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:

Delucchi Manuel
Cappellini Matteo

Student ID:

s4803977
s4822622

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo
George Kurshakov

November 28, 2023

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.

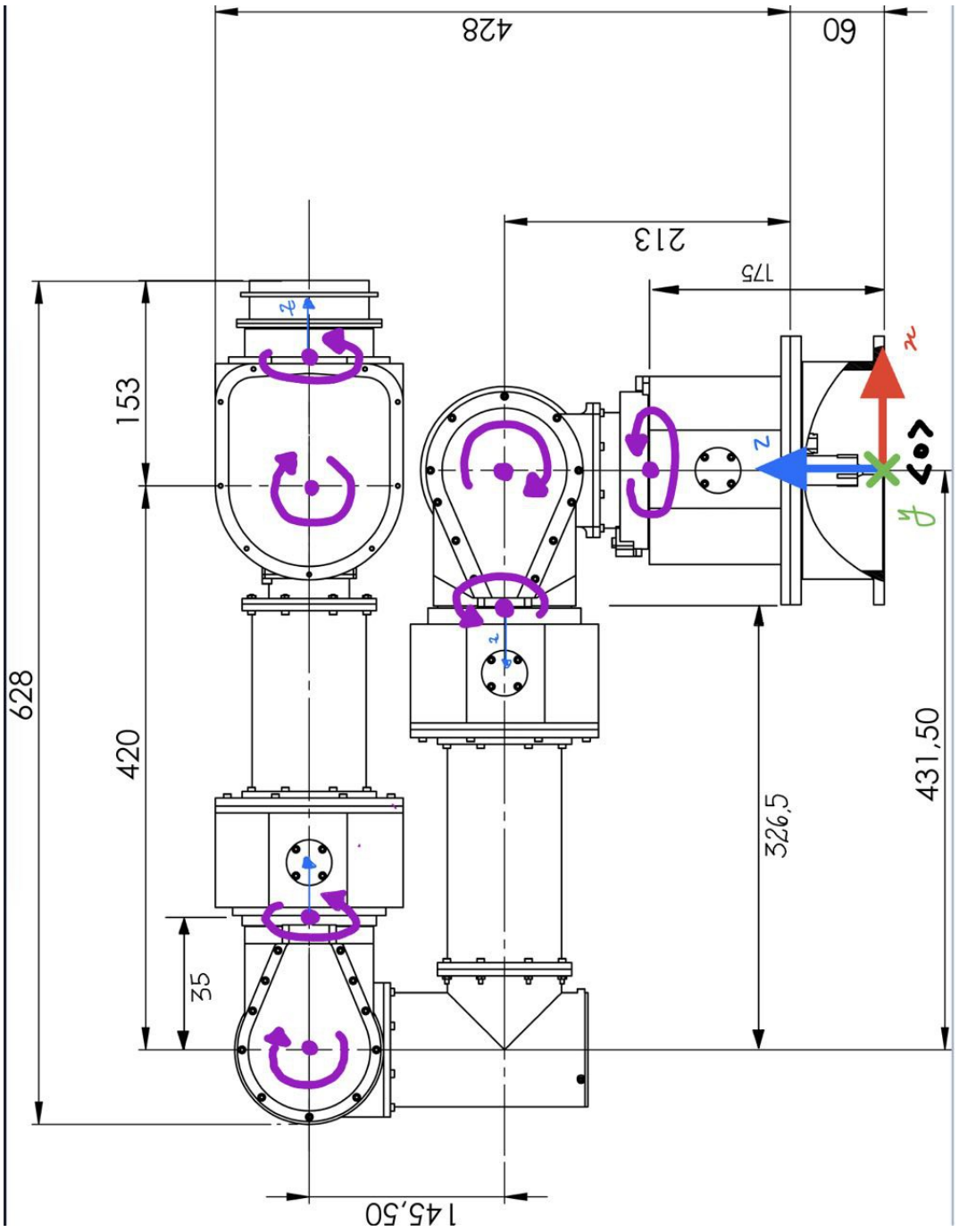


Figure 1: CAD model of the robot

2 Exercise 1

This assignment focuses on the geometry and direct kinematics of an industrial **7-degree-of-freedom (7-DoF) manipulator**. The task involves using MATLAB to model the manipulator's structure using Denavit-Hartenberg (DH) notation and implementing functions to calculate its direct geometric properties.

The assignment involves defining the manipulator's CAD model, setting joint configurations, implementing essential functions and computing transformation matrices between links. It also covers the plot of the links positions over time between given configurations to understand the manipulator's movement and behavior.

2.1 Q1.1

The ***Denavit-Hartenberg (DH)*** parameters serve as a standard to describe kinematic chains in robotic manipulators:

- The ***z-axis*** (blue arrow) aligns with the direction of the joint's rotation axis.
- The ***x-axis*** (red arrow) is perpendicular to both z_n and z_{n-1} axes.
- The ***y-axis*** (green arrow) orientation follows the ***Right-Hand Rule (RHR)*** based on the relationship between the x- and z-axes and such that the positive rotation is the same as showed in the CAD model (purple arrow).

Hence, reference frames are drawn on the CAD model to visually represent the frames for each link:

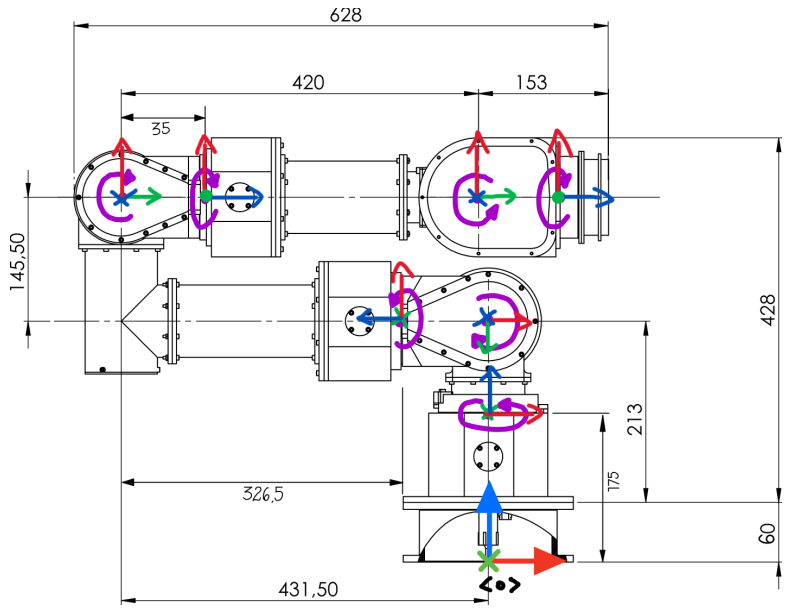


Figure 2: "CAD model of the 7-DoF Industrial Manipulator"

Now, it was necessary to implement the *BuildTree()* function in order to define the transformation matrices for each joint in the industrial 7-DoF manipulator.

Matrices are constructed to represent both orientation and position information using a 4x4 matrix notation, known as ***Transformation Matrix*** of frame $< i+1 >$ with respect to frame $< i >$. These transformation matrices facilitate the computation of a point's position in one frame concerning another.

$$ {}^{i}_{i+1}T = \begin{bmatrix} {}^{i}_{i+1}R & {}^{i}r_{i+1/i} \\ 0_{3x1} & 1 \end{bmatrix} \tag{1} $$

Where ${}^{i}_{i+1}R$ is a rotation matrix filled by watching the change in orientation of the frame $< i+1 >$ with respect to the previous observed frame $< i >$ and $r \in \mathbb{R}^3$ is a column vector representing the position of $< i+1 >$ with respect to $< i >$, obtained using the dimensions highlighted in the CAD model of the manipulator.

The 4x4 transformation matrix, having the first three elements of the fourth row consistently zero and the last one always unitary, allows us to handle transformations in 3D space. In fact, by considering the frame

$< i+1 >$, positioned with respect to $< i >$ within the configuration, we can compute its position ${}^{i}P$ with respect to frame $< i >$ by knowing the position ${}^{i+1}P$ of a generic point P with respect to frame $< i+1 >$, as follows:

$$ {}^{i}P = {}^{i}\, r_{i+1/i} \; + \; {}^{i}_{i+1}R \, {}^{i+1}P $$

As changing the coordinates of projected point from one frame to another, always requires its reference origin to be changed accordingly.

Let us now represent projected points via the so called homogeneous coordinates:

$$ {}^{i+1}\bar{P} = \begin{bmatrix} {}^{i+1}P \\ 1 \end{bmatrix} \in \mathbb{R}^4 $$

Once the transformation matrix has been computed, the change of coordinates from $< i >$ to $< i+1 >$ is given by:

$$ {}^{i+1}\bar{P} = {}^{i+1}_{i}\, T \, {}^{i}\bar{P} $$

Due to these considerations we filled the *BuildTree()* function with the following transformation matrices:

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

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

2.2 Q1.2

This section focuses on determining how a matrix associated with a joint undergoes rotation when the joint itself rotates. Specifically, the goal is to calculate the transformation matrix between one link and the next one, when transitioning from an initial configuration. For this aim we implemented two functions:

- *DirectGeometry()*: that computes the transformation matrix ${}^{i}_{i+1}T$ for a joint, taking into account its type, which can be either rotational (0) or prismatic (1), as well as its current position q_i.
- *GetDirectGeometry()*: that uses *DirectGeometry()* to generate all the model's matrices representing transformations between joint positions for a given set of configurations.

In the given CAD model, all joints are specifically designed as rotational and no prismatic joints are present. For rotational joints, the function exclusively computes the rotation matrix around the z-axis by multiplying ${}^{i}_{i+1}R$ by the rotation matrix designed around it, reflecting the joint's ability to rotate specifically around this axis within the DH convention. This process accurately captures the rotational movement within the robotic system. On the other hand, for prismatic joints, the function adds a displacement equivalent to the product of the z-axis of the $(i+1)$ joint times the corresponding q value to the distance between the two adjacent joints ${}^{i}r_{i+1/i}$.

For this task, we are 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** $= [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$.

Let's consider for example the last one: **q** $= [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2/3\pi, 0]$. The resulting transformation matrices delineating the relationship between each link and its subsequent counterpart are the followings:

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

$$ {}^{3}_{4}T = \begin{bmatrix} 0 & 1 & 0 & 145.5 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 326.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{4}_{5}T = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 35 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ {}^{5}_{6}T = \begin{bmatrix} -0.5 & -0.866 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0.866 & -0.5 & 0 & 385 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{6}_{7}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 153 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

It is relevant to observe that, since the manipulator is composed of rotational joints only, the components related to the distances and the axes of rotation z- between two adjacent joints, remain unchanged.

2.3 Q1.3

In the context of a manipulator with multiple links it is possible to analyze the transformation matrices between any two different frames. Each link of the manipulator is associated with its own coordinate frame and the transformation matrix allows us to describe how one frame is related to another. In order to study the relation between two non-adjacent links, we need to consider all the intermediate transformations along the **Tree of Frames** connecting them.

To develop this part of the assignment we are tasked with the implementation of the following MATLAB functions: *GetFrameWrtFrame()*, *GetTransformationWrtBase()* and *GetBasicVectorWrtBase()*.

The first function is used to compute the transformation matrix from the i-th to the j-th joint in the configuration. To do so, it takes as input the vector containing all the transformation matrices obtained in the previous step and the numbers of the two links between which we want to compute this transformation. Then, we simply built a for loop to scan the vector, starting from the position containing the transformation matrix ${}^{i+1}_{i}T$ up to the one containing ${}^{j}_{j-1}T$ and multiply all the intermediates matrices one by another. This process results in the cumulative transformation matrix ${}^{j}_{i}T$.

Example, construction of the transformation matrix between frame $< 1 >$ and frame $< 4 >$:

$$ {}^{1}_{4}T = {}^{1}_{2}T * {}^{2}_{3}T * {}^{3}_{4}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 98 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & -1 & -105 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 145.5 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 326.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & -431.5 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 243.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

The development of *GetTransformationWrtBase()* is conducted by following the same steps. In fact, to compute the transformation matrix from the manipulator's base to the i-th joint in the configuration, it takes as input the vector containing all the transformation matrices and the number of the link up to which we want to compute it. Then we multiply all the intermediates transformation matrices going from ${}^{1}_{0}T$ up to ${}^{i-1}_{i}T$ one by another. Resulting in the cumulative transformation matrix ${}^{i}_{0}T$.

Finally, we want to compute the basic vector going from frame $< i >$ to the base $< 0 >$. For this step it is needed to pass the number of the frame $< i >$ we want to consider and the vector containing all the transformation matrices. Then, by calling the function *GetTransformationWrtBase()*, computed in the previous part we can obtain the transformation matrix ${}^{i}_{0}T$, from which we will extract its last column, corresponding to the translation vector. By doing so we can obtain the basic vector going from frame $< i >$ to the robot's base frame.

Example, construction of the transformation matrix between frame $< 0 >$ and frame $< 3 >$:

$$ {}^{0}_{3}T = {}^{0}_{1}T * {}^{1}_{2}T * {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 175 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 98 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 0 & -1 & -105 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1 & -105 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 273 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

From the matrix ${}^{3}_{0}T$ we can then extract the basic vector $r = \begin{bmatrix} -105 \\ 0 \\ 273 \end{bmatrix}$

2.4 Q1.4

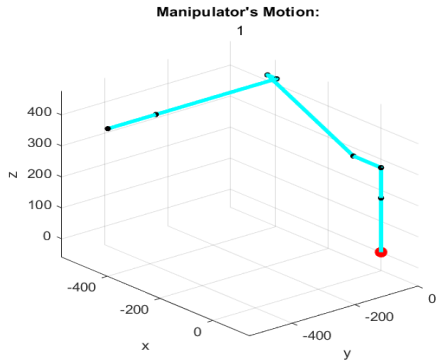
For the development of the fourth task we decided to implement a new function, called *PlotMotion()*, in order to observe the manipulator's motion from the initial configuration q_i to the final one q_f.

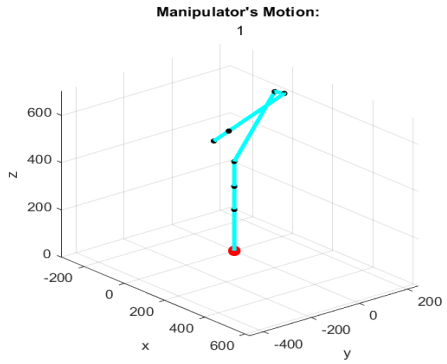
To generate the intermediate joint configurations *qsteps* between the initial and the final states we used the MATLAB function *linspace()*, which creates a linearly spaced vector of joint positions for each link. Then we can compute the vector containing all the transformation matrices using the previously seen *GetDirectGeometry()* function. At this point, we construct a matrix to store the basic vectors of each link with respect to the base, including the base itself which is stored in the first column.

Finally, thanks to the MATLAB functions *line()* and *plot3D()*, we are able to observe the complete 3D motion of the manipulator. Joints are represented as black dots and the links connecting them are displayed as cyan lines, while the base frame is represented as a red circle. The view is set at every iteration in order to better follow the plot's perspective and frames are animated using *getframe*. Clearing after each iteration with *cla()* ensures a dynamic and precise visualization of the robot's motion.

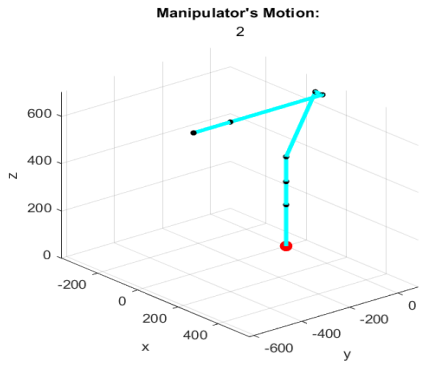
Here we report the starting and ending position of the manipulator given the different starting and ending configurations:

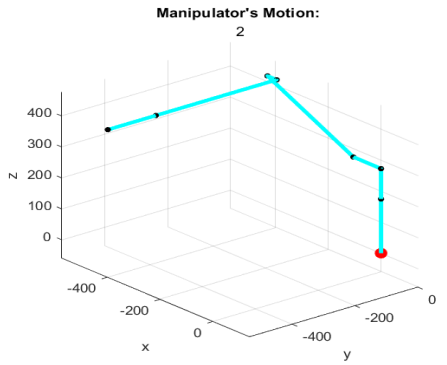
- $\mathbf{q}_i = [0, 0, 0, 0, 0, 0, 0]$ and $\mathbf{q}_f = [\pi/4, \pi/2, -\pi/8, -\pi/2, \pi/4, 2\pi/3, 0]$



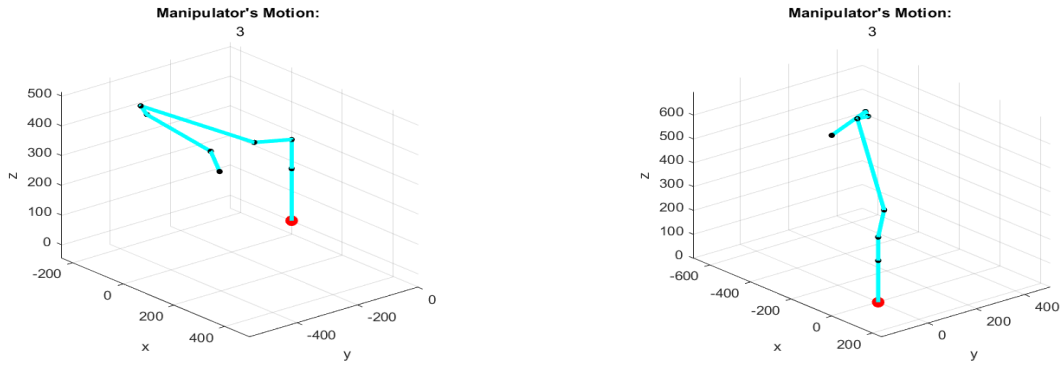


- $\mathbf{q}_i = [0, \pi/2, 0, -\pi/2, 0, 0, 0]$ and $\mathbf{q}_f = [0, 0, 0, 0, 0, 0, 0]$





- $\mathbf{q}_i = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and $\mathbf{q}_f = [2, 2, 2, 2, 2, 2, 2]$



2.5 Q1.5

For the final part of the assignment we wan to highlight the behaviour of every single joint, to do so we decided to set a singular initial configuration $\mathbf{q}_i = [0, 0, 0, 0, 0, 0, 0]$ and we changed every joint one at a time.

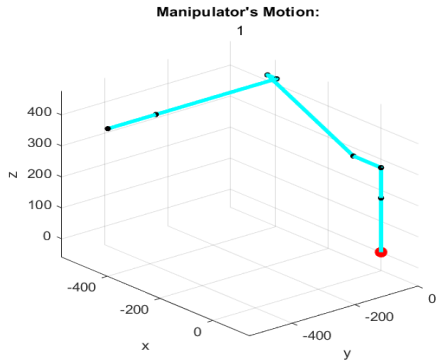


Figure 3: Manipulator's Initial Configuration $\mathbf{q}_i = [0, 0, 0, 0, 0, 0, 0]$.

Starting from this initial configuration, we analyzed the behavior of each joint by adjusting one by one their current positions by a factor of $\frac{\pi}{2}$. In order, you can see the following configurations being:

- Manipulator's Motion 1: $\mathbf{q}_f = [\frac{\pi}{2}, 0, 0, 0, 0, 0, 0]$
- Manipulator's Motion 2: $\mathbf{q}_f = [0, \frac{\pi}{2}, 0, 0, 0, 0, 0]$
- Manipulator's Motion 3: $\mathbf{q}_f = [0, 0, \frac{\pi}{2}, 0, 0, 0, 0]$
- Manipulator's Motion 4: $\mathbf{q}_f = [0, 0, 0, \frac{\pi}{2}, 0, 0, 0]$
- Manipulator's Motion 5: $\mathbf{q}_f = [0, 0, 0, 0, \frac{\pi}{2}, 0, 0]$
- Manipulator's Motion 6: $\mathbf{q}_f = [0, 0, 0, 0, 0, \frac{\pi}{2}, 0]$

These are the results that we obtained:

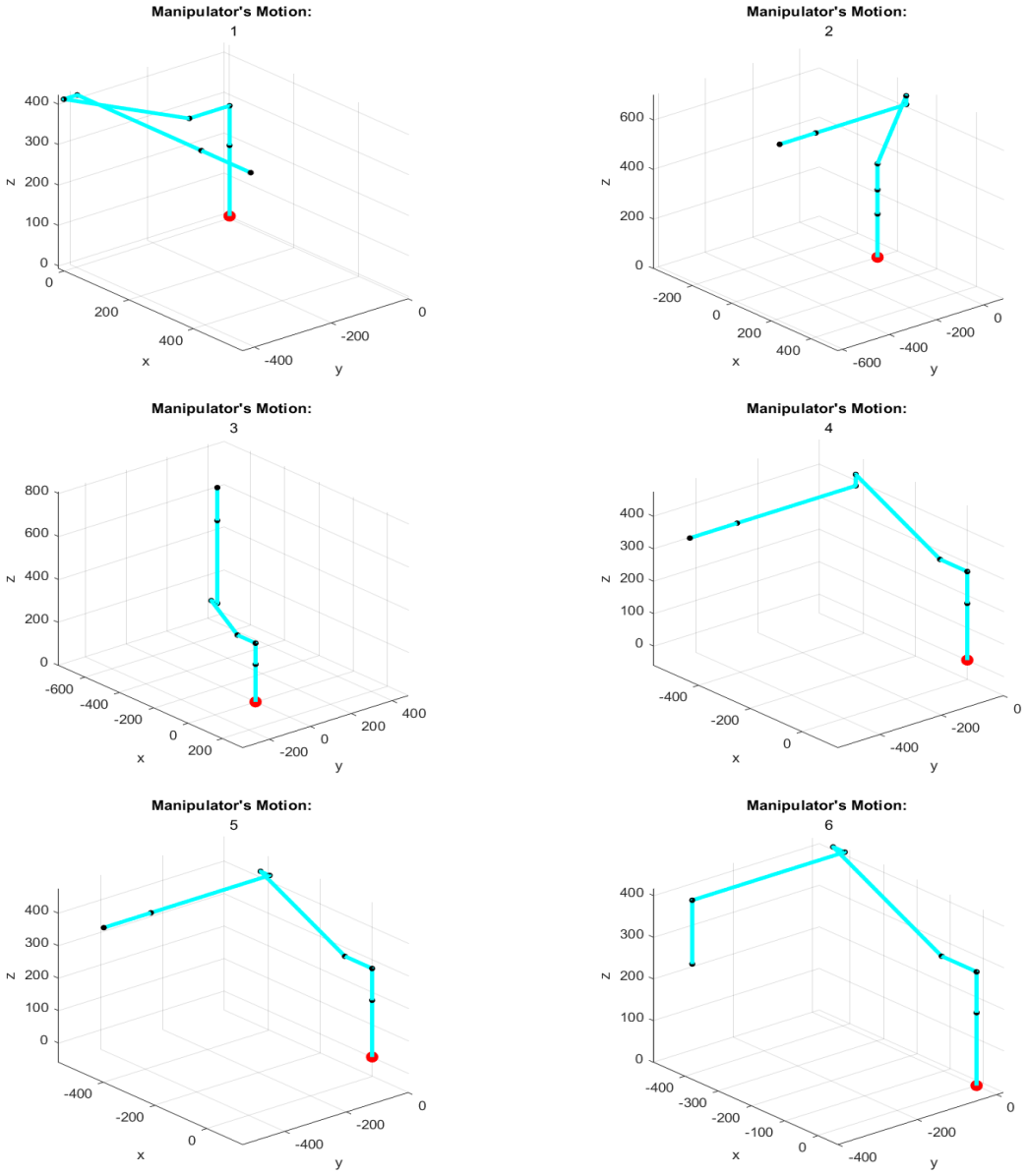


Figure 4: ”Manipulator’s Final Configuration.”

Please note that the final joint is always set at 0, as its rotation is not visibly observable.