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:

Di Donna Alberto
Dondero Enrico

Student ID:

s4944656
s4938123

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo
George Kurshakov

November 4, 2024

Contents

1 Assignment description **3**
1.1 Exercise 1 . . . 3

2 Exercise 1 **5**
2.1 . . . 5
2.2 . . . 5
2.3 . . . 6
2.4 . . . 7
2.5 . . . 8

3 Appendix **9**
3.1 Appendix A . . . 9

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.

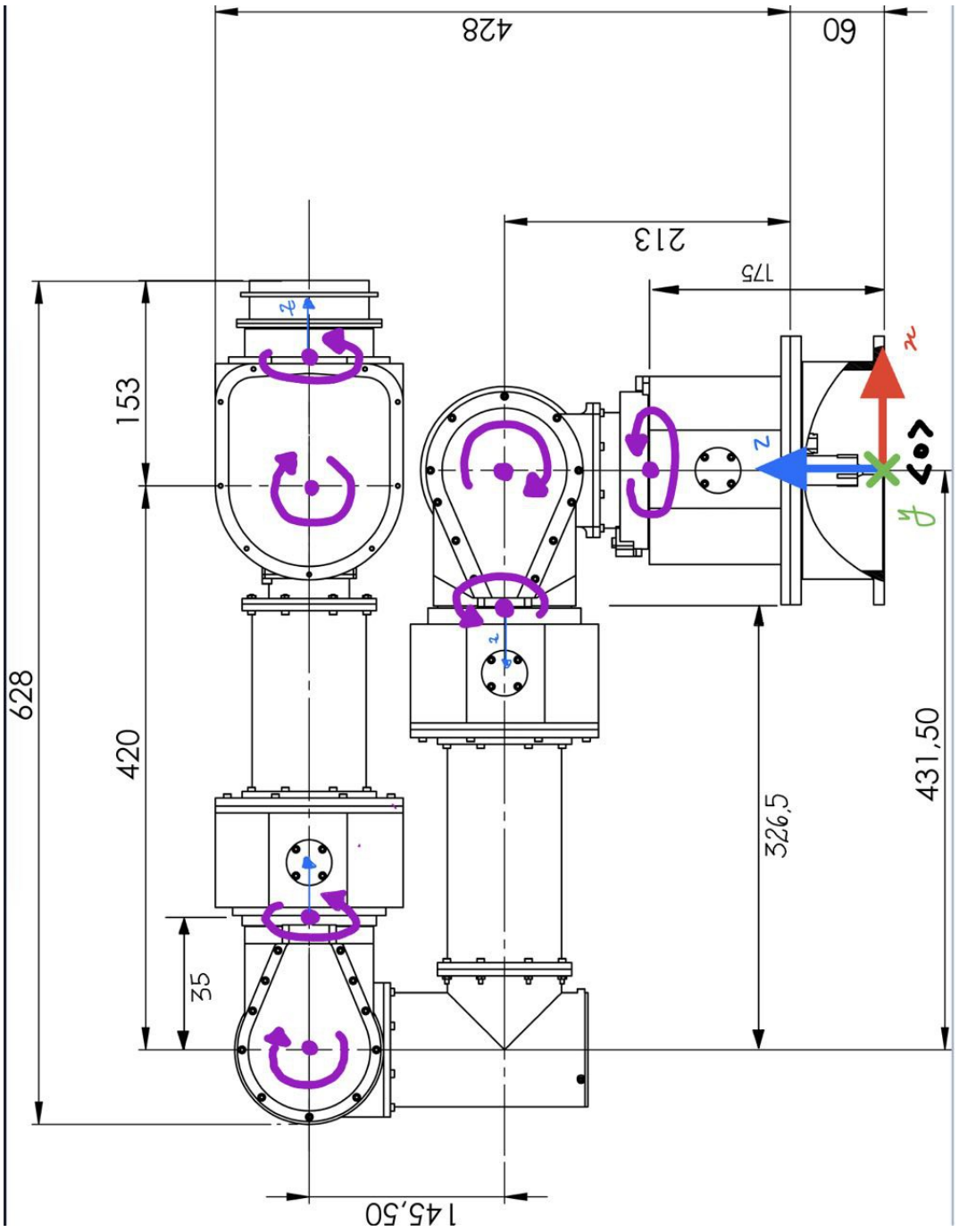


Figure 1: CAD model of the robot

2 Exercise 1

This exercise was about designing a model for a 7-Dof robot arm and performing simulations on its capabilities in terms of direct kinematics.

2.1

This part was about defining the position and the orientation of the reference frames for all the joints in the chain. As can be seen in appendix A every frame was placed at the position of the joints.
To define the matrices in Matlab, it was necessary to implement the *BuildTree* function. This does not need any arguments, but it is only used to generate the model for the robot arm as an array of transformation matrices, each one defined taking into consideration only its previous joint.

$$ {}^{0}_{1}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 175 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 98 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{3}T = \begin{bmatrix} 0 & 0 & -1 & -105 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{3}_{4}T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & -145.5 \\ -1 & 0 & 0 & 326.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{4}_{5}T = \begin{bmatrix} 0 & 0 & 1 & 35 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 385 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{6}_{7}T = \begin{bmatrix} 0 & 0 & 1 & 153 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

2.2

The *DirectGeometry* function has been developed to remain hidden as it was only called inside the *GetDirectGeometry* function. This was because the former was used to compute the forward geometry given the transformation matrix of a particular joint w.r.t the previous, the joint variable q, and the type of the joint (rotational or prismatic), used to differentiate between a rotation or a translation w.r.t the z-axis.
The latter was given the configuration tree as input, together with the joint variables vector **q** and the joint type vector. Its job was to apply the vector of joint variables to the starting configuration tree using the *DirectGeometry* function.

1.2.a
The first array of joint variables as expected produces no rotation whatsoever, the resulting configuration is equal to the starting one.

1.2.b
The array $\mathbf{q} = [0, 0, 0, 0, 0, \pi/2, 0]$, produces a rotation of the sixth joint by 90°. Below are shown the initial transformation matrix for this joint and the final one:

$$ {}^{6}_{5}T_i = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 385 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{6}_{5}T_f = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 385 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1} $$

1.2.c Similarly here the vector $\mathbf{q} = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$hints to a rotation of the second and fourth frame by respectively 90° and -90°. The two final transformation matrices are the following:

$$ {}^{1}_{2}T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 98 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{4}_{3}T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & -145.5 \\ 0 & -1 & 0 & 326.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2} $$

1.2.d The last rotation involves every joint except for the last one, the full tree can be seen below:

$$ {}^{0}_{1}T = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1 & 175 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{1}_{2}T = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 98 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{2}_{3}T = \begin{bmatrix} 0 & 0 & -1 & -105 \\ -0.3827 & 0.9239 & 0 & 0 \\ 0.9239 & 0.3827 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{3}_{4}T = \begin{bmatrix} 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & -145.5 \\ 0 & -1 & 0 & 326 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{4}_{5}T = \begin{bmatrix} 0 & 0 & 1 & 35 \\ 0.7071 & 0.7071 & 0 & 0 \\ -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{5}_{6}T = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0.866 & -0.5 & 0 & 0 \\ -0.5 & -0.866 & 0 & 385 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$${}^{6}_{7}T = \begin{bmatrix} 0 & 0 & 1 & 153 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

As it can be seen for every transformation provided, the z component of the rotation matrix does not change, this is because of the choice of the reference frames. Each of them is oriented in such a way that any given rotation or translation is made along this axis.

2.3

To complete this exercise were implemented three functions: *GetFrameWrtFrame*, *GetTransformationWrtBase*, and *GetBasicVectorWrtBase*.

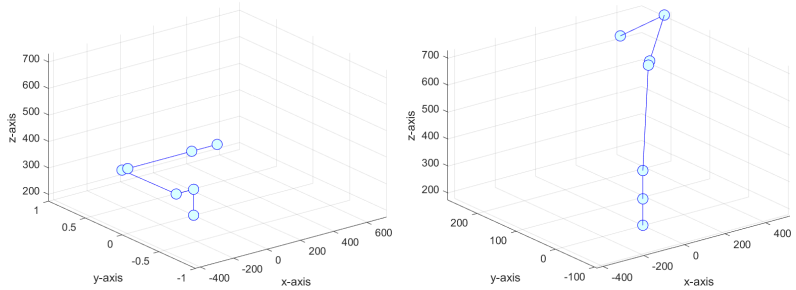
- The function *GetFrameWrtFrame* takes as input the geometric model, the index for the referenced frame and for the referring frame, and returns the transformation matrix of the second frame with respect to the first one.
  Inside the function, we need to check if the number of the first frame is lower than the number of the second one. In this case, we extract the transformation matrices from the geometric model, starting from the one that describes the transformation between the first frame and the next one, until the transformation matrix of the penultimate frame with respect to the last one, that is the second frame taken as input. Then we multiply all the extracted transformation matrices.
  Otherwise, if the number of the first frame is greater than the number of the second one, we extract the transformation matrix between the first frame and the previous one, and then we continue going downward until the transformation matrix between the penultimate frame and the second one taken as input. Now, before multiplying all matrices, we need to invert them, because in the geometric model, we have defined the transformation matrices in the opposite direction.
  Here are some examples: ${}^{1}_{5}T = \begin{bmatrix} 0 & 0 & 1 & -396.5 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 243.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ${}^{4}_{7}T = \begin{bmatrix} 0 & 0 & 1 & 573 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  ${}^{6}_{2}T = \begin{bmatrix} 1 & 0 & 0 & 11.5 \\ 0 & 1 & 0 & 145.5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- The function *GetTransformationWrtBase* takes as input the geometric model and the number of the link for which we want to compute the transformation matrix with respect to the base.
  In this function we extract from the geometric model all the transformation matrices, starting from the one between the base and the first frame, until the one between the link taken as input and its previous link. Finally, we multiply all these matrices to obtain the transformation matrix between the base and the given link.
  ${}^{0}_{1}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 175 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ${}^{0}_{2}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 273 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ${}^{0}_{3}T = \begin{bmatrix} 0 & 0 & -1 & -431.5 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 273 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  ${}^{0}_{4}T = \begin{bmatrix} 1 & 0 & 0 & -431.5 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 418.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ${}^{0}_{5}T = \begin{bmatrix} 0 & 0 & 1 & -396.5 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 418.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ ${}^{0}_{6}T = \begin{bmatrix} 1 & 0 & 0 & -11.5 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 418.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
  ${}^{0}_{7}T = \begin{bmatrix} 0 & 0 & 1 & 141.5 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 418.5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- The function *GetBasicVectorWrtBase* takes as input the geometric model and the number of a link and return the basic vector from this link and the robot base frame.
  Inside this function we call the function *GetTransformationWrtBase*, passing to it the two same input taken by *GetBasicVectorWrtBase*, in order to obtain the transformation matrix between the chosen link and the base. Finally we can extract the vector from the first three rows of the last column of this returned matrix. This function is called in the main inside a for loop that iterate over all links, so at the

end we obtain a matrix, in whose columns there are the basic vectors of the links which have the same number of the column:

$$\begin{bmatrix} 0 & 0 & -431.5 & -431.5 & -396.5 & -11.5 & 141.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 175 & 273 & 273 & 418.5 & 418.5 & 418.5 & 418.5 \end{bmatrix}$$

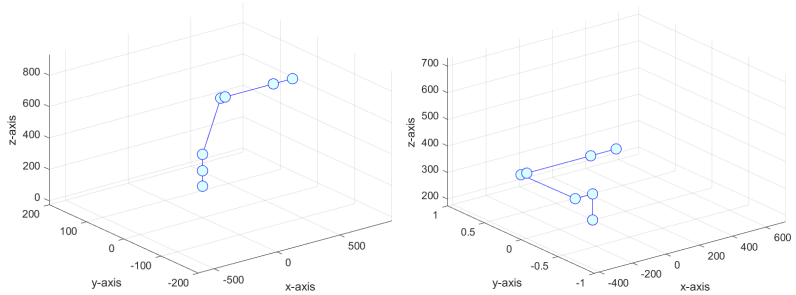
2.4

To perform this task it was chosen to develop a function named *plot_link_chain* , that was used to plot the initial and final configuration while animating the transition between the two.
The function takes as input the initial and final joint variables vector, the configuration tree, and the *linkType* array. It made use of the function *stepJointPosition* to create the three-dimensional matrix to contain the coordinates of the joints w.r.t base for every time-step, given the geometric model obtained for each step with the *GetDirectGeometry*, thus resulting in a 3x7x100 matrix. This is then fed to the plotting part where the starting configuration is plotted generating a *plot3* object and updating the coordinates step after step using the *step()* function.



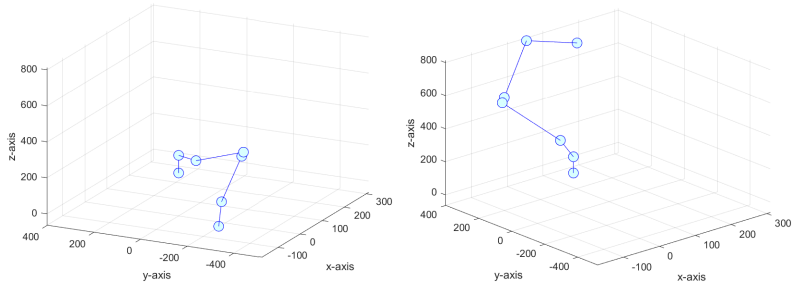
(a) (b)

Figure 2: Transition between initial configuration
at $\mathbf{q} = [0, 0, 0, 0, 0, 0, 0]$ to **q**=$[\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2\pi/3, 0]$



(a) (b)

Figure 3: Transition between initial configuration
at $\mathbf{q} = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$ to **q**=$[0, 0, 0, 0, 0, 0, 0]$

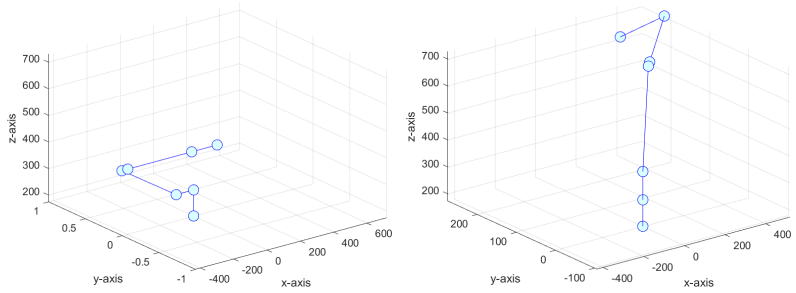


(a) (b)

Figure 4: Transition between initial configuration
at $\mathbf{q} = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ to **q**=$[2, 2, 2, 2, 2, 2, 2]$

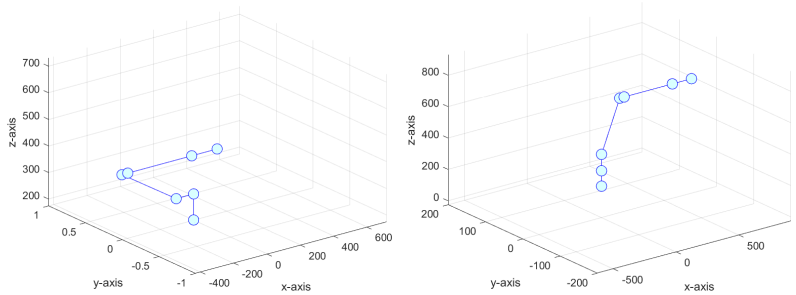
2.5

To complete the last exercise it was implemented a function called *plotJointSteps* that worked similarly to *plot_link_chain*, but with the difference being that the **q** array is split into 7 different ones. Each of them has only the i-th component from the original one and the other elements set to zero. The function was centered around a for loop cycling through the joints. The initial configuration was then updated one joint at a time with the corresponding **q** array using the same structural elements of *plot_link_chain*.
To test the algorithm were used two joint variables arrays from the previous tasks to better understand the contribution of each joint motion, while the third one was $\mathbf{q} = [\pi, \pi/2, 0, -\pi, 0, 0, 0]$, this particular array was chosen to show the full upward extension of the robot arm.



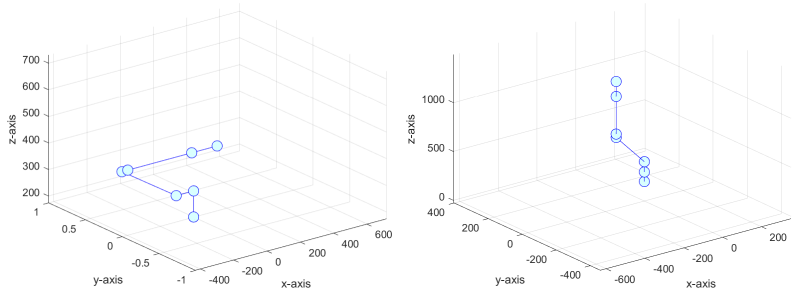
(a) (b)

Figure 5: Transition between initial configuration
at $\mathbf{q} = [0, 0, 0, 0, 0, 0, 0]$ to **q**=$[\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2\pi/3, 0]$



(a) (b)

Figure 6: Transition between initial configuration
at **q**=$[0, 0, 0, 0, 0, 0, 0]$ to $\mathbf{q} = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$



(a) (b)

Figure 7: Transition between initial configuration at **q**=$[0, 0, 0, 0, 0, 0, 0]$ to $\mathbf{q} = [\pi, \pi/2, 0, -\pi, 0, 0, 0]$

3 Appendix

3.1 Appendix A

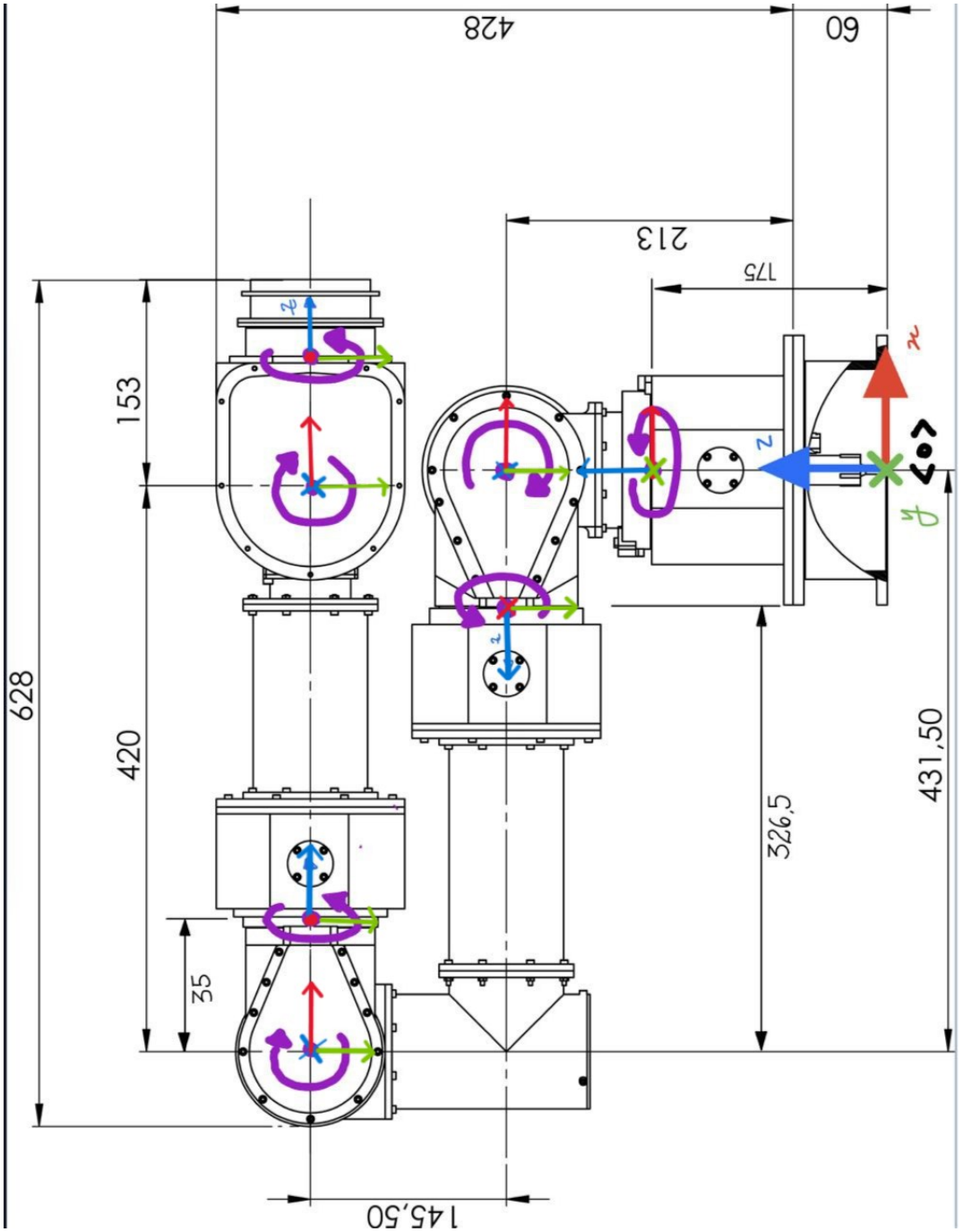


Figure 8: CAD model of the robot with reference frame