UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

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

MODELLING AND CONTROL OF MANIPULATORS

Second Assignment

Manipulator Geometry and Direct Kinematics

Author:

Colomba Ilaria
Ierardi Ambra
Malatesta Federico

Student ID:

s4829201
s4843302
s4803603

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo
George Kurshakov

January 24, 2024

Contents

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

Table 1: Nomenclature Table

1 Assignment description

The second assignment of Modelling and Control of Manipulators focuses on manipulators geometry and direct kinematics.

- Download the .zip file called MOCOM-LAB2 from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling the predefined files called ”*main.m*”, ”*BuildTree.m*”, ”*GetDirectGeometry.m*”, ”*DirectGeometry.m*”, ”*GetTransformationWrtBase.m*”, ”*GetBasicVectorWrtBase.m*” and ”*GetFrameWrtFrame.m*”.
- Write a report motivating your answers, following the predefind format on this document.

1.1 Exercise 1

Given the following CAD model of an industrial 7 dof manipulator:

Q1.1 Define all the model matrices, by filling the structures in the *BuildTree()* function. Be careful to define the z-axis coinciding with the joint rotation axis, and such that the positive rotation is the same as showed in the CAD model you received. Draw on the CAD model the reference frames for each link and insert it into the report.

Q1.2 Implement a function called *DirectGeometry()* which can calculate how the matrix attached to a joint will rotate if the joint rotates. Then, develop a function called GetDirectGeometry() which returns all the model matrices given the following joint configurations:

- $\mathbf{q} = [0, 0, 0, 0, 0, 0, 0]$.
- $\mathbf{q} = [0, 0, 0, 0, 0, \pi/2, 0]$.
- $\mathbf{q} = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$.
- $\mathbf{q} = [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$.

Comment the results obtained.

Q1.3 Calculate all the transformation matrices between any two links, between a link and the base and the corresponding distance vectors, filling respectively: *GetFrameWrtFrame()*, *GetTransformationWrtBase()*, *GetBasicVectorWrtBase()*

Q1.4 Given the following starting and ending configuration:

- $\mathbf{q}_i = [0, 0, 0, 0, 0, 0, 0]$ and $\mathbf{q}_f = [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$
- $\mathbf{q}_i = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$ and $\mathbf{q}_f = [0, 0, 0, 0, 0, 0, 0]$
- $\mathbf{q}_i = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$

Plot the intermediate link positions in between the two configurations (you can use the plot3() or line() functions) and comment the results obtained.

Q1.5 Test your algorithm by changing one joint position at the time and plot the results obtained for at least 3 configurations.

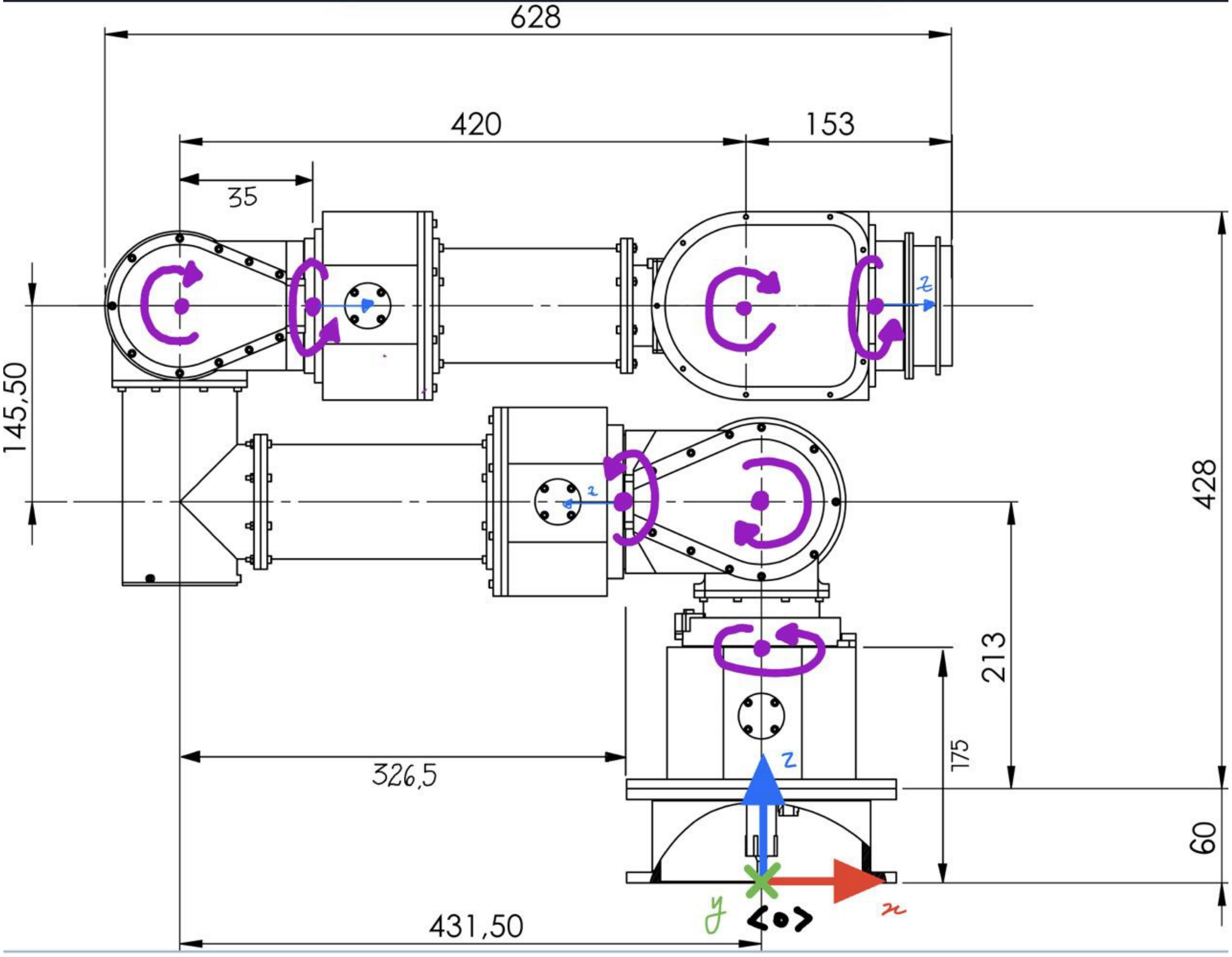


Figure 1: CAD model of the robot

2 Exercise 1

[Comment] For each exercise report the results obtained and provide an explanation of the result obtained (even though it might seem trivial).
The purpose of this project is to analyze of an **industrial robot** with 7 degrees of freedom (*DOF*).

2.1 Q1.1

The first request of the problem is to define all the matrices of the model: this computation is performed using a function called **BuildTree()**, in which each **transformation matrix** is defined.
The model of the matrix is defined as follows:

$$^i_jT = \begin{bmatrix} ^i_jR & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix}$$

where ^i_jR is the rotation matrix *3x3* of each link that describes the rotation of each link with respect to the previous one and **r** is the position vector *3x1* that describes the position of each link with respect to the previous one, while *i* and *j* are the indexes which describe respectively the previous joint and the next one. **0** is a *1x3* vector of zeros.
By computing all of these quantities for each link, a three dimensional matrix is obtained, which contains the transformation matrices for each link. It is possible to see by the transformation matrices obtained that the three rows combined with the first three columns indicate the rotational matrix, whereas the first three rows combined with the last column indicates the position of the frame with respect to the corresponding axis. ^i_jT, overall, represents the transformation matrix between the i-th and j-th frames. The transformation matrices for each link are the following:

- for the first link, it is:

$$^0_1R = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.175 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- for the link number 2, it is obtained:

$$^1_2R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.098 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- for the link number 3, the matrix is:

$$^2_3R = \begin{bmatrix} 0 & 0 & 1 & 0.105 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- for the link number 4, the result is:

$$^3_4R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -0.1455 \\ -1 & 0 & 0 & 0.3265 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- for the link number 5, it is:

$$^4_5R = \begin{bmatrix} 0 & 0 & 1 & 0.035 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- for the link number 6, the transformation matrix is:

$$
{}^{5}_{6}R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.385 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

- for the last link, the matrix is:

$$
{}^{6}_{7}R = \begin{bmatrix} 0 & 0 & 1 & 0.153 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

The next step of this section was to draw on the *CAD model* of the robot all the reference frames referred to each joint, paying attention on the definition of the *z-axis*, which is important because it coincides with the joint's rotational axis, as the conventional notation suggests to do. In order to do this, it has been decided to use the following convention: the next joint's *z-axis* is parallel to the *x-axis* of the previous one. Found the *z-axis*, it is possible to represent also the *x-axis* and the *y-axis*, using the right hand rule.
The model of the robot we are interested in is shown in the following *CAD model*. The model contains also the frames chosen for each joint.

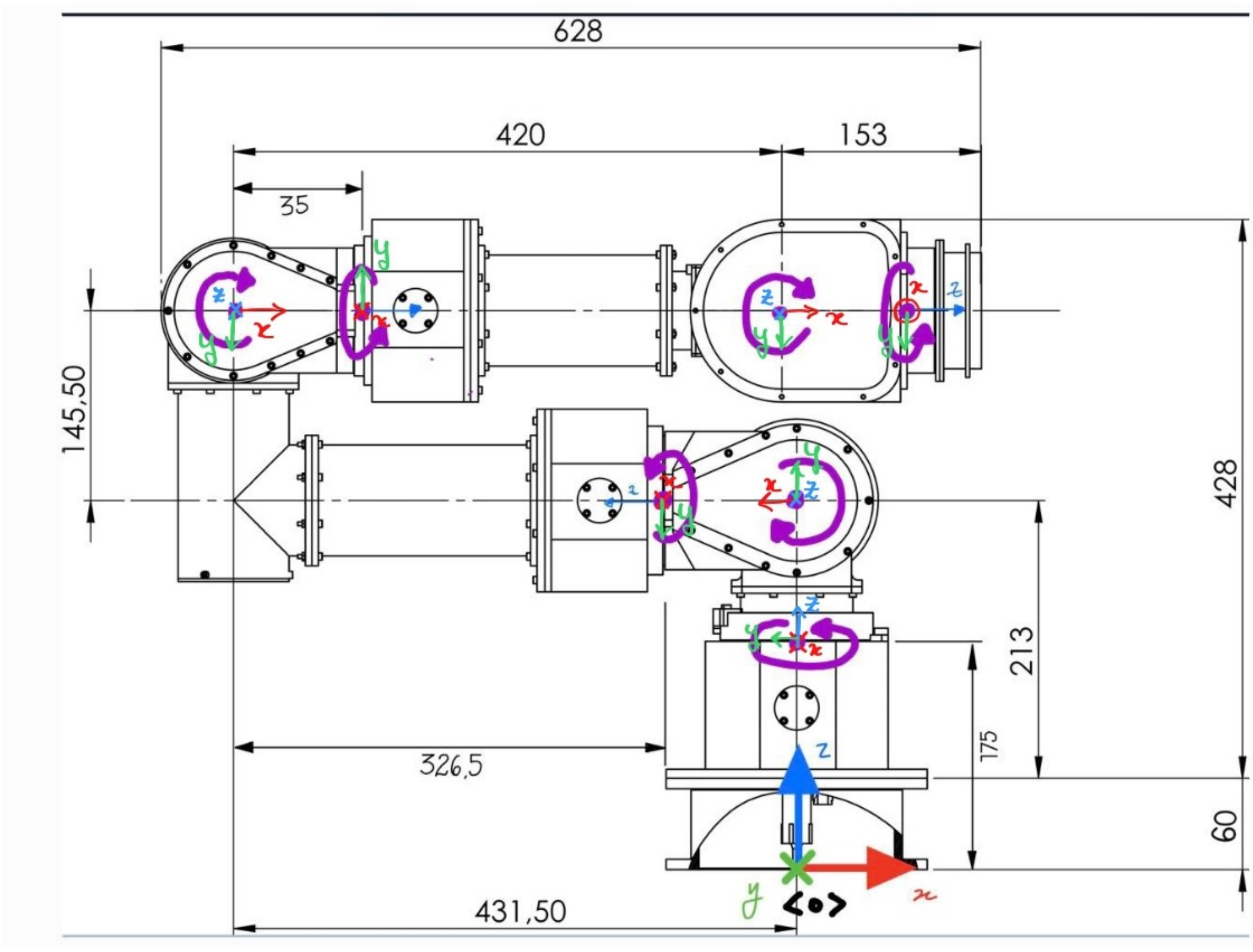


Figure 2: CAD model of the robot, with reference frames

Then, develop a function called GetDirectGeometry() which returns all the model matrices given the following joint configurations:

- **q** $= [0, 0, 0, 0, 0, 0, 0]$.
- **q** $= [0, 0, 0, 0, 0, \pi/2, 0]$.

- **q** $= [0, \pi/2, 0, -\pi/2, 0, 0, 0]$.
- **q** $= [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$.

2.2 Q1.2

The second request of the assignment is to compute how the matrix attributed to a joint changes if the joint rotates. In order to do this, the function **DirectGeometry()** was computed. This function takes in input:

- the current link position represented by the vector q_i which contains the joint variables (angles in case of rotational joints and length in case of the prismatic ones, even if in this case they are not present);
- the constant transformation matrix ${}^i_jT$ between the base of the link $< i >$ and the following one;
- the type of link, represented by a number: *0* for the *revolute joints* and *1* for the *prismatic* ones.

The output of the function is a transformation matrix ${}^i_jT_q$ which represents the transformation between the base of the joint $< i >$ and its following frame, taking into account the actual rotation/translation of the joint itself. The function computes the output in this way:

- if the joint is **rotational** there is a rotation around the z-axis. In this case, the $\mathbf{r}_q$ vector contained in the ${}^i_jT_q$ output matrix is equal to the **r** vector which is in the ${}^i_jT$ input matrix. The ${}^i_jR_q$ matrix contained in the ${}^i_jT_q$ matrix is computed by multiplying the ${}^i_jR$ matrix of ${}^i_jT$ with the rotation matrix:

$$R_z(q_i) = \begin{bmatrix} \cos(q_i) & -\sin(q_i) & 0 \\ \sin(q_i) & \cos(q_i) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

  In formulas:

$$\mathrm{r}_q = \mathrm{r}$$
$${}^i_jR_q = {}^i_jR * R_z(q_i)$$

  So that:

$${}^i_jT_q = \begin{bmatrix} {}^i_jR * R_z(q_i) & \mathbf{r} \\ \mathbf{0} & 1 \end{bmatrix}$$

- if the joint is a **prismatic** one, we have a translation along the z-axis and so we have to add the translation vector in the translation part of the matrix ${}^i_jT_q$. In formulas:

$$\mathbf{r_q} = \mathbf{r} + q_i * \mathbf{k}$$

  where **k** is the last column of the ${}^i_jR$ matrix, and

$${}^i_jR_q = {}^i_jR$$

  so that:

$${}^i_jT_q = \begin{bmatrix} {}^i_jR* & \mathbf{r} + q_i * \mathbf{k} \\ \mathbf{0} & 1 \end{bmatrix}$$

After that, the function **GetDirectGeometry()** was created in order to calculate all the model matrices given the following joint configurations: **q** $= [0, 0, 0, 0, 0, 0, 0]$. **q** $= [0, 0, 0, 0, 0, \pi/2, 0]$. **q** $= [0, \pi/2, 0, -\pi/2, 0, 0, 0]$. **q** $= [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$.

It returns all the model matrices given the joint configuration **q**.
This function takes as inputs:

- the current links position *q*;
- the vector of matrices containing the transformation matrices ${}^i_jR$ from joint i to joint j;
- the vector whose size is the number of links, identifying the joint type: 0 for revolute joints, 1 for prismatic joints, as previously mentioned.

The output of the function is the vector of matrices ${}^i_jT_q$ containing the transformation matrices from link i to link j for the input **q**. These transformation matrices of each link are computed recalling, for each link, the function *DirectGeometry()* created in the previous part of the assignment.

2.3 Q1.3

The third task of the project was to compute all the transformation matrices between any two links, between a link and the base and the corresponding distance vectors. In order to do this three new functions were created: **GetFrameWrtFrame()**, **GetTransformationWrtBase()**, **GetBasicVectorWrtBase()**.
The function **GetFrameWrtFrame()** gets in input:

- the number of the $i-th$ link;
- the number of the $j-th$ link;
- the vector of matrices ${}^{b}_{e}T_i$ containing the transformation matrices from link i to link $i+1$ for the current q. ${}^{b}_{e}T_i$ is called *BiTei* in the *Matlab* code.

The output of this function is the transformation matrix ${}^{i}_{j}T$ between link i and link j for the configuration described in ${}^{b}_{e}T_i$.
The computation of the output comes from the multiplication of the transformation matrices of each link between link i and link j. The function works in the following way:

- if the $i-th$ link is smaller than the $j-th$ one, the matrix is computed this way:

$${}^{i}_{j}T = {}^{i}_{j}T * {}^{b}_{e}T_i$$

;

- otherwise, if the $i-th$ link is bigger than the $j-th$ one, the function applies proceeds in an alternative way:

$${}^{i}_{j}T^{-1} = {}^{i}_{j}T = \begin{bmatrix} {}^{i}_{j}R^{T} & -{}^{i}_{j}R^{T} *^{a} \mathbf{O}_b \\ \mathbf{0}_{3x1} & 1 \end{bmatrix}$$

The vector ${}^{a}\mathbf{O}_b$ is the last column of ${}^{b}_{e}T_i$ and the ${}^{i}_{j}R^{T}$ is the transpose of the ${}^{i}_{j}R$ matrix inside ${}^{b}_{e}T_i$.
The function **GetTransformationWrtBase()** take as inputs:

- ${}^{b}_{e}T_i$ which is the vector of matrices containing the transformation matrices form link i to link $i+1$ for the current joint position **q**;
- the number of link for which computing the transformation matrix.

The output of the function is the transformation matrix ${}^{b}_{i}T$ from the robot's base to the $i-th$ joint in the configuration identified by ${}^{b}_{e}T_i$.
In order to compute the output, the function works multiplying the transformation matrices of the links from the base to the specified link. In formula: ${}^{b}_{i}T = {}^{b}_{i}T * {}^{b}_{e}T_i$
The function **GetBasicVectorWrtBase()** takes as inputs:

- the transformation matrix ${}^{b}_{e}T_i$ in between frame i and frame j;
- the number of the analyzed link;

The output of the function is the vector **r** which is the vector from frame i to the robot's base frame $< O >$. In order to obtain the output, the function computes the transformation matrix from the base to the specified link, calling the function **GetTransformationWrtBase()**. Then it extracts the translation part of the transformation matrix which corresponds to the first three rows of the last column of it, and this is the vector **r**.

2.4 Q1.4

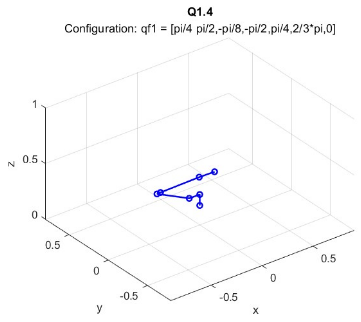
In order to plot the intermediate link positions between two configurations it's necessary to compute the difference between the two vectors (initial and final) and then divides it for the $numberOfSteps$.
In this way it's created a vector containing the variation of every link for each step $qstepi$ and it's copied to $qstepi\_reference$ to maintain a vector containing all the Δs with

$$\mathbf{i} = [1,2,3]. \tag{1}$$

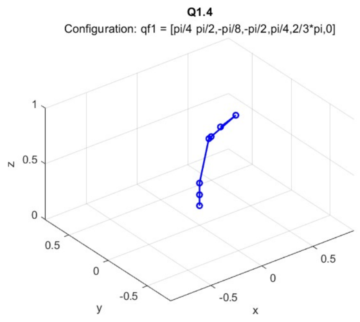
To plot the intermediate link position a loop simulating the movement of a robot has been used.
At each step of the movement, it first calculates the transformation matrix for that step using $GetDirectGeometry$. This matrix represents how much the robot's joints should rotate or translate at this step to go from the initial

position to the final position.
Then, for each link in the robot, it calculates the position of that link with respect to the base of the robot by means of $GetBasicVectorWrtBase$, exploiting the transformation matrix calculated earlier.
Next, it plots the position of the link. When this is done for all links, it effectively shows the position of the entire robot at this step of the movement.
After plotting, it updates the current step, representing the movement forward in time.

In the first configuration, the initial position of the joints is equal to $\mathbf{q}_i = [0, 0, 0, 0, 0, 0, 0]$, and a final one of $\mathbf{q}_f = [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$.



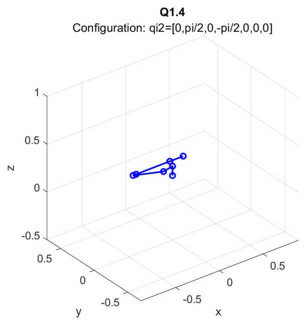
(a) First configuration



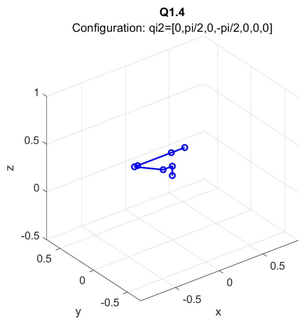
(b) Last configuration

Figure 3: First case

In the second case, the initial disposition of the joints is equal to $\mathbf{q}_i = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$ and the final one is corresponding to $\mathbf{q}_f = [0, 0, 0, 0, 0, 0, 0]$.



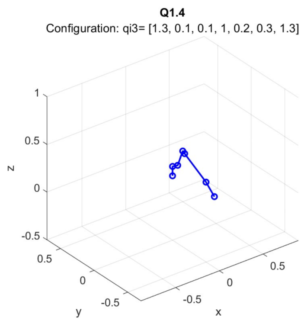
(a) First configuration



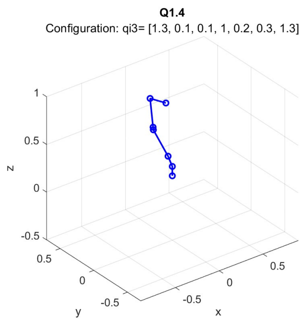
(b) Last configuration

Figure 4: Second case

It is possible to observe that the initial configuration is very similar to the one from the previous case: that is because only two angles are changing, namely the second and the fourth.
For the last example, it was chosen a n initial position of the joints equal to $\mathbf{q}_i = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and the final one corresponding to $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$.



(a) First configuration



(b) Last configuration

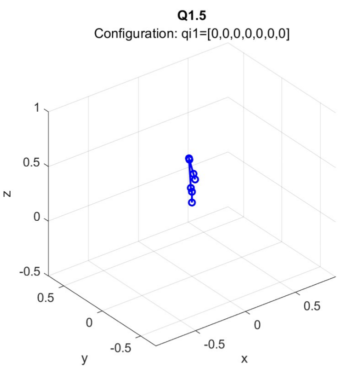
Figure 5: Last case

2.5 Q1.5

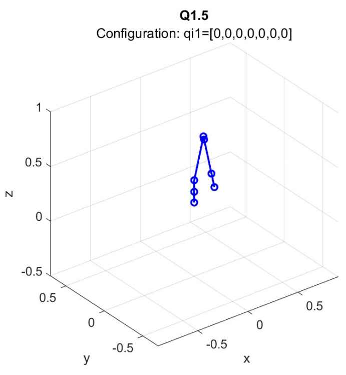
The algorithm previously implemented was further tested plotting the transformation from the initial configuration to the final one, moving one joint at the time. In order to do so, it was necessary to modify some parameters used: first, for each link it was created a vector consisting in the step increment of the position of the interested joint, which has components equal to zero for each other link. For example, for the first joint, the step vector has the following shape:

$$q\_step = \begin{bmatrix} \Delta & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

where Δ represents the initial increment from the initial configuration. At each step, the position vector is incremented of the step quantity: this way, with the function **GetBasicVectorWrtBase.m**, the vector connecting each joint to the base is obtained, and so it is possible to draw them with the $plot3()$ $MatLab$ function, as in the previous task. The initial configurations are equal to the ones showed in the *subsection 1.4*, so they won't be re-proposed.
For sake of clearness, some examples are shown in the following images:



(a)



(b)

Figure 6: First configuration, intermediate steps

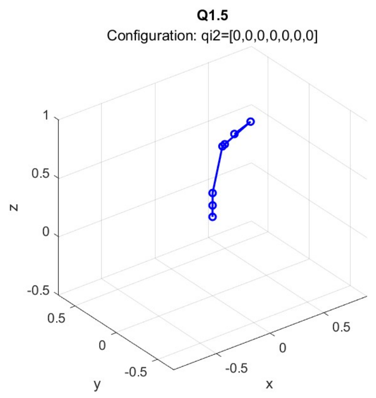
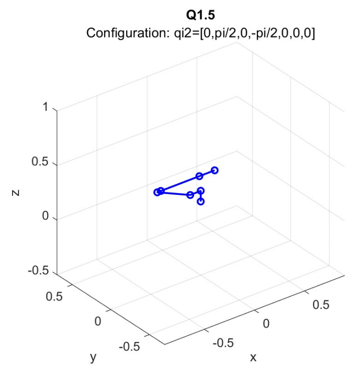
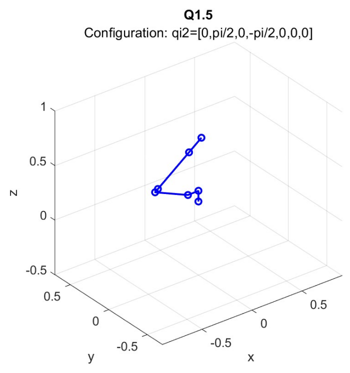


Figure 7: First configuration, final position

In these images, it is possible to notice two frames of the first configuration, starting from an initial position of the joints equal to $\mathbf{q}_i = [0, 0, 0, 0, 0, 0, 0]$, and a final one of $\mathbf{q}_f = [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$, so the only joint which stays still is the last one.



(a)



(b)

Figure 8: Second configuration, intermediate steps

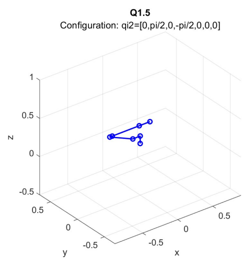
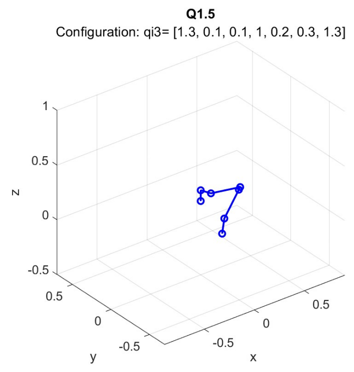
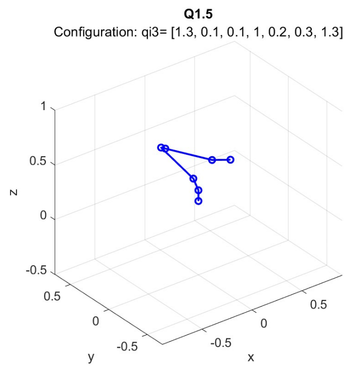


Figure 9: Second configuration, final position

In these two pictures, it is possible to notice two steps of the configuration with original vector of the joints equal to $\mathbf{q}_i = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$ and the ending one corresponding to $\mathbf{q}_f = [0, 0, 0, 0, 0, 0, 0]$, so in this case the only joints moving are the second and the fourth.



(a)



(b)

Figure 10: Third configuration, intermediate steps

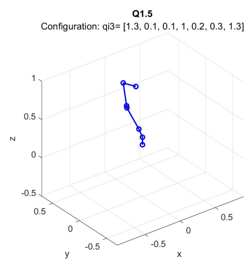


Figure 11: Third configuration, final position

In the last images, there is the structure in different time instants, with original configuration of $\mathbf{q}_i = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and the final one of $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$. In this case, all the joints are moving.

In the end, it is possible to observe how in all the three cases, whether moving all the joints together or moving one at the time, the final configuration is the same, while the intermediate configurations are not the same, since they follow different paths.