MECA-H410
ROBOTICS

Project report

Raphaël BOITTE
Maxime BUSSIOS
Inès CASTILLO FERNANDEZ
Tigran PLETSER
Samuel VAGMAN
Julien VAN DELFT

Professor:
Michaël VAN DAMME
Assistant:
Ilias EL MAKRINI

Academic year 2020 - 2021

Contents

1 Introduction

In the context of the *MECA-H410 Robotics* course, a Franka Emika robot has been programmed in order to pour a glass. It is a robot of seven degrees of freedom, but since the goal is only to pick a glass and pour the liquid in it somewhere, in fact only four degrees of freedom will be needed. This will be explained in this report, as the programming and simulation parts.

The first step will be to determine the inverse kinematics of the robot. In order to do so, the frames and the Denavit-Hartenberg parameters will be determined. Then, the robot will be programmed in order to be able to execute the function one gave to it on the ROS (Robot Operating Software).

2 Definition of the frames

A first step is to define the frames in which one is working. In this case, the frames were defined following the Denavit-Hartenberg convention. On the figure below the seven joints are shown :

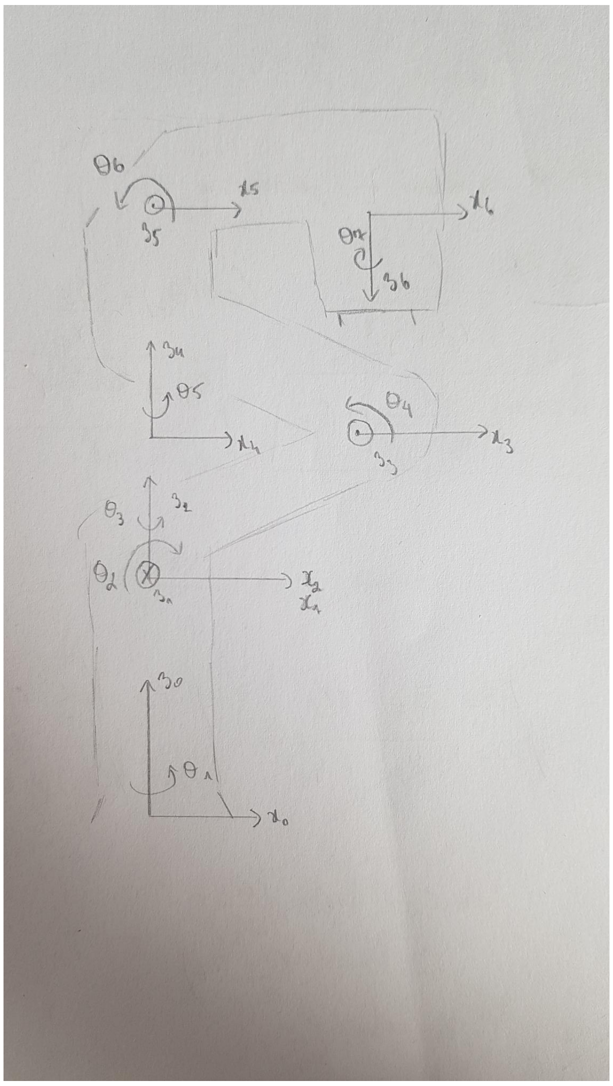

Figure 1: Franka robot

After assigning frames to the robot, the Denavit-Hartenberg parameters can be deduced. Essentially, every frame transformation can be represented by four transformation parameters : a_i, d_i, α_i and θ_i. Two of them represent lengths, and the other two represent angles. They are defined by :

- a_i : distance DO_i, measured along x_i, it represents the length of a link i
- d_i : distance $0_{i-1}D$, measured along z_i
- α_i : twist angle between z_{i-1} and z_i around x_i

- θ_i : angle between x_{i-1} and x_i around z_i

They are given in the table below.

	a_i	α_i	d_i	θ_i
1	0	$-\frac{\pi}{2}$	$0,333$	θ_1
2	0	$\frac{\pi}{2}$	0	θ_2
3	$0,0825$	$\frac{\pi}{2}$	$0,316$	θ_3
4	$-0,0825$	$-\frac{\pi}{2}$	0	θ_4
5	0	$\frac{\pi}{2}$	$0,384$	θ_5
6	$0,088$	$\frac{\pi}{2}$	0	θ_6
7	0	0	$0,107$	θ_7

3 Inverse kinematics

3.1 Kinematic decoupling

The first idea was to use kinematic decoupling to simplify the computation of inverse kinematics by splitting the big problem into two smaller ones. This method allows to keep 6 degrees of freedom of the robot. To be able to apply kinematic decoupling, the robot must contain a spherical wrist meaning a point where three joint axes are intersecting whatever the angles. Although the Franka Emika robot does not have a spherical wrist with the upper joints, one can remark that a spherical shoulder is present at the base of the robot. For computation purposes, the robot would be flipped over, the base being the new end-effector. This way, one can consider that the robot possesses a spherical wrist. Now, the 6 degrees of freedom problem can be decoupled into two systems with 3 degrees of freedom, one determining the position of the new end-effector and the other its orientation.

First, the angles of the three joints close to the new base (close to the gripper) are computed depending on the wanted position of the new end-effector using a geometrical approach. After that, the orientation can be deduced using the wanted orientation of the end-effector and the orientation matrix of the three first joints computed before.

This method is powerful but it introduces a complication in our case. Indeed, as the robot is flipped, the desired position of the end effector that needs to be put as input is the desired position of the base in the gripper frame. This means that to have the real desired position of the gripper, this position needs to be transformed in the base one. This method being way more difficult than necessary, it has been rapidly abandoned.

3.2 Geometrical approach

The approach now consists in, with a given desired position and orientation of the end-effector p_i, to find the joint variables θ_i to get p_i such that : $\theta_i = f(p_i)$. There are seven joint variables, as shown above, but since the goal is to take a glass at a certain position, and then pour the liquid in it, only four degrees of freedom are needed, so only four joint variables. It was decided to block the variables θ_3, θ_5 and θ_6. So the four remaining degrees of freedom are : (θ_1, θ_2, θ_4 and θ_7), as shown on the figure below :

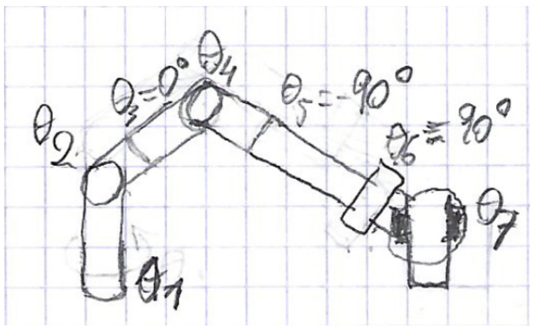

Figure 2: Degrees of freedom

The next step is to calculate these variables. On the following figure 3 one can see the robot seen from above, this way one can see that the angle θ_1 can be calculated as follow, and independently from the other variables. Knowing that θ_g is the angle between the

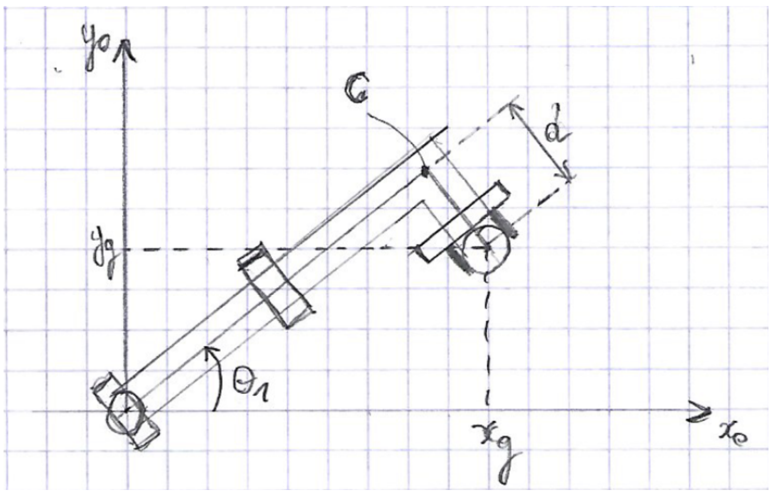

Figure 3: Upper view - determination of θ_1

origin and the extremity of the robot's position (of joint 7), one has :

$$\theta_g = atan2(x_g, y_g) \tag{1}$$

The position of point C is (x_c, y_c), so from the figure, we get that

$$\begin{cases} x_c = x_g - d sin\theta_1 \\ y_c = y_g + d cos\theta_1 \end{cases} \tag{2}$$

Finally, one can find θ_1, the angle of the rotation of the robot w.r.t. the vertical axis x_0 :

$$\theta_1 = atan2(x_g, y_g) + asin(\frac{d}{\sqrt{x_g^2 + y_g^2}}) \tag{3}$$

For finding the angles θ_2 and θ_4, the following scheme shows the situation, seen from the side.

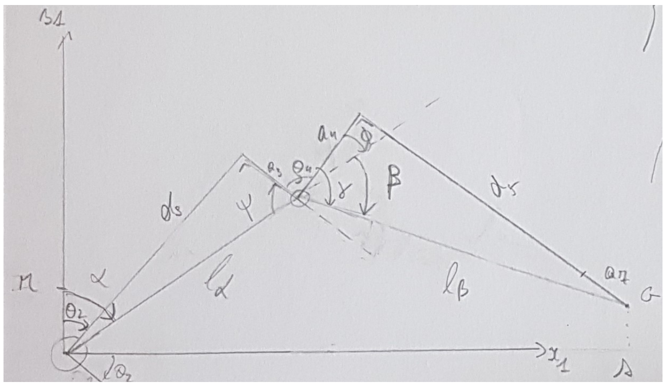

Figure 4: Side view - determination of θ_2 and θ_4

One can observe that this situation is the same type of situation as a two link manipulator, where once again, the position of joint 7 is given by (x_g, y_g). First of all, the distances r and s are given by :

$$\begin{cases} r = & z_g - d_1 \\ s = & \sqrt{x_c^2 + y_c^2} \end{cases} \tag{4}$$

The lengths of both arms of the robot are given by :

$$\begin{cases} l_\alpha = & \sqrt{d_3^2 + a_3^2} \\ l_\beta = & \sqrt{(d_5 + a_6)^2 + a_3^2} \end{cases} \tag{5}$$

To calculate the angles defined by the arms of the robot and the vertical axis, we need to take in account the offset of joint 4. This is done by first introducing the angles γ and ψ :

$$\begin{cases} \gamma = & atan(\frac{d_5+a_6}{a_3}) \\ \psi = & atan(\frac{d_3}{a_3}) \end{cases} \tag{6}$$

Then, the angles α and β are given by:

$$\begin{cases} \alpha = & \theta_2 + atan(\frac{a_3}{d_3}) \\ \beta = & -\theta_4 + \psi + \gamma - \pi \end{cases} \tag{7}$$

These angles can be considered as offsets of the angles θ_2 and θ_4. Finally, one gets :

$$\begin{cases} cos\beta = & \frac{r^2+s^2-l_\alpha^2-l_\beta^2}{2l_\alpha l_\beta} := D \\ \beta = & atan2(D; \pm\sqrt{1-D^2}) \\ \theta_4 = & -\beta + \psi + \gamma - \pi \\ \alpha = & atan2(r; s) - atan2(l_\alpha + {}_\beta cos\beta; l_\beta sin\beta) \\ \theta_2 = & \alpha - atan(\frac{a_3}{d_3}) \end{cases} \tag{8}$$

Finally, to find θ_7, the following drawings can help visualize the situation of the gripper. Indeed, the angle of joint 7 is set so it doesn't move (it stays horizontally) until it reaches the glass. It only changes when the robot starts pouring the liquid of the glass. To determine θ_7, the following steps are applied. First of all, the angle δ is defined such that

$$\delta = \frac{\pi}{2} - (-\theta_7) \tag{9}$$

where θ_7 is defined negative. One also knows that

$$\pi = (\frac{\pi}{2} - \theta_2) + \pi - (-\theta_4) + \delta \tag{10}$$

where θ_4 is also defined negative. This leads to the fact that

$$0 = \frac{\pi}{2} - \theta_2 + \theta_4 + \delta \tag{11}$$

So

$$\delta = \theta_2 - \theta_4 - \frac{\pi}{2} \tag{12}$$

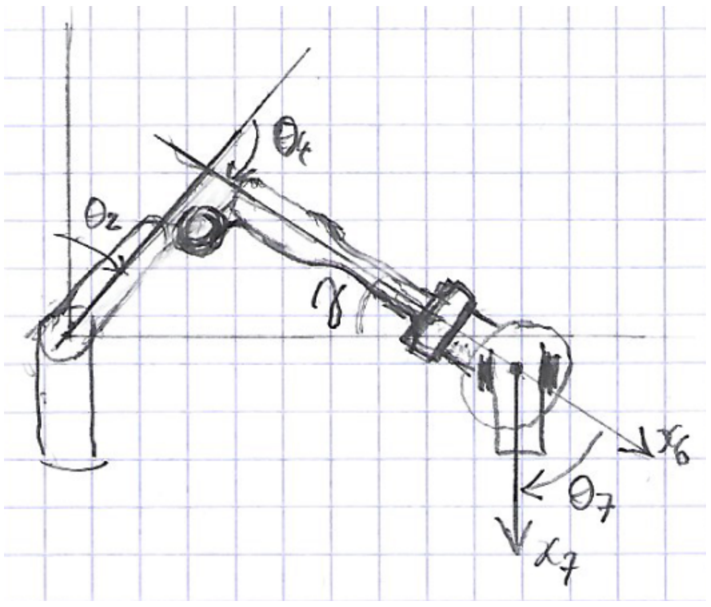

Figure 5: Determination of θ_7

which leads to

$$\frac{\pi}{2} + \theta_7 = \theta_2 - \theta_4 - \pi \tag{13}$$

One finally obtains for the angle of the gripper :

$$\theta_7 = \theta_2 - \theta_4 - \pi \tag{14}$$

One can also take the desired orientation of the gripper in account by introducing a pouring angle θ_p and including it in the equation of θ_7:

$$\theta_7 = \theta_p + \theta_2 - \theta_4 - \pi \tag{15}$$

4 Validation of the results

To validate the results of the joint variables, the principle is the following. First, the direct kinematics equations are computed using the symbolic toolbox of *Matlab*.

Let us denote:

$$A_i = T_i^{i-1} \tag{16}$$

where T_i^{i-1} is the homogeneous transformation from frame i to frame i-1. A_i can be calculated from the Denavit-Hartenberg parameters :

$$A_i = \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

The direct kinematics are then calculated as:

$$A = A_1A_2A_3A_4A_5A_6A_7 \tag{18}$$

The matrices are computed using the Denavit-Hartenberg parameters given previously (see section 2) and the predefined values of $\theta_3 = 0$, $\theta_5 = \frac{-\pi}{2}$ and $\theta_6 = \frac{\pi}{2}$.

$$A_1 = \begin{pmatrix} cos(\theta_1) & 0 & -sin(\theta_1) & 0 \\ sin(\theta_1) & 0 & cos(\theta_1) & 0 \\ 0 & -1 & 0 & 0.333 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A_2 = \begin{pmatrix} cos(\theta_2) & 0 & sin(\theta_2) & 0 \\ sin(\theta_2) & 0 & -cos(\theta_2) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0.0825 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0.316 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A_4 = \begin{pmatrix} cos(\theta_4) & 0 & -sin(\theta_4) & -0.0825 * cos(\theta_4) \\ sin(\theta_4) & 0 & cos(\theta_4) & -0.0825 * sin(\theta_4) \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0.384 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad A_6 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0.088 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_7 = \begin{pmatrix} cos(\theta_7) & -sin(\theta_7) & 0 & 0 \\ sin(\theta_7) & cos(\theta_7) & 0 & 0 \\ 0 & 0 & 1 & 0.107 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Now that the direct kinematics equations are found, the inverse kinematics equations implemented in *Matlab* are used to calculate the joint angles $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)$, starting from the position and orientation of the end-effector. To keep things simple, the orientation is first set to 0 (vertical orientation) such that one can mainly focus on achieving the right position (x, y, z).

After calculating the joint angles, it can now be checked that the direct kinematics give the same position that was used to calculate the inverse kinematics The position can be found the 3 first elements of the last column of the matrix A.

To illustrate this procedure, an arbitrary position in the achievable domain of the Franka robot is taken :

$$(x, y, z) = (0.2, 0, 0.9) \tag{19}$$

For this position, one obtains the following homogeneous transformation :

$$A = \begin{bmatrix} 0 & 0.8449 & 0.535 & 0.2 \\ 0 & 0.535 & -0.8449 & 0 \\ -1 & 0 & 0 & 0.9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

It can be observed on this transformation that the position is indeed achieved, which indicates that the inverse kinematics are likely to be correct. Also, the achieved rotation matrix has its x-axis in the vertical direction, which is consistent with the desired vertical orientation of the end-effector. Let us now take another position in the middle of the first quadrant:

$$(x, y, z) = (0.5, 0, 5, 0.5) \tag{21}$$

One obtains the following transformation:

$$A = \begin{bmatrix} 0 & 0.592 & 0.806 & 0.5 \\ 0 & 0.806 & -0.592 & 0.5 \\ -1 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

Again, the desired position is achieved, and the orientation is vertical.

To further validate the inverse kinematics, this procedure was applied to multiple points, and there was a perfect matching between the desired position and the achieved one.

It is important to note that, in practice, the validation was an iterative procedure. It has been used to verify the inverse kinematics, which didn't work at the beginning. This was due to multiple errors, and some equations (l_β and θ_4) had to be corrected before finding satisfying results. One also had to take in account the fact that the function $atan2$ is defined differently in *Matlab* than in the equations found in the reference book: in *Matlab*, $atan2(y, x)$ has to be used while $atan2(x, y)$ was used in the book.

5 Task design

The goal of the simulation is to grab a glass, carrying it to another position without spilling the fictitious liquid inside it, pouring the liquid and finally replacing the glass at its initial position.

The workspace considered is a part of the one shown in figure 6. For the 4 degrees of freedom case considered here, the left half of both views is not achievable. In addition, peculiar positions of the robot inside that workspace may not be achievable but as they have not been encountered when designing the trajectory, they are not discussed.

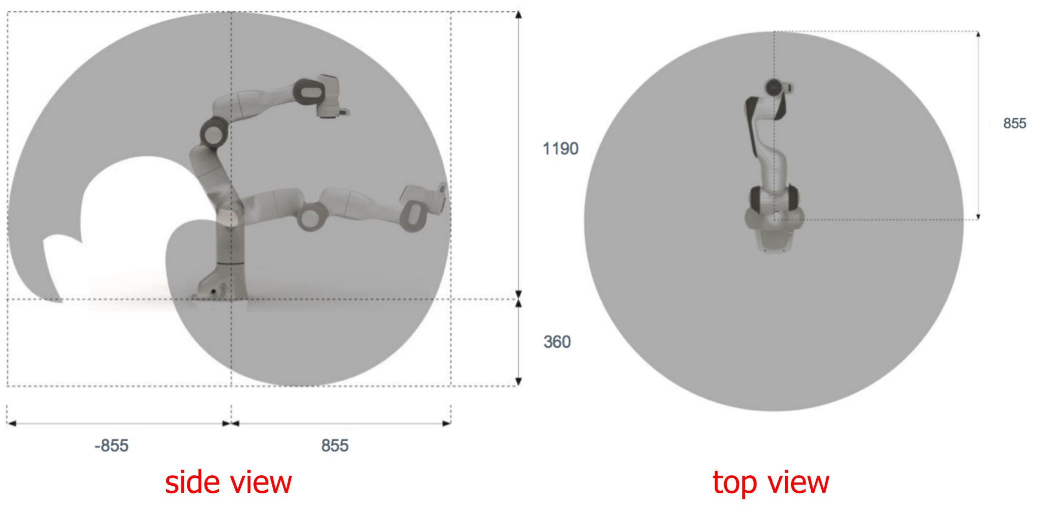

Figure 6: Workspace representation for the 7dof Panda robot (Robotics 1, university of Roma, 2011 exam)

To perform the task, approaching and exiting strategies must be designed in order to not collide with the glass and to grab it in a proper manner.

Through the whole motion of the gripper, it is kept at constant orientation in order to not spill any liquid. θ_7 is decreased by $\frac{\pi}{2}$ when pouring and is reset right after.

- *Pre-grasp approach :* The gripper is open at the start. The robot moves from its current position ($x_{current}$, $y_{current}$, $z_{current}$) vertically until reaching ($x_{current}$, $y_{current}$, $z_{current}+offset$). It then goes a bit above the glass (x_{glass}, y_{glass}, $z_{glass}+offset$). That pair of points ensure to never collide with the glass. Afterward it goes down vertically until reaching half of the glass height. The gripper is then closed and the pre-grasp approach is done.
- *Post-grasp approach :* The robot goes back to (x_{glass}, y_{glass}, $z_{glass}+offset$), goes down vertically until (x_{glass}, y_{glass}, z_{glass}), open its gripper and then goes back to (x_{glass}, y_{glass}, $z_{glass}+offset$). Eventually it goes back to its initial position.

Finally the trajectory goes as follow :

- pre-grasp approach

- move to a position a bit above the bowl
- The gripper does a rotation of $\frac{\pi}{2}$ to pour the liquid
- The rotation is reverted
- post-grasp approach
- flex under the tables to assert dominance

6 ROS

In order to perform the simulation of the pouring task by the franka robot, the online ROS (Robot Operating System) simulator *TheConstruct* has been used. A pre-made project has been provided in which the kinematic and task of the robot was yet to implement. A cylinder representing the glass, a bowl, to fictitiously receive the liquid poured, and two tables have been added to the simulation in order to modelize an actual scene of the world.

The inverse kinematics equations found in section 3.2 are implemented as a function (move_robot(x,y,z)) which take as inputs the desired position, in cartesian coordinates, of the end-effector and provide as outputs the joint angles of the robot. The orientation of the end-effector being constant while moving the glass (keeping it vertical), orientation angles are not needed as inputs. The pouring angle introduced in equation 15 is handled in the main() function.

In addition, an offset of $\frac{\pi}{4}$ is added to θ_7 in order for the zero of equation 14 to correspond with the actual zero of the joint angle.

The predefined angles of the gripper for the open state have been changed to their maximal values in order to be able to effectively grab the cylinder provided to modelize the glass.

The gripping of objects cannot be actually performed in the project provided, due to the absence of force control in the program. In consequence, the task is performed without carrying the glass.

In the main, the path points defined in section 5 have been implemented as a list and the robot goes successively from one to the next by mean of a for loop. Opening and closing of the gripper as well as pouring motion of θ_7 are handled with the help of if statements inside the loop.

7 Conclusion

To conclude, this project was solved successfully. The direct and inverse kinematic has been calculated, implemented, and validated using Matlab and a ROS simulator. Then, a pouring motion of an imaginative glass into a bowl has been shown in the simulator.
This project was a great application of the theoretical concept seen in the Robotics course. Two approaches have been proposed to solve the inverse kinematic, which was the most challenging part.The first approach was based on kinematic decoupling with 6 DOF. The second approach, better suited to the pouring task, was based on a geometric method with 4 DOF. Once this part was validated, the implementation of the inverse kinematic equations into a python code was an interesting way of discovering ROS and gazebo. Seeing the robot moving in this simulator was very satisfying and fun to see. Many thanks to the assistant Dr. ir. Ilias El Makrini for his precious help during the sessions.