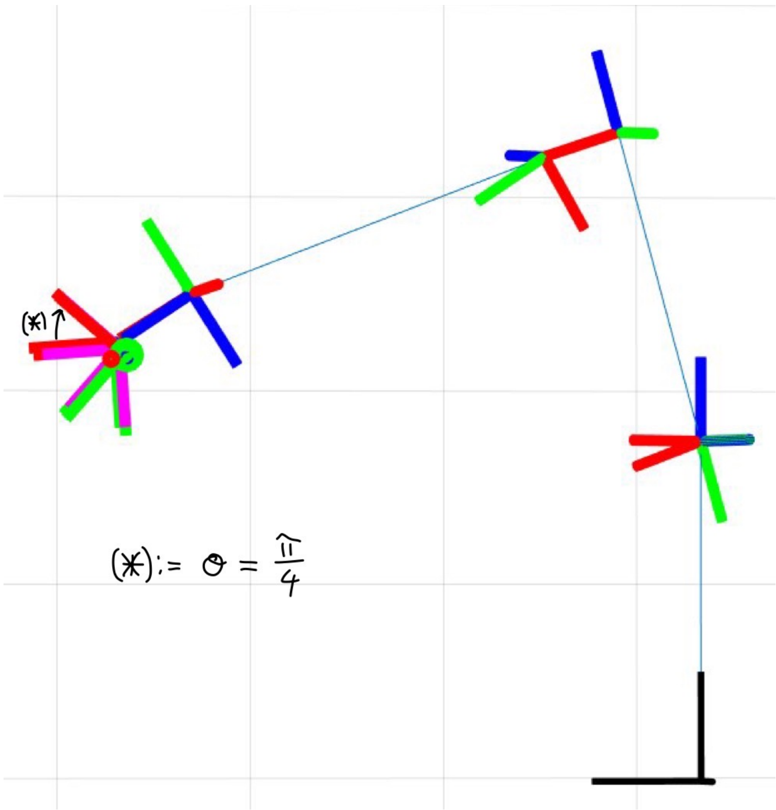

Figure 9: Configuration end-effector and tool. In this figure you can see the particular rotation $\theta = pi/4$ along the z axis.

4 Appendix

[Comment] Add here additional material (if needed)

4.1 Appendix A

4.2 Appendix B