UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

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

MODELLING AND CONTROL OF MANIPULATORS

Third Assignment

Jacobian Matrices and Inverse Kinematics

Author:

Di Donna Alberto
Dondero Enrico

Student ID:

s4944656
s4938123

Professors:

Giovanni Indiveri
Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Francesco Giovinazzo
George Kurshakov

November 4, 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
${}^{a}O_{b}$	Vector defining frame $< b >$ wit respect to frame $< a >$	aOb

Table 1: Nomenclature Table

1 Assignment description

The third assignment of Modelling and Control of Manipulators focuses on the definition of the Jacobian matrices for a robotic manipulator and the computation of its inverse kinematics.

The third assignment consists of three exercises. You are asked to:

- Download the .zip file called MOCOM-LAB3 from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling in the predefined files. In particular, you will find two different main files: *"ex1.m"* for the first exercise and *"ex2_ex3.m"* for the second and third exercises.
- Write a report motivating your answers, following the predefined format on this document.
- **Putting code in the report is not an explanation!**

1.1 Exercise 1

Given the CAD model of the robotic manipulator from the previous assignment and using the functions already implemented:

Q1.1 Compute the Jacobian matrices for the manipulator for the following joint configurations:

- $\mathbf{q}_1 = [1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8]$
- $\mathbf{q}_2 = [0.3, 1.4, 0.1, 2.0, 0, 1.3, 0]$
- $\mathbf{q}_3 = [0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.]$
- $\mathbf{q}_4 = [1, 1, 1, 1, 1, 1, 1]$

1.2 Exercise 2

In the second exercise the model of a Panda robot by Franka Emika is provided. The robot geometry and jacobians can be easily retrieved by calling the following built-in functions: *"getTransform()"* and *"geometricJacobian()"*.

Q2.1 Compute the cartesian error between the robot end-effector frame ${}^b_eT$ and the goal frame ${}^b_gT$.
${}^b_gT$ must be defined knowing that:

- The goal position with respect to the base frame is ${}^bO_g = [0.55, -0.3, 0.2]^\top (m)$
- The goal frame is rotated of $\theta = \pi/6$ around the y-axis of the robot end-effector initial configuration.

Q2.2 Compute the desired angular and linear reference velocities of the end-effector with respect to the base: ${}^b\nu^*_{e/0} = \alpha \cdot \begin{bmatrix} \omega^*_{e/0} \\ v^*_{e/0} \end{bmatrix}$, such that $\alpha = 0.2$ is the gain.

Q2.3 Compute the desired joint velocities. (Suggested matlab function: *"pinv()"*).

Q2.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"*.

1.3 Exercise 3

Repeat the Exercise 2, by considering a tool frame rigidly attached to the robot end-effector according to the following specifications:

$$eRt = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 & 0 \\ sin(\phi) & cos(\phi) & 0 & 0 \\ 0 & 0 & 1 & \end{bmatrix}, \phi = -44.98(deg), {}^eO_t = [0, 0, 21.04]^\top (cm)$$

Q3.1 Compute the cartesian error between the robot tool frame ${}^b_tT$ and the goal frame ${}^b_gT$.
${}^b_{gt}T$ must be defined knowing that:

- The goal position with respect to the base frame is ${}^bO_g = [0.55, -0.3, 0.2]^\top (m)$
- The goal frame is rotated of $\theta = \pi/6$ around the y-axis of the robot tool frame initial configuration.

Q3.2 Compute the angular and linear reference velocities of the tool with respect to the base: ${}^b\nu^*_{e/0} = \alpha \cdot \begin{bmatrix} \omega^*_{e/0} \\ v^*_{e/0} \end{bmatrix}$, such that $\alpha = 0.2$ is the gain.
Q3.3 Compute the desired joint velocities. (Suggested matlab function: *"pinv()"*).
Q3.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"*.
Q3.5 Comment the differences with respect to Exercise2.

2 Exercise 1

This assignment involved the development and use of tools to compute forward and inverse kinematics for open chain mechanisms through the computation of the Jacobians and inverse Jacobians. The first part was done using the robot model developed in the second assignment, while the second part made use of the pre-built model of a real machine called Panda by Franka Robotics (former Franka Emika).

2.1 Task 1.1

The completition of this task was subordinated to the development of a function to compute the Jacobian matrix for a given configuration of the robot. The starting point are the rules to build the matrix itself. To build this it is necessary to understand how changes in the joint configuration vector affects the linear and angular velocity of the end-effector.

$$\begin{bmatrix} {}^{\mathbf{o}}\omega_{\mathbf{e/o}} \\ {}^{\mathbf{o}}\mathbf{v}_{\mathbf{E/O}} \end{bmatrix} = {}^{o}J \cdot \dot{\mathbf{q}} \tag{1}$$

By construction, the Jacobian matrix must be $6 \times n$, where n is the number of joints in the chain. It follows that the problem of inverse kinematics is ill-posed as not any velocity of the end-effector may be achievable due to an insufficient number of joints or singular configurations of the chain.
The matrix can be divided into an angular part and a linear one:

$${}^{o}J^{A}_{e/i} = \begin{cases} {}^{o}\mathbf{k}_i & \Gamma_i = R \\ 0 & \Gamma_i = P \end{cases} \qquad {}^{o}J^{L}_{e/i} = \begin{cases} {}^{o}(\mathbf{k}_i \times \mathbf{r}_{E/O_i}) & \Gamma_i = R \\ {}^{o}\mathbf{k}_i & \Gamma_i = P \end{cases} \tag{2}$$

Where ${}^{o}\mathbf{k}_i$ is the axis of rotation for revolut joints (or direction of translation for prismatic joints). Γ_i is the i-th joint type, and $\mathbf{r}_{E/0}$ is the position vector of the end-effector w.r.t the i-th frame.
The full Jacobian is built as follows:

$${}^{o}J_{e/o} = \begin{bmatrix} {}^{o}J^{A}_{e/o} \\ {}^{o}J^{L}_{e/o} \end{bmatrix}$$

The *GetJacobian()* function followed the structure above for the construction of the Jacobian as the angular and linear parts were filled separately, joint-by-joint. This means the matrix was built column by column, where each one represents ${}^{o}J_{e/i}$. The first step for each iteration was to compute the transformation matrix of frame $< i >$ w.r.t the base.
Using the function *GetTransformationWrtBase()* developed in the previews assignments. Knowing that by construction $\mathbf{k}_i$ for every frame corresponded to the axis of rotation (or direction of translation), to obtain ${}^{o}\mathbf{k}_i$ it was sufficient to extract the first three elements of the third columns of said transformation matrix, knowing it to be equivalent to:

$${}^{o}\mathbf{k}_i = {}^{i}_{o}R \cdot i^{\mathbf{k}}_{i} \tag{3}$$

for revolute joints, and zero otherwise. While for the linear part, it was necessary to compute $\mathbf{r}_{E/i}$. This was done as follows:

$${}^{o}\mathbf{r}_{E/O_i} = {}^{o}\,\mathbf{r}_{E/O} - {}^{o}\,\mathbf{r}_{O_i/O} \tag{4}$$

Where the two vectors were extracted from the corresponding transformation matrices: ${}^{o}_{e}T$, ${}^{o}_{i}T$.

Once defined the function to compute the Jacobian, it was possible to put it through the test using one joint variables vector $\mathbf{q} = [1.3, 1.3, 1.3, 1.3, 1.3, 1.3]$ a starting configuration with transformation matrix w.r.t base equal to:

$${}^{o}_{e}T = \begin{bmatrix} -0.8236 & -0.5301 & -0.2017 & 169.0722 \\ 0.1179 & 0.1880 & -0.9751 & -362.6284 \\ 0.5548 & -0.8268 & -0.0923 & 547.8475 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{5}$$

1.1.a Given $\mathbf{q}_a = [1.8, 1.8, 1.8, 1.8, 1.8, 1.8, 1.8]$ with corresponding transformation matrix of frame $< e >$ w.r.t $< 0 >$:

$${}^{o}_{e}T = \begin{bmatrix} -0.0415 & -0.0397 & 0.9983 & -691.8250 \\ -0.9983 & 0.0432 & -0.0398 & 23.7273 \\ -0.0416 & -0.9983 & -0.0415 & 429.8602 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{6}$$

It easily follows that: ${}^{o}\mathbf{r}_{E/O} = [-691.8250, 23.7273, 429.8602]$.
Computing:

$${}^{o}\mathbf{k}_i = {}^{o}_{i}R\,{}^{i}\mathbf{k}_i \tag{7}$$

given that the choice of reference frames is such that $\forall i \in [1,n], \quad {}^{i}\mathbf{k}_i = [0,0,1]^T$, as previously stated is equal to extracting the third column of ${}^{o}_{i}R$, or the first three elements of the third column of the corresponding transformation matrix.
Each individual position vector of the EE frame w.r.t every frame and every axis vector are computed below:

$$ {}^{o}\mathbf{r}_{E/O_1} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_1/O} = \begin{bmatrix} -691.8250 \\ 23.7273 \\ 254.8602 \end{bmatrix}, \quad {}^{o}\mathbf{k}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{8} $$

$$ {}^{o}\mathbf{r}_{E/O_2} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_2/O} = \begin{bmatrix} -691.8250 \\ 23.7273 \\ 156,8602 \end{bmatrix}, \quad {}^{o}\mathbf{k}_2 = \begin{bmatrix} -0.0416 \\ 0.9991 \\ 0 \end{bmatrix} \tag{9} $$

$$ {}^{o}\mathbf{r}_{E/O_3} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_3/O} = \begin{bmatrix} -587.0066 \\ 19.3651 \\ 152.4943 \end{bmatrix}, \quad {}^{o}\mathbf{k}_3 = \begin{bmatrix} -0.9983 \\ 0.0415 \\ 0.0416 \end{bmatrix} \tag{10} $$

$$ {}^{o}\mathbf{r}_{E/O_4} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_4/O} = \begin{bmatrix} -266.8590 \\ 12.0969 \\ -6.3303 \end{bmatrix}, \quad {}^{o}\mathbf{k}_4 = \begin{bmatrix} 0.0433 \\ 0.9982 \\ 0.0415 \end{bmatrix} \tag{11} $$

$$ {}^{o}\mathbf{r}_{E/O_5} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_5/O} = \begin{bmatrix} -231.8919 \\ 10.5811 \\ -6.3315 \end{bmatrix}, \quad {}^{o}\mathbf{k}_5 = \begin{bmatrix} -0.9991 \\ 0.0433 \\ 0 \end{bmatrix} \tag{12} $$

$$ {}^{o}\mathbf{r}_{E/O_6} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_6/O} = \begin{bmatrix} 152.7469 \\ -6.0925 \\ -6.3454 \end{bmatrix}, \quad {}^{o}\mathbf{k}_6 = \begin{bmatrix} -0.0432 \\ -0.9956 \\ -0.0831 \end{bmatrix} \tag{13} $$

$$ {}^{o}\mathbf{r}_{E/O_7} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_7/O} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad {}^{o}\mathbf{k}_7 = \begin{bmatrix} 0.9983 \\ -0.0398 \\ -0.0415 \end{bmatrix} \tag{14} $$

$$ \tag{15} $$

As can be seen above, there are certain elements with significant values such as ${}^{o}\mathbf{k}_1$ which stays fixed with respect to base for all configurations of the robot, while in the formula (15), the vector ${}^{o}\mathbf{r}_{E/O_7} = {}^{o}\mathbf{r}_{E/O} - {}^{o}\mathbf{r}_{O_7/O}$ is always zero since $E \equiv O_7$
The overall Jacobian is:

$$ {}^{o}J_{e/o} = \begin{bmatrix} 0 & -0.0416 & -0.9983 & 0.0433 & -0.9991 & -0.0432 & 0.9983 \\ 0 & -0.9991 & 0.0415 & 0.9982 & 0.0433 & -0.9956 & -0.0398 \\ 1.0000 & 0 & 0.0416 & 0.0415 & 0.0000 & -0.0831 & -0.0415 \\ -23.7273 & -156.7246 & 5.5301 & -6.8214 & -0.2746 & 5.8115 & 0 \\ -691.8250 & 6.5224 & 127.8225 & -10.8127 & -6.3339 & -12.9600 & 0 \\ 0 & -692.2133 & 5.0554 & 266.9019 & -0.5284 & 152.3393 & 0 \end{bmatrix} \tag{16} $$

1.1.b

q=[0.3, 1.4, 0.1, 2.0, 0, 1.3, 0], with corresponding transformation matrix

$$ {}^{o}_{e}T = \begin{bmatrix} 0.1911 & 0.9799 & -0.0564 & -239.9942 \\ -0.0810 & -0.0415 & -0.9959 & 621.6542 \\ -0.9782 & 0.1949 & 0.0715 & 613.0600 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{17} $$

From which is extracted ${}^{o}\mathbf{r}_{E/O} = [-239.9942, 621.6542, 613.0600]$.

Each building element of the Jacobian is computed below:

$$ {}^o\mathbf{r}_{E/O_1} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_1/O} = \begin{bmatrix} -239.9942 \\ 621.6542 \\ 438.0600 \end{bmatrix}, \quad {}^o\mathbf{k}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{18} $$

$$ {}^o\mathbf{r}_{E/O_2} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_2/O} = \begin{bmatrix} -239.9942 \\ 621.6542 \\ 340.0600 \end{bmatrix}, \quad {}^o\mathbf{k}_2 = \begin{bmatrix} -0.9996 \\ -0.0292 \\ 0 \end{bmatrix} \tag{19} $$

$$ {}^o\mathbf{r}_{E/O_3} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_3/O} = \begin{bmatrix} -237.2223 \\ 526.7671 \\ 295.1851 \end{bmatrix}, \quad {}^o\mathbf{k}_3 = \begin{bmatrix} -0.0264 \\ 0.9037 \\ 0.4274 \end{bmatrix} \tag{20} $$

$$ {}^o\mathbf{r}_{E/O_4} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_4/O} = \begin{bmatrix} -84.9728 \\ 225.3354 \\ 178.0035 \end{bmatrix}, \quad {}^o\mathbf{k}_4 = \begin{bmatrix} -0.1576 \\ -0.4259 \\ 0.8909 \end{bmatrix} \tag{21} $$

$$ {}^o\mathbf{r}_{E/O_5} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_5/O} = \begin{bmatrix} -78.6103 \\ 193.8603 \\ 164.0809 \end{bmatrix}, \quad {}^o\mathbf{k}_5 = \begin{bmatrix} -0.1818 \\ 0.8993 \\ 0.3978 \end{bmatrix} \tag{22} $$

$$ {}^o\mathbf{r}_{E/O_6} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_6/O} = \begin{bmatrix} -8.6224 \\ -152.3652 \\ 10.9320 \end{bmatrix}, \quad {}^o\mathbf{k}_6 = \begin{bmatrix} -0.8931 \\ -0.0183 \\ 0.4495 \end{bmatrix} \tag{23} $$

$$ {}^o\mathbf{r}_{E/O_7} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_7/O} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad {}^o\mathbf{k}_7 = \begin{bmatrix} -0.0564 \\ -0.9959 \\ 0.0715 \end{bmatrix} \tag{24} $$

$$ \tag{25} $$

With overall Jacobian matrix:

$$ {}^oJ_{e/o} = \begin{bmatrix} 0 & -0.9996 & -0.0264 & -0.1576 & -0.1818 & 0.8931 & -0.0564 \\ 0 & -0.0292 & 0.9037 & -0.4259 & 0.8993 & -0.0183 & -0.9959 \\ 1.0000 & 0 & 0.4274 & 0.8909 & 0.3978 & 0.4495 & 0.0715 \\ -621.6542 & -9.9296 & 41.6252 & -276.5749 & 70.4402 & 68.2865 & 0 \\ -239.9942 & 339.9150 & -93.5916 & -47.6510 & -1.4426 & -13.6390 & 0 \\ 0 & -628.3968 & 200.4688 & -71.7059 & 35.4520 & -136.2348 & 0 \end{bmatrix} \tag{26} $$

1.1.c

q=[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0], with corresponding transformation matrix

$$ {}^o_eT = \begin{bmatrix} -0.0415 & -0.0397 & 0.9983 & -691.8250 \\ -0.9983 & 0.0432 & -0.0398 & 23.7273 \\ -0.0416 & -0.9983 & -0.0415 & 429.8602 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{27} $$

From which is extracted ${}^o\mathbf{r}_{E/O} = [-691.825, 23.7273, 429.8602]$.

Each building element of the Jacobian is computed below:

$$ {}^o\mathbf{r}_{E/O_1} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_1/O} = \begin{bmatrix} -691.8250 \\ 23.7273 \\ 254.8602 \end{bmatrix}, \quad {}^o\mathbf{k}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{28} $$

$$ {}^o\mathbf{r}_{E/O_2} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_2/O} = \begin{bmatrix} -691.8250 \\ 23.7273 \\ 156.8602 \end{bmatrix}, \quad {}^o\mathbf{k}_2 = \begin{bmatrix} -0.0416 \\ -0.9991 \\ 0 \end{bmatrix} \tag{29} $$

$$ {}^o\mathbf{r}_{E/O_3} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_3/O} = \begin{bmatrix} -587.0066 \\ 19.3651 \\ 152.4943 \end{bmatrix}, \quad {}^o\mathbf{k}_3 = \begin{bmatrix} -0.9983 \\ 0.0415 \\ 0.0416 \end{bmatrix} \tag{30} $$

$$ {}^o\mathbf{r}_{E/O_4} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_4/O} = \begin{bmatrix} -266.8590 \\ 12.0969 \\ -6.3303 \end{bmatrix}, \quad {}^o\mathbf{k}_4 = \begin{bmatrix} 0.0433 \\ 0.9982 \\ 0.0415 \end{bmatrix} \tag{31} $$

$$ {}^o\mathbf{r}_{E/O_5} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_5/O} = \begin{bmatrix} -231.8919 \\ 10.5811 \\ -6.3315 \end{bmatrix}, \quad {}^o\mathbf{k}_5 = \begin{bmatrix} -0.0432 \\ -0.9956 \\ -0.0831 \end{bmatrix} \tag{32} $$

$$ {}^o\mathbf{r}_{E/O_6} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_6/O} = \begin{bmatrix} 152.7469 \\ -6.0925 \\ -6.3454 \end{bmatrix}, \quad {}^o\mathbf{k}_6 = \begin{bmatrix} -0.0432 \\ -0.9956 \\ -0.0831 \end{bmatrix} \tag{33} $$

$$ {}^o\mathbf{r}_{E/O_7} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_7/O} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad {}^o\mathbf{k}_7 = \begin{bmatrix} 0.9983 \\ -0.0398 \\ -0.0415 \end{bmatrix} \tag{34} $$

$$ \tag{35} $$

With overall Jacobian matrix:

$$ {}^oJ_{e/o} = \begin{bmatrix} 0 & -0.0416 & -0.9983 & 0.0433 & -0.9991 & -0.0432 & 0.9983 \\ 0 & -0.9991 & 0.0415 & 0.9982 & 0.0433 & -0.9956 & -0.0398 \\ 1.0000 & 0 & 0.0416 & 0.0415 & 0.0000 & -0.0831 & -0.0415 \\ -23.7273 & -156.7246 & 5.5301 & -6.8214 & -0.2746 & 5.8115 & 0 \\ -691.8250 & 6.5224 & 127.8225 & -10.8127 & -6.3339 & -12.9600 & 0 \\ 0 & -692.2133 & 5.0554 & 266.9019 & -0.5284 & 152.3393 & 0 \end{bmatrix} \tag{36} $$

1.1.d

q=[0, 0.1, 0.2, 0.3, 0.4, 0.5, 0], with corresponding transformation matrix

$$ {}^o_eT = \begin{bmatrix} -0.5555 & 0.2023 & 0.8065 & -154.6401 \\ -0.5332 & 0.6576 & -0.5322 & 417.4497 \\ -0.6381 & -0.7257 & -0.2574 & 689.6159 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{37} $$

From which is extracted ${}^o\mathbf{r}_{E/O} = [-154.6401, 417.4497, 689.6159]$.

Each building element of the Jacobian is computed below:

$$ {}^o\mathbf{r}_{E/O_1} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_1/O} = \begin{bmatrix} -154.6401 \\ 417.4497 \\ 514.6159 \end{bmatrix}, \quad {}^o\mathbf{k}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{38} $$

$$ {}^o\mathbf{r}_{E/O_2} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_2/O} = \begin{bmatrix} -154.6401 \\ 417.4497 \\ 416.6159 \end{bmatrix}, \quad {}^o\mathbf{k}_2 = \begin{bmatrix} -0.7457 \\ -0.6663 \\ 0 \end{bmatrix} \tag{39} $$

$$ {}^o\mathbf{r}_{E/O_3} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_3/O} = \begin{bmatrix} -108.0281 \\ 365.2809 \\ 338.3168 \end{bmatrix}, \quad {}^o\mathbf{k}_3 = \begin{bmatrix} -0.4439 \\ 0.4968 \\ 0.7457 \end{bmatrix} \tag{40} $$

$$ {}^o\mathbf{r}_{E/O_4} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_4/O} = \begin{bmatrix} 69.6563 \\ 329.2597 \\ 30.2532 \end{bmatrix}, \quad {}^o\mathbf{k}_4 = \begin{bmatrix} 0.8673 \\ 0.0293 \\ 0.4968 \end{bmatrix} \tag{41} $$

$$ {}^o\mathbf{r}_{E/O_5} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_5/O} = \begin{bmatrix} 74.1350 \\ 295.0360 \\ 24.4498 \end{bmatrix}, \quad {}^o\mathbf{k}_5 = \begin{bmatrix} -0.128 \\ 0.9778 \\ 0.1658 \end{bmatrix} \tag{42} $$

$$ {}^o\mathbf{r}_{E/O_6} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_6/O} = \begin{bmatrix} 123.4004 \\ -81.4249 \\ -39.3867 \end{bmatrix}, \quad {}^o\mathbf{k}_6 = \begin{bmatrix} -0.2192 \\ 0.1352 \\ -0.9663 \end{bmatrix} \tag{43} $$

$$ {}^o\mathbf{r}_{E/O_7} = {}^o\mathbf{r}_{E/O} - {}^o\mathbf{r}_{O_7/O} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad {}^o\mathbf{k}_7 = \begin{bmatrix} 0.8065 \\ -0.5322 \\ -0.2574 \end{bmatrix} \tag{44} $$

$$ \tag{45} $$

With overall Jacobian matrix:

$$ {}^oJ_{e/o} = \begin{bmatrix} 0 & -0.7457 & -0.4439 & 0.8673 & -0.1280 & -0.2192 & 0.8065 \\ 0 & -0.6663 & 0.4968 & 0.0293 & 0.9778 & 0.1352 & -0.5322 \\ 1.0000 & 0 & 0.7457 & 0.4968 & 0.1658 & -0.9663 & -0.2574 \\ -417.4497 & -277.5812 & -104.3007 & -162.7062 & -25.0121 & -84.0017 & 0 \\ -154.6401 & 310.6726 & 69.6297 & 8.3685 & 15.4209 & -127.8723 & 0 \\ 0 & -414.3274 & -108.4836 & 283.5442 & -110.2442 & 1.1715 & 0 \end{bmatrix} \tag{46} $$

3 Exercise 2

The second exercise was based around a real manipulator platform called Panda, from Franka Robotics. This is a 7-Dof robot with only revolute joints configured in an open chain mechanism.

2.1

Our task was to compute the cartesian error between the end-effector frame and the goal frame, given the position vector of the goal frame ${}^oO_g$, and its rotation w.r.t the e-e frame of $\pi/6$, around the y-axis of the robot end-effector initial configuration, corresponding to a rotation matrix:

$$ {}^e_gR = \begin{bmatrix} cos(\pi/6) & 0 & sin(\pi/6) \\ 0 & 1 & 0 \\ -sin(\pi/6) & 0 & cos(\pi/6) \end{bmatrix} \tag{47} $$

It was possible to obtain the transformation matrix of the goal w.r.t the base, ${}^o_gT$, by using the built-in function *getTransform()* to retrieve the transformation matrix of the e-e w.r.t. the base, ${}^o_eT$, from which we can obtain the rotation matrix ${}^o_eR$ (that is the square matrix in the first three rows and columns of the transformation matrix). Then multiplying the latter by the rotation matrix of the goal w.r.t. the e-e, ${}^e_gR$, we get the rotation matrix of the goal w.r.t. the base ${}^o_gR$:

$$ {}^o_gR = {}^o_e\,R \cdot {}^e_g\,R \tag{48} $$

Finally the transformation matrix of the goal w.r.t. the base can be computed using the position vector of the goal frame ${}^oO_g$, in fact by construction:

$$ {}^o_gT = \begin{bmatrix} {}^o_gR & {}^oO_g \\ 0_{3x3} & 1 \end{bmatrix} \tag{49} $$

2.1

Once finished the initialization of the variables it was necessary to create a loop to allow the motion of the robot.
Here during each iteration (so every time sample):

- The transformation matrix ${}^{o}_{e}T$ is recomputed, always using *getTransform()*, because during the motion the configuration of the robot changes.
- The jacobian matrix, is computed using the built-in function *geometricJacobian()* and extracting the necessary information from the returned matrix, which are the first seven columns because the file *panda.mat* has a RobotTree composed of 9 frames (7+the tool's fingers)
- The linear error is the position vector from the EE frame to the goal frame. ${}^{o}O_{g}$ is given, while ${}^{o}O_{e}$ can be retrieved by the transformation matrix ${}^{b}_{e}T$, indeed it is in the first three rows of the last column of this matrix. Now the linear error is the difference between ${}^{o}O_{g}$ and ${}^{o}O_{e}$

$$\text{linear error} \quad = \quad {}^{o}O_{g} - {}^{o}O_{e} \tag{50}$$

- The angular error is the angle between the EE frame and the goal frame around the axis vector. Here we have used the *ComputeInverseAngleAxis()* function, which has been defined in the first laboratory, to obtain the equivalent Angle-axis representation of the rotation matrix ${}^{e}_{g}R$. The resulting rotation vector will go to zero as the EE frame approaches the same orientation as the goal frame.

$$\rho \to 0 \iff {}^{e}_{g}R \to \mathbb{I} \tag{51}$$

The rotation matrix ${}^{e}_{g}R$ can be computed by multiplying the rotation matrix of the base w.r.t. the e-e, ${}^{e}_{o}R$ (obtained by doing the transpose of the rotation matrix ${}^{b}_{e}R$ that can be retrieved from the transformation matrix ${}^{o}_{e}T$), and the rotation matrix of the goal w.r.t. the base, ${}^{o}_{g}R$.

$$ {}^{e}_{g}R = {}^{e}_{o}R\,{}^{o}_{g}R \tag{52}$$

Finally the angle error can be computed by projecting on the base frame the product of θ and *v* (transpose)

$$\text{angle error} \quad = \quad {}^{o}_{e}R \cdot (\theta \cdot v) = \quad {}^{o}_{e}R\,\rho \tag{53}$$

2.2

Now we have to compute the desired linear and angular velocities of the end-effector with respect to the base and we need to do it inside the same foor loop defined before.

Given the linear error, that we can call *r*, and the linear gain α, we can define the desired derivative of *r*,

$$\dot{\mathbf{r}} \triangleq -\alpha \cdot \mathbf{r}$$

But we can also say that (by definition of *r*):

$$\dot{\mathbf{r}} = v_{g/o} - v_{e/o}$$

So we can conclude that the desired velocity for the end-effector w.r.t. the base is:

$$v_{e/o} = v_{g/o} + \alpha \cdot \mathbf{r}$$

Since the velocity of the goal w.r.t. the base is zero $v_{e/o} = \alpha \cdot \mathbf{r}$

In the same way given the angle error, that we can call ρ, we can define the desired derivative of ρ w.r.t. e-e frame:

$$\dot{\rho} = -\alpha \cdot \rho$$

So we can state that the angular velocity of the goal w.r.t. the e-e is:

$$\omega_{g/e} = -\alpha \cdot \rho$$

that, decomposing the velocity on the base frame, can be rewritten in this way:

$$\omega_{g/o} - \omega_{e/o} = -\alpha \cdot \rho$$

and so we can conclude that the reference behaviour for $\omega_{e/o}$ is:

$$\omega_{e/o} = \omega_{g/o} + \alpha \cdot \rho$$

but $\omega_{g/o}$ is zero so $\omega_{e/o} = \alpha \cdot \rho$

2.3

The next step is to compute the desired joint velocities and we need to do it always in the same for loop. Knowing $\omega_{e/o}$ and $v_{e/o}$ we can store them in a unique vector:

$$\dot{\bar{x}} = \begin{bmatrix} \omega_{e/o} \\ v_{e/o} \end{bmatrix} \tag{54}$$

And, in general, we can write the following linear problem:

$$\dot{x} = {}^{o}J_{e/o} \cdot \dot{q}$$

Starting from this equation we want to know which $\dot{q}$ generate our desired velocity $\dot{\bar{x}}$, in particular we want to generate the smallest vector $\dot{\bar{q}}$ such that the following equation is satisfied:

$$\dot{\bar{x}} = {}^{o}J_{e/o} \cdot \dot{\bar{q}}$$

In fact, if our robot has more than six degrees of freedom the inverse problem has ∞^{n-6} solutions (where n is the number of degrees of freedom). To do that we have used the *pinv()* matlab function that takes as input the jacobian and returns the pseudo-inverse of the jacobian, that if we call it $J^{\#}$ we can finally write:

$$\dot{\bar{q}} = J^{\#} \cdot \dot{\bar{x}}$$

2.4

In the end, always in the same for loop, we have to compute the new vector of configurations q. This computation has been made in the function *KinematicSimulation()*. This function takes as input: the current robot configuration q, the joints velocity q_dot, the sample time ts, the lower joints bound q_min, and the upper joints bound q_max. While it returns the update q.
The function computes the joint variables vector by integrating numerically its derivative.

$$q_{new} = q + q\_dot \cdot ts$$

This takes into account old contributions while adding the new one built as the product between *ts* and the derivative of **q** for that particular time step.
if q_{new} is bigger than q_max, then q_{new} is set to be equal to q_max. In the same way, if q_{new} is smaller than q_min, then q_{new} is set to be equal to q_min.
This is due to the physical limitations of the machine and so to avoid saturation

Results

The transformation matrix ${}^{b}_{e}T$ of the manipulator in the initial configuration is:

$${}^{b}_{e}T = \begin{bmatrix} 0.7298 & -0.6824 & -0.0412 & 0.3141 \\ -0.6834 & -0.7299 & -0.0163 & -0.0047 \\ -0.0190 & 0.0401 & -0.9990 & 0.6934 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{55}$$

It is possible to see the results by comparing ${}^{b}_{e}T$ (with the manipulator in the final configuration) and ${}^{b}_{g}T$:

$${}^{b}_{e}T = \begin{bmatrix} 0.6538 & -0.6822 & 0.3273 & 0.5483 \\ -0.5845 & -0.7301 & -0.3540 & -0.2982 \\ 0.4804 & 0.0402 & -0.8761 & 0.2026 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{56}$$

$${}^{b}_{g}T = \begin{bmatrix} 0.6526 & -0.6824 & 0.3292 & 0.5500 \\ -0.5837 & -0.7299 & -0.3558 & -0.3000 \\ 0.4830 & 0.0400 & -0.8746 & 0.2000 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{57}$$

As can be seen, the end-effector frame reaches the goal position and orientation with sufficient precision. This can be seen also by looking the angular error and the linear error with the robot in the final configuration, in fact they are small enough:

$$\text{angular\_error} = \begin{bmatrix} -0.0019 \\ -0.0022 \\ -0.0001 \end{bmatrix}, \text{linear\_error} \quad = \begin{bmatrix} 0.0017 \\ -0.0018 \\ -0.0026 \end{bmatrix}.$$

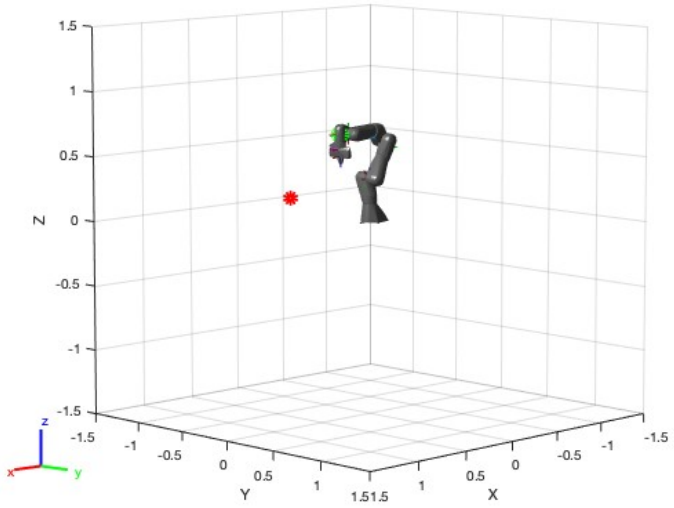

Figure 1: initial configuration

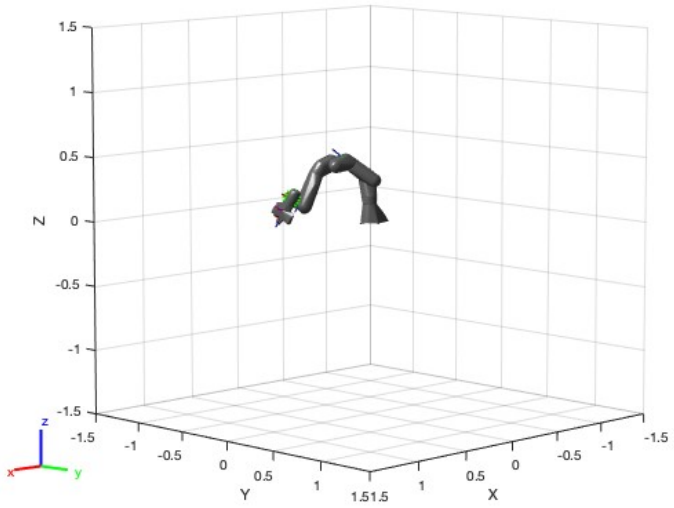

Figure 2: final configuration

4 Excercise 3

This task was accomplished similarly to the second one. In addition to previous hypotheses, for this exercise was available information about the tool attached to the EE.
Given that the tool frame was positioned at $e_t^O = [0, 0, 21.4]^T$ centimeters w.r.t the EE frame (was then converted to meters) and with rotation matrix:

$$^e_tR = \begin{bmatrix} cos(\phi) & -sin(\phi) & 0 \\ sin(\phi) & cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{58}$$

Where $\phi = -44.98°$ and thus was converted to radiants using the built-in function *deg2rad()*.
Lastly, the goal frame was rotated by $\pi/6$ radiants around the y-axis of the tool frame:

$$ {}^{t}_{g}R = \begin{bmatrix} cos(\pi/6) & 0 & sin(\pi/6) \\ 0 & 1 & 0 \\ -sin(\pi/6) & 0 & cos(\pi/6) \end{bmatrix} \tag{59} $$

To obtain ${}^{0}_{g}R$ it was sufficient to multiply rotation matrices already at hand:

$$ {}^{0}_{g}R = {}^{0}_{t}R\,{}^{t}_{g}R \tag{60} $$

This result together with ${}^{0}O_{g}$, provided by the specifics allowed the computation of ${}^{0}_{g}T$:

$$ {}^{0}_{g}T = \begin{bmatrix} {}^{0}_{g}R & {}^{0}O_{g} \\ 0_{3\times3} & 1 \end{bmatrix} = \begin{bmatrix} 0.8854 & 0.0332 & 0.4636 & 0.5500 \\ 0.0363 & -0.9993 & 0.0022 & -0.3000 \\ 0.4634 & 0.0149 & -0.8860 & 0.2000 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{61} $$

3.1
To compute the cartesian error at each time step, a similar approach was followed to the second task:

$$ lin\_err = {}^{0}O_{g} - {}^{0}O_{t} \tag{62} $$

$$ ang\_err = {}^{0}_{t}R \cdot \rho \tag{63} $$

Where ρ is the rotation vector obtained from the *ComputeInverseAngleAxis()* function. The premultiplication is used to project this vector onto the base frame.
The major difference between the two tasks comes with the computation of the Jacobian, given that we wanted to control the linear and angular velocity of the tool, to compute its Jacobian was used the Rigid Body Jacobian Matrix $S_{t/e}$ defined as follows:

$$ S_{t/e} = \begin{bmatrix} \mathbb{I}_3 & 0_{3\times3} \\ [{}^{e}O_{t}\times]^{T} & \mathbb{I}_3 \end{bmatrix} \tag{64} $$

The final Jacobian then is:

$$ {}^{0}J_{t} = {}^{e}S_{t}\,{}^{0}J_{e} \tag{65} $$

3.2
After that, as was done in the second task, it was computed the desired velocities vector:

$$ \dot{x} = \begin{bmatrix} {}^{o}\omega^{*}_{t/0} \\ {}^{o}v^{*}_{t/0} \end{bmatrix} = \alpha \cdot \begin{bmatrix} ang\_err \\ lin\_err \end{bmatrix} \tag{66} $$

3.3 Which is then used to compute the desired joint velocities vector $\dot{q}$ using the pseudoinverse of the Jacobian, computed using the built-in function *pinv()*:

$$ \dot{\mathbf{q}} = {}^{0}J^{\#}_{t/0} \cdot \dot{\mathbf{x}} \tag{67} $$

Finally the joint positions vector for the next time step is computed using the *KinematicSimulation()* function.
3.4 Below are the starting and ending pose of the robot arm:

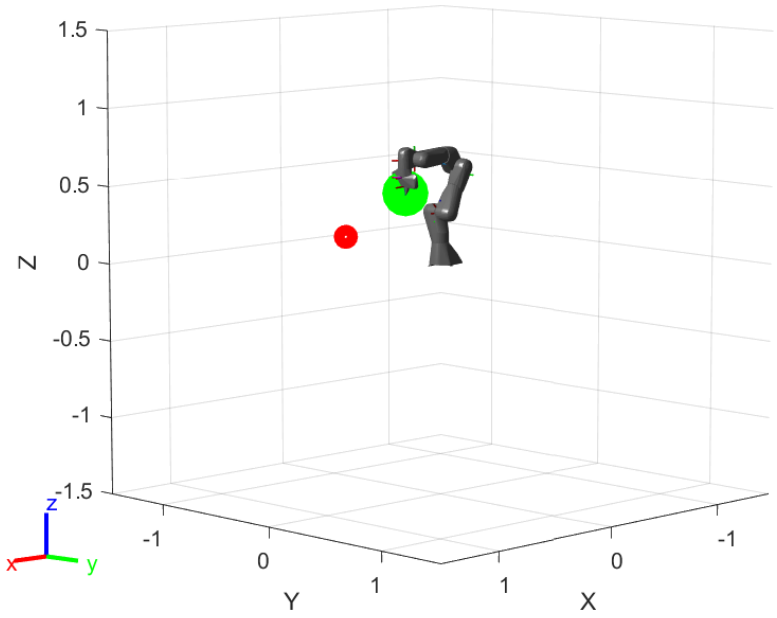

Figure 3: initial configuration

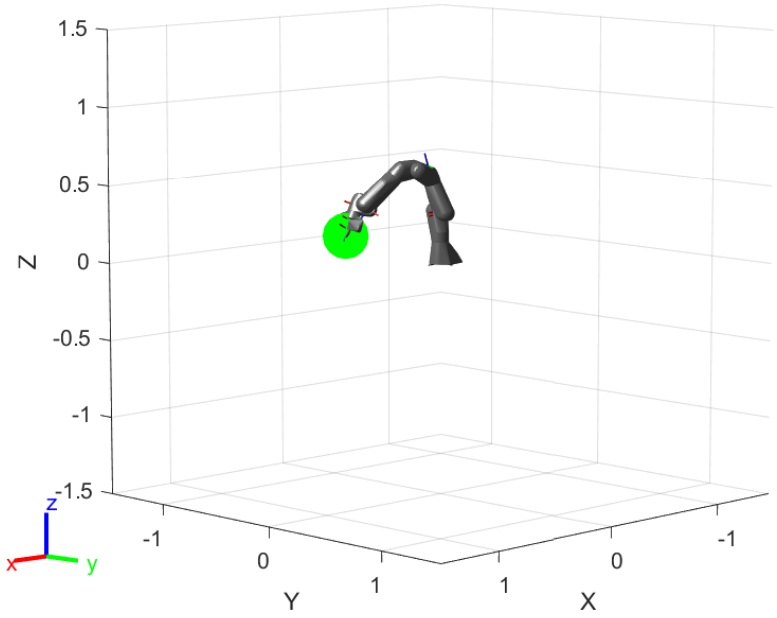

Figure 4: Final configuration

It is possible to see the results by comparing ${}^{b}_{t}T$ and ${}^{b}_{g}T$:

$$ {}^{b}_{t}T = \begin{bmatrix} 0.8863 & 0.0329 & 0.4620 & 0.5534 \\ 0.0363 & -0.9993 & 0.0017 & -0.2981 \\ 0.4618 & 0.0153 & -0.8869 & 0.2017 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} \tag{68} $$

$$ {}^{b}_{g}T = \begin{bmatrix} 0.8854 & 0.0331 & 0.4636 & 0.5500 \\ 0.0363 & -0.9993 & 0.0021 & -0.3000 \\ 0.4633 & 0.0149 & -0.8860 & 0.2000 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{69} $$

As can be seen, the tool frame reaches the goal position and orientation with sufficient precision.

3.5 As previously introduced, the main difference between the second and third task is the computation of the Jacobian. In the last exercise, a tool is applied to the end effector. The added frame can be considered fixed with respect to the EE frame. Given this premise, it can be explained how the Rigid Body Jacobian works since it is known that:

$$^0\omega_{t/0} = ^0\omega_{t/e} - ^0\omega_{e/0}$$

but $^0\omega_{t/e} = 0$ because the link is rigid and the tool is attached to the end-effector.
The same can be written in a matrix form:

$$^0\omega_{t/0} = \begin{bmatrix}\mathbb{I}_3 & 0_{3\times3}\end{bmatrix} \begin{bmatrix}^0\omega_{e/0} \\ ^0v_{e/0}\end{bmatrix} \tag{70}$$

For what regards the linear velocities instead:

$$v_{t/0} = \frac{d_0}{dt}(r_{e/0} + r_{t/e}) = v_{e/0} + \frac{d_e}{dt}(r_{t/e}) + \omega_{e/0} \times r_{t/e}$$

but $\frac{d_e}{dt}(r_{t/e}) = 0$ and the vector product is anti-commutative so:

$$v_{t/0} = v_{e/0} - r_{t/e} \times \omega_{e/0}$$

now we can say that, in general $v_1 \times v_2 = [v_1\times]v_2$ and that $-[v_1\times] = [v_1\times]^T$ and so we can conclude:

$$v_{t/0} = v_{e/0} + [r_{t/e}\times]^T\omega_{e/0}$$

or in matrix form

$$^0v_{t/0} = \begin{bmatrix}[r_{t/e}\times]^T & \mathbb{I}_3\end{bmatrix} \begin{bmatrix}^0\omega_{e/0} \\ ^0v_{e/0}\end{bmatrix} \tag{71}$$

putting togheter the angular part (70) and the linear part (71) we come back to the Rigid Body Jacobian written before (64)