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:

Ines Haouala

Student ID:

s5483776

Professors:

Giovanni Indiveri
Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo

December 30, 2022

Contents

Mathematical expression	Definition	MATLAB expression
$< T >$	Model Transformation Matrix	T
Rz	Rotation matrix around z axis	Rz
q	Robot joints configurations	q
iTj	Transformation matrix from link i to link j	iTj
bTi	Transformation matrix With respect to base	bTi
bri	Distance vectors between tranformation matrices	bri

Table 1: Nomenclature Table

1 Assignment description

Kinematics is a fundamental concept in Robotics which is useful in studying the relationship between the robot's joint coordinates and its spatial layout. Kinematics can be used to determine the positioning of an end-effector of an industrial robot and the general motion of the robot such that the robot can successfully navigate in its workspace avoiding obstacles. It is important to note, the study of Kinematics is not concerned with forces and torques but only with the instantaneous values of the robot's coordinates [1]. The second assignment of Modelling and Control of Manipulators focuses on manipulators geometry and direct kinematics. A CAD model of an industrial 7 dof manipulator is provided, refer to Appendix A. This robot model will be used throughout this report. Transformation matrices relative to the robot's joints and different sets of configurations are computed and analyzed. In the last section of this report, geometric calculations will be performed from a set of initial and final configuration and will then be used to map the configuration space to workspace.

2 Exercise 1

Q1.1

A CAD model of an industrial 7 dof manipulator is provided, refer to Appendix A. Reference frame for each link is defined and drawn as shown in figure 1. It is important to note that for all reference frames, the z-axis coincides with the joint rotation axis, and such that the positive rotation is the same as showed in the CAD model. From the CAD model, we also deduct the lengths of each link of the industrial robot. From all the extracted information above, we can build the following function on MATLAB: *BuildTree()*. This function builds a tree of frames for the given manipulator. As an example of the output of the *BuildTree()* function is:

- Relative to the first joint, the obtained model matrix is: $T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 175 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
- Relative to the last joint, the obtained model matrix is: $T = \begin{bmatrix} 0 & 0 & -1 & -153 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

From the model of the first joint, we observe that the rotational matrix (the upper-left 3×3 matrix) Rz is the idendity matrix and this is because there is no rotation between the base frame and joint 1. However, there is a translation of 175 at the z axis. This is well demonstrated in the model matrix and figure 1. Whereas, from the model of the last joint as an example, we observe that the rotational matrix Rz is different from the idendity matrix and this is because there is a rotation around the z axis. Adding to that, there is a translation of 153 along the x axis, relative to the last link length. This is well demonstrated in the above model matrix and figure 1.

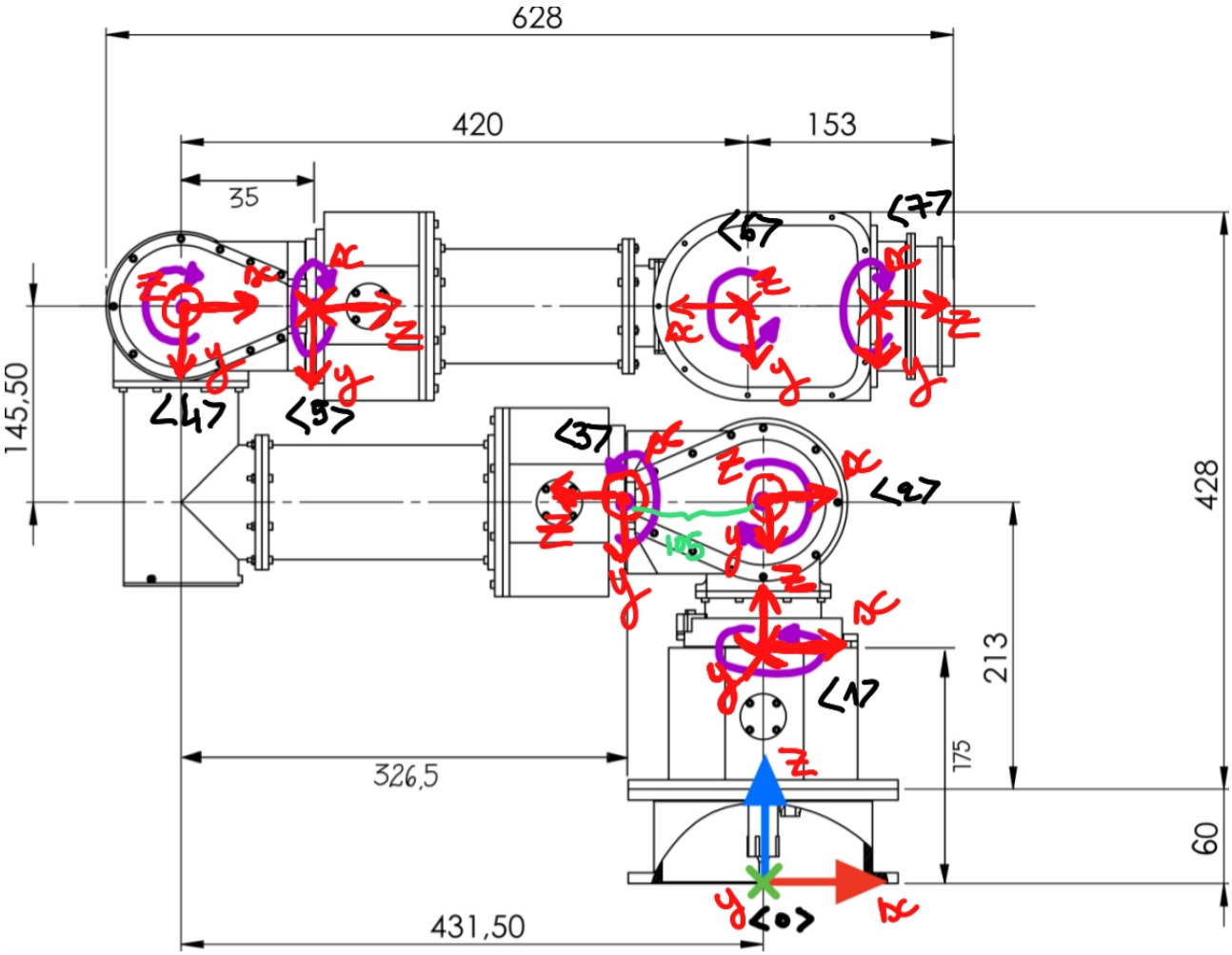


Figure 1: CAD Model with reference frames

Q1.2

In this section, we are looking to compute the rotational matrix when a specific joint rotates and given a specific configuration. This is implemented on MATLAB using the function *DirectGeometry()*. The following lines of code are from *DirectGeometry()* function that calculates how the matrix attached to a joint will rotate if the joint rotates:

```
function biTei = DirectGeometry(qi, linkType)
  biTei=zeros(4,4);
  if linkType == 0
    biTei = [cos(qi) , -sin(qi) , 0 , 0 ; sin(qi) , cos(qi)
    , 0 , 0 ; 0 , 0 , 1 , 0 ; 0 , 0 , 0, 1];
```

It is to be noted that the output of the above is the result of the rotation of joints only. Here, the translations between the joints frames are not put into consideration. Also, in the given model all the joints are rotational, there is no prismatic joints. The size of the array output is: 4*4*1 Then, based on the given CAD model and using *DirectGeometry()* function, we can retrieve all the model matrices knowing the set of joint configuration q = [1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3]. The following lines of code are from *GetDirectGeometry()* function:

```
function [iTj_q] = GetDirectGeometry(q, linkType, numberOfLinks)
  for i = 1:numberOfLinks
    iTj_q(:,:,i) = DirectGeometry(q(i), linkType);
  end
  %adding the lenghts of the links, 1st link/joint, list goes on until 7th link
  iTj_q(3,4,1) = 175 ;
```

It is to be noted that the output of the above is the result of the rotation of all of the joints as well as of the translation of all of the frames. The translation of joints frames is derived from the lengths of the links.The size of the array output is: 4*4*7 as the number of joints/links is 7. The following is an example of the last joint output iTj_q(1,1,7)= -153, which also verifies the mentioned above:

$$iTj_q(:,:,7) = \begin{bmatrix} 0.2675 & -0.9636 & 0 & -153.0000 \\ 0.9636 & 0.2675 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The full code is also available in Appendix B.

Q1.3

This section aims to extract all the transformation matrices between any two links, between a link and the base and the corresponding distance vectors. For these purposes, three function are implemented on MATLAB: *GetFrameWrtFrame()*, *GetTransformationWrtBase()*, and *GetBasicVectgorWrtBase()*, refer to Appendix B. The *GetTransformationWrtBase()* outputs an array bTi of size: 4*4*7. This array collects the set of transformation matrices between the 7 links of the industrial robot. For examples:

$$bTi(:,:,1) = \begin{bmatrix} 0.2675 & -0.9636 & 0 & 0 \\ 0.9636 & 0.2675 & 0 & 0 \\ 0 & 0 & 1.0000 & 175 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$bTi(:,:,1)$ is the first transformation matrix and is the same as iTj_q(:,:,1) from *GetDirectGeometry()* function.

$$bTi(:,:,7) = \begin{bmatrix} -0.9477 & -0.3191 & 0 & -332.2553 \\ 0.3191 & -0.9477 & 0 & 216.4284 \\ 0 & 0 & 1.0000 & 984.5000 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

bTi(:,:,7) is the transformation matrix between the base link and the next 6 links. In other words, bTi(:,:,7)=$T1(q1)$*$T2(q2)$*$T3(q3)$*$T4(q4)$*$T5(q5)$*$T6(q6)$

The *GetFrameWrtFrame()* computes the transformation matrices between two chosen frames/links. As shown in the code from Appendix B, the starting link or link-Number-i is set as an example at 3 and the last link or link-Number-j is set as an example at 6. Therefore, *GetFrameWrtFrame()* outputs the array iTj that representes the transformation matrices between link 3 and link 6. The size of iTj is: 4*4*(link-Number-j - link-Number-i)= 4*4*4. The following is the resultant arrays $iTj(:,:,1)$ and $iTj(:,:,4)$ as examples obtained from *GetFrameWrtFrame()*.

$$iTj(:,:,1) = \begin{bmatrix} -0.7259 & 0.6878 & 0 & -105.0000 \\ -0.6878 & -0.7259 & 0 & 0 \\ 0 & 0 & 1 & 273 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$iTj(:,:,4) = \begin{bmatrix} 0.0540 & -0.9985 & 0 & -188.6719 \\ 0.9985 & 0.0540 & 0 & 74.7022 \\ 0 & 0 & 1 & 984.5 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The *GetBasicVectorWrtBase()* outputs the distance vectors, as shown below. This function will be used in the next section to plot the intermediate link positions in between initial and final configurations.

$$bri = \begin{bmatrix} 0 & 0 & -105. & -28.0874 & 265.1710 & 160.5908 & -332.2553 \\ 0 & 0 & 0 & -246.6736 & -93.0487 & 230.6173 & 216.4284 \\ 175 & 273 & 273 & 599.5 & 599.5 & 984.5 & 984.5 \end{bmatrix}$$

Q1.4

In this part, the derived forward kinematic and geometric calculations from the previous sections will be used to map the intermediate link positions in between two sets of initial/start and final configurations. To achieve this, *GetDirectGeometry()* and *GetBasicVectorWrtBase()* are implemented on MATLAB to respectively compute the transformation matrices between the links and the corresponding distance vectors. The outputs of *GetBasicVectorWrtBase()* is mapped and plotted using the MALTLAB function *plot3()*. This is well demonstrated in figures 2,3,4 and 5, where the red line corresponds to the mapping of the initial configuration and the green line corresponds to the mapping of the finish configuration. There are three case with different sets of start and finish configurations:

- Start configuration, $qi = [1.3, 1.3, 1.3, 1.3, 1.3, 1.3, 1.3]$ and finish configuration $qf = [2, 2, 2, 2, 2, 2, 2]$, the obtained figure is:

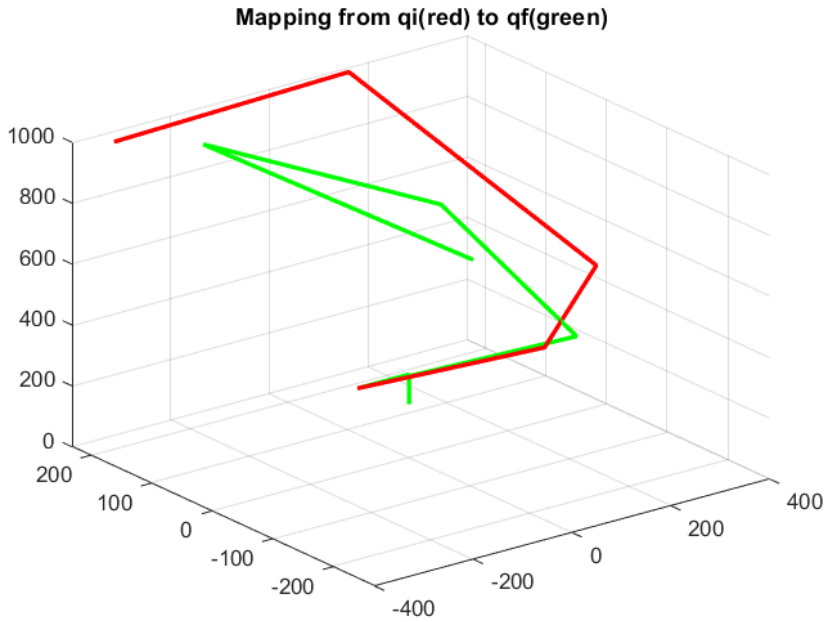


Figure 2: Case1, intial/final configs mapping

- Start configuration, $qi = [1.3, 0, 1.3, 1.7, 1.3, 0.8, 1.3]$ and finish configuration $qf = [2, 0, 1.3, 1.7, 1.3, 0.8, 1.3]$, the obtained figures are:

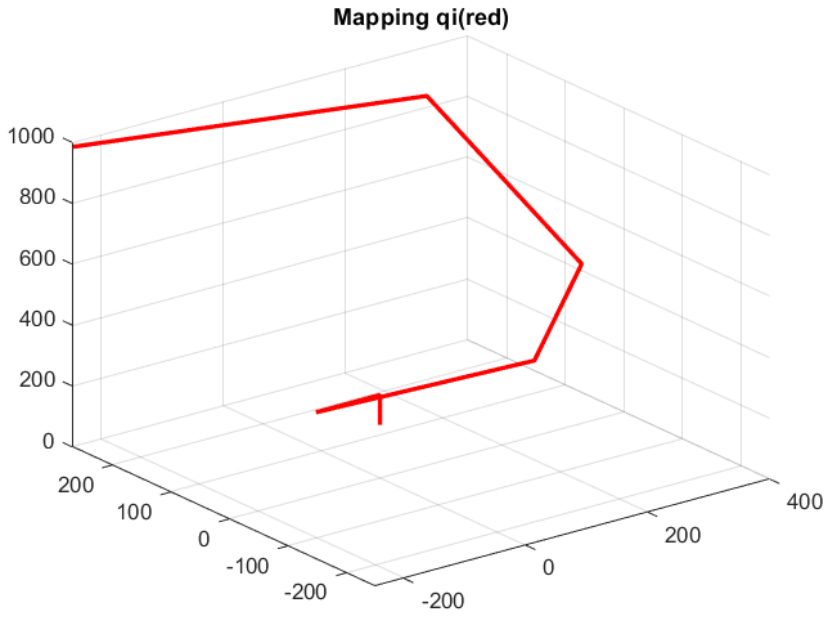


Figure 3: Case2, intial configs mapping

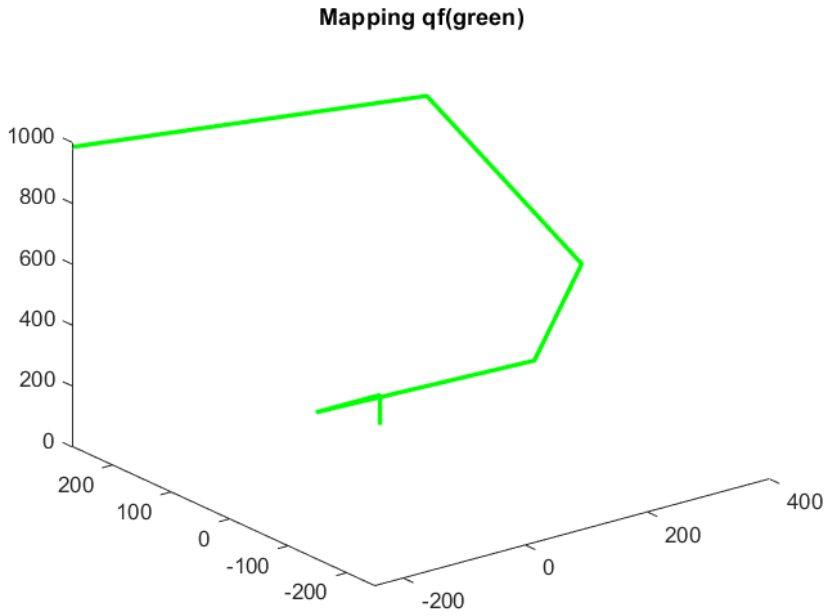


Figure 4: Case2, final configs mapping

- Start configuration, $qi = [1.3, 0.1, 0.1, 1, 0.2, 0.3, 1.3]$ and finish configuration $qf = [2, 2, 2, 2, 2, 2, 2]$, the obtained figure is:

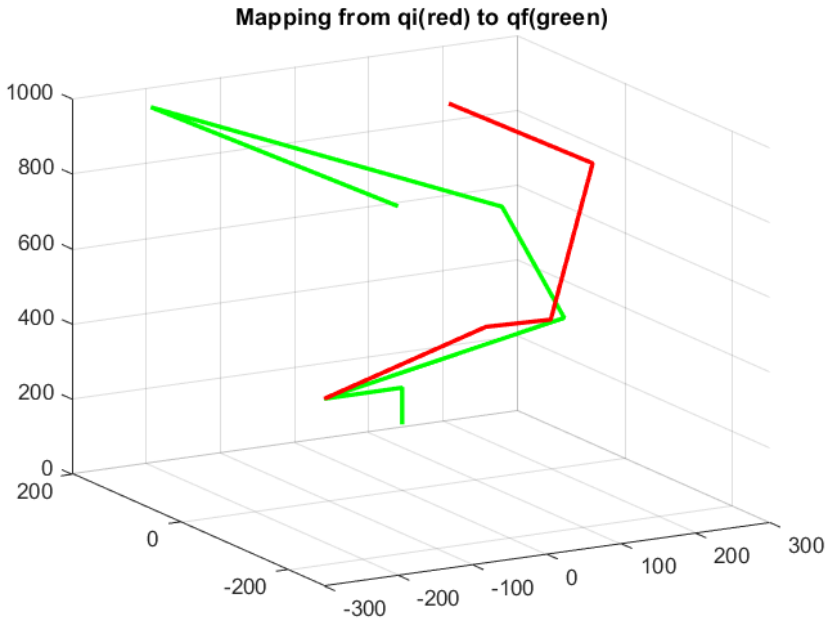


Figure 5: Case3, intial/final configs mapping

Q1.5

This sections is looking into the effect of a single change in the joint configuration on the overall joints positions. To do this, one joint position is changed at a time. Meaning, there will be four different sets of joints configuration, where:

- initial joint configuration: $qstart = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]$
- first change, last joint set to 0.9: $q1 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.9]$
- second change, fourth joint set to 0.5: $q2 = [0.1, 0.1, 0.1, 0.5, 0.1, 0.1, 0.1]$
- third change, sixth joint set to 0.7: $q3 = [0.1, 0.1, 0.1, 0.1, 0.1, 0.7, 0.1]$

For a better visualization, all of the 4 configurations are plotted on the MATLAB graph and where the red, blue,yellow and green lines correspond respectively to: $qstart$, $q1$, $q2$ and $q3$. This is demonstrated in figure 6, and in the section 1.5 in the MATLAB code from Appendix B.

From figure 6, the red line corresponds to the initial configuration of joints, qstart. As shown in the same figure, as we move to the next configuration by changing one single joint at a time, the blue, yellow and green lines start diverging, either partially or completely, from the red line. Therefore, even by changing one joint at a time, the intermediate joints positions can be changes. The yellow line in figure 6 from the third change, q3, is a good example of that.

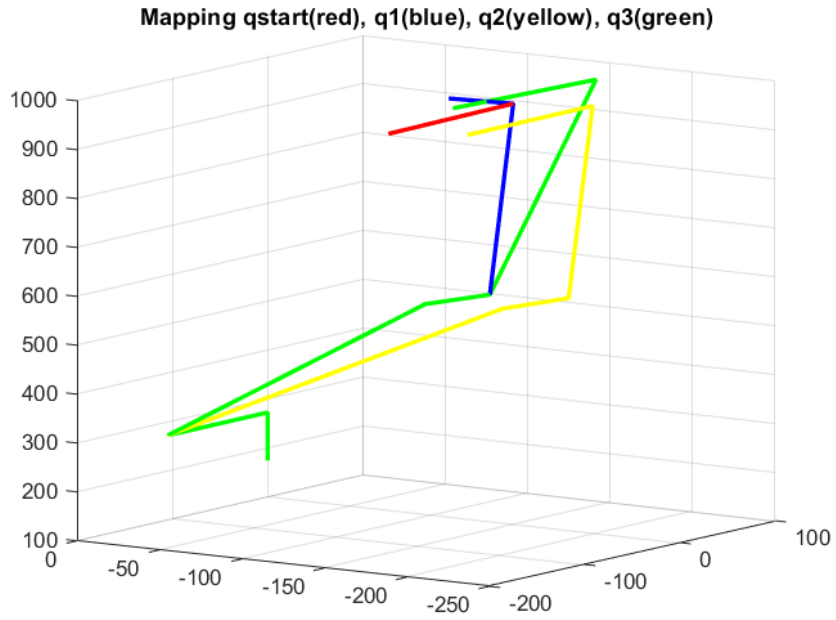


Figure 6: Four configurations mapping

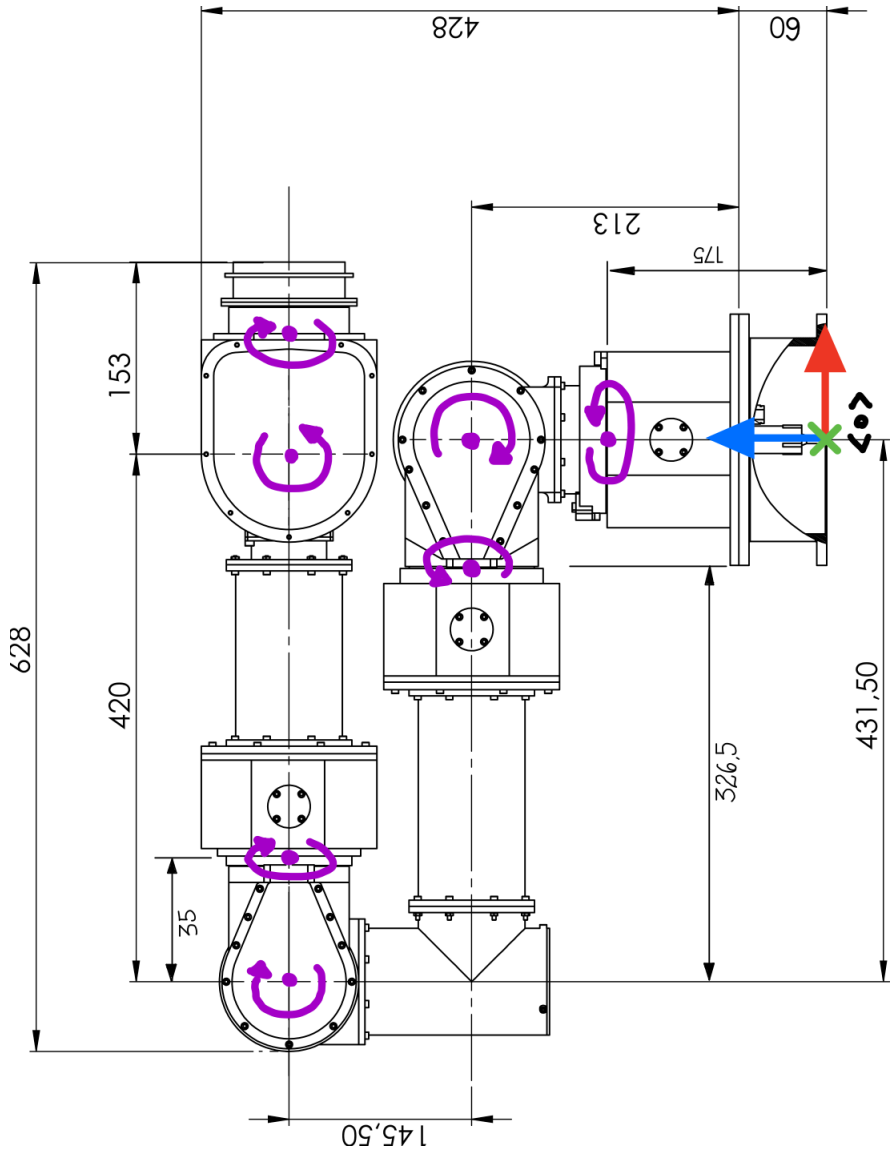
.

3 References

[1] “Kinematics,” motion.cs.illinois.edu. http://motion.cs.illinois.edu/RoboticSystems/Kinematics.html

4 Appendix

4.1 Appendix A



4.2 Appendix B

The following is the link to the second MCM assignment folder/repository that is uploaded on Github. Inside the folder, there is the main.m that contains the MATLAB script to execute the tasks in exercice 1. In the same directory, there are files that contain useful functions used in the main.m file. The MCM folder is also available on Aulaweb as a zip file.
Link:https://github.com/Fritta013/Modeling-and-Control-Assignment-2.git