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:

Lovecchio Marco s7647356
Marmolino Giorgio s7721100
Petrosilli Marco s7812048

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Luca Tarasi
George Kurshakov

December 11, 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

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 *template_MATLAB-assignment2* from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling the template classes called *geometricModel* and *kinematicModel*
- Write a report motivating your answers, following the predefined 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 the method of *geometricModel* called *updateDirectGeometry()* which should compute the model matrices as a function of the joint position q. Explain the method used and comment on the results obtained.

Q1.3 Implement the method of *geometricModel* called *getTransformWrtBase()* which should compute the transformation matrix from the base to a given frame. Calculate the following transformation matrices: ${}^{b}_{e}T$, ${}^{5}_{3}T$. Explain the method used and comment on the results obtained.

Q1.4 Implement the method of *kinematicModel* called *updateJacobian()* which should compute the jacobian of a given geometric model considering the possibility of having *rotational* or *prismatic* joints. Compute the Jacobian matrix of the manipulator for the end-effector. Explain the method used and comment on the results obtained.

Remark: The methods should be implemented for a generic serial manipulator. For instance, joint types, and the number of joints should be parameters.

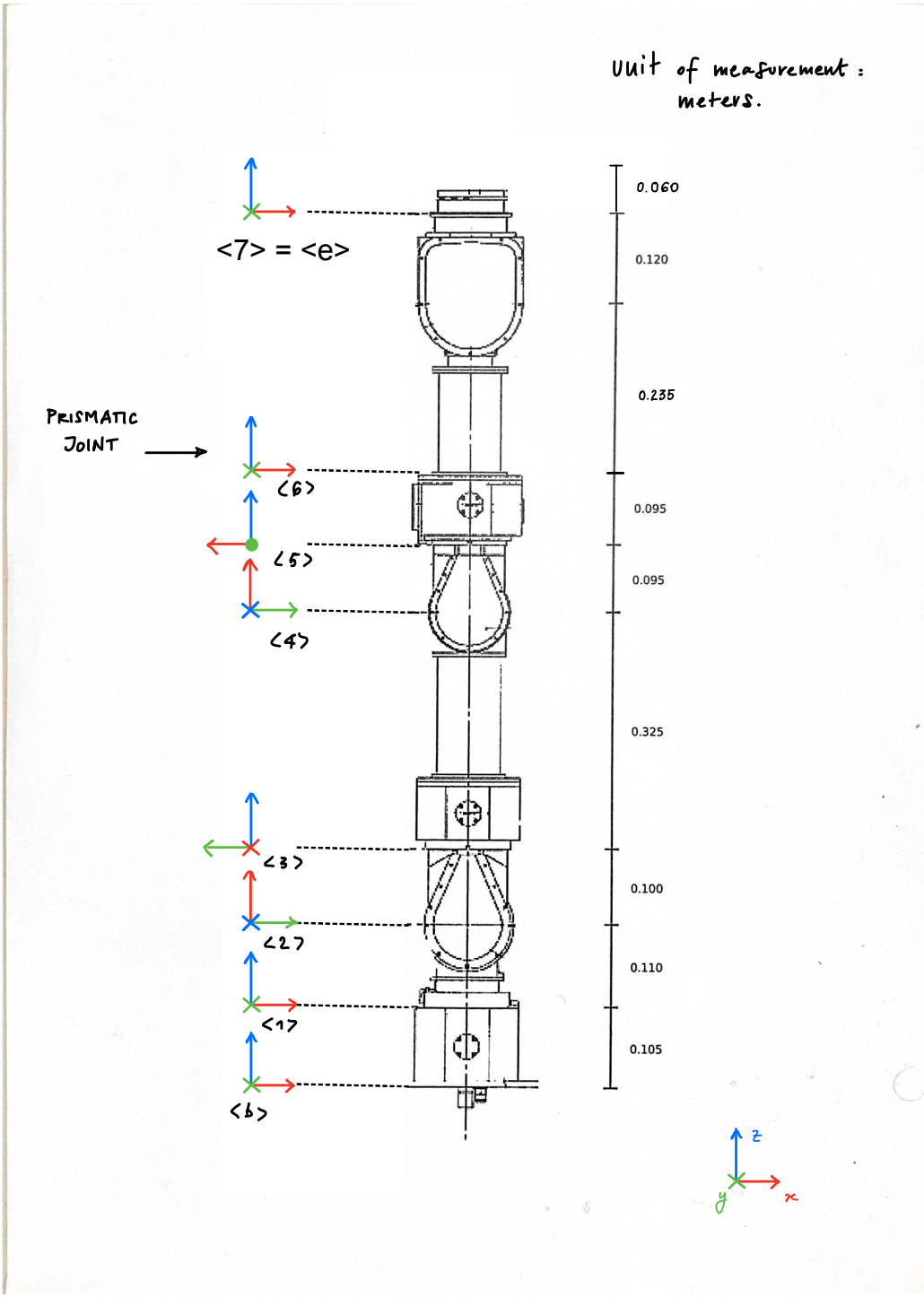


Figure 1: CAD model of the robot

2 Exercise 1.1 - BuildTree() function

In this exercise we compute the transformation matrices along the chain between every consecutive couple from the base frame $< 0 >$ to the end-effector frame $< e >$. To reach this aim we have to keep into account how the structure of a transformation matrix is made. Given two frames $< a >$ and $< b >$ we can write:

$$ {}^a_bT = \begin{pmatrix} {}^a_bR & {}^aO_b \\ 0_{1x3} & 1 \end{pmatrix} $$

Considering:

- The rotation matrix ${}^a_bR \in \mathbb{R}^{3\times3}$: it represent the rotation matrix of frame $< b >$ with respect to frame $< a >$;
- The translation vector ${}^aO_b \in \mathbb{R}^{3\times1}$: it represent the displacement between the origins of the two frames (this distance is generally given by the lenght of the links of the manipulator between two consecutive joints);
- 0 is a $zero-$vector such as $0 \in \mathbb{R}^{1\times3}$;
- 1 is a scalar value.

In this way, we have a matrix able to compute the rotation and translation in a single matrix operation. The *BuildTree()* function is structured in order to return a $4 \times 4 \times 7$ element, which contains all the transformation matrices along the manipulator's chain. So, the obtained transformation matrices are the following:

This matrix represents a pure translation along the z_b-axis of 0.105m. There is no rotation (so the upper-left 3×3 submatrix R is equal to the identity matrix) and, as we expect, the orientations of frames $< b >$ and $< 1 >$ are coincident.

$$ {}^b_1T = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.105 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

This matrix represent a rotation of 90 degrees clockwise about z_1-axis and a rotation about the new y_2-axis of 90 degrees clockwise; the matrix include a displacement of 0.110m along z_1-axis.

$$ {}^1_2T = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0.11 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

This matrix represent a rotation of 90 degrees counterclockwise about y_2-axis and a rotation about the new z_3-axis of 180 degrees; the matrix include a displacement of 0.100m along x_2 direction.

$$ {}^2_3T = \begin{pmatrix} 0 & 0 & 1 & 0.1 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

This matrix represent a rotation of 90 degrees clockwise about y_3-axis and a rotation about the new x_4-axis of 180 degrees; the matrix include a displacement of 0.325m along z_3 direction.

$$ {}^3_4T = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.325 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

This matrix represent a rotation of 90 degrees clockwise about y_4-axis and a rotation about the new z_5-axis of 90 degrees counterclockwise; the matrix include a displacement of 0.095m along x_4 axis.

$$ {}^4_5T = \begin{pmatrix} 0 & 0 & 1 & 0.095 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

This matrix represent a rotation of 180 degrees about z_5 axis with a displacement of 0.095m along z_5 axis.

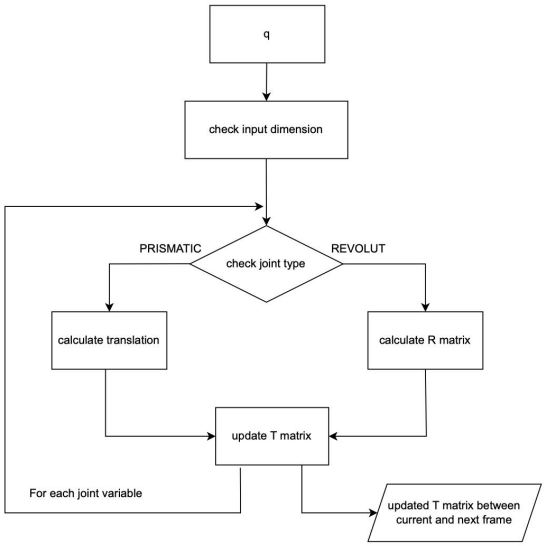
$$ {}^5_6T = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0.095 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

This matrix represents a pure translation along the z_6 direction of 0.355m. In this 4×4 matrix there is no rotation (so the submatrix R is equal to the identity matrix) and, as we expect, the orientations of frames $< 6 >$ and $< e >$ are coincident.

$$ {}^6_eT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.355 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

3 Exercise 1.2 - updateDirectGeometry() method implementation

The exercise involve the implementation of the method *updateDirectGeometry()*, with the aim of computing the model matrices as a function of the joints position vector q. The function takes as input the joints position vector q. In order to avoid errors, first of all we perform a check on the dimension of the input vector that must coincide with the number of joints of the geometric model which we are applying the method on (if the check fails, it returns an error message). If no problem occurs, we update all transformation matrices (one for each couple of consecutive joints of the manipulator), based on the joint type vector (which values are 0 for revolut joints and 1 for prismatic joints), as follows:



- Revolut joint: the rotation matrix has this structure for all revolut joints, since we defined the z-axis coinciding with the joint rotation axis:

$$\mathrm{R} = \begin{pmatrix} \cos(q_i) & -\sin(q_i) & 0 \\ \sin(q_i) & \cos(q_i) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

  The updated matrix is given by:

$$^{i-1}_{i}T(q_i) = ^{i-1}_{i} T(0) \begin{pmatrix} R(q_i) & O_{3x1} \\ 0_{1x3} & 1 \end{pmatrix} \tag{1}$$

- Prismatic joint: the rotation matrix coincides with the identity matrix, so we update only the third component of the translation vector, since we defined the z-axis coinciding with the sliding axis:

$$\mathrm{t} = \begin{pmatrix} 0 \\ 0 \\ q_i \end{pmatrix}$$

  The updated matrix is given by:

$$^{i-1}_{i}T(q_i) = ^{i-1}_{i} T(0) \begin{pmatrix} I_{3x3} & t \\ 0_{1x3} & 1 \end{pmatrix} \tag{2}$$

3.1 Obtained results

Taking into account the following joints position vector:

$$\underline{q} = \begin{pmatrix} \pi/4 & -\pi/4 & 0 & -\pi/4 & 0 & 0.15 & \pi/4 \end{pmatrix}^T$$

we obtain the following trasformation matrices:

$${}^0_1T = \begin{pmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1 & 0.105 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^1_2T = \begin{pmatrix} -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.7071 & 0.7071 & 0 & 0.11 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^2_3T = \begin{pmatrix} 0 & 0 & 1 & 0.1 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^3_4T = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0.325 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^4_5T = \begin{pmatrix} 0 & 0 & 1 & 0.095 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^5_6T = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0.245 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$${}^6_eT = \begin{pmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1 & 0.355 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The resulting transformation matrices are consistent with the properties of rigid transformations, ensuring an accurate representation of the joints position and orientation. Additionally, the calculated final configurations of the manipulator align with the joint positions specified by the input vector.

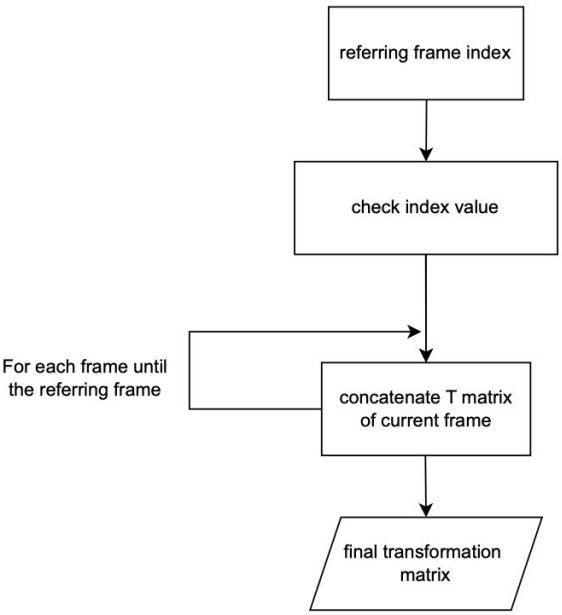
4 Exercise 1.3 - getTransformWrtBase() method implementation

The exercise involve the implementation of the method *getTransformWrtBase()*, with the aim of computing the model transformation matrices from the base to a given referring frame. The function takes as input a number representing the index of the frame which we are interested in compute the transformation matrix. In order to avoid error, first of all we perform a check on the input number, in order to be sure that it is not higher then the total number of joints of the manipulator and that it is a number corresponding to an existing frame (so $k > 1$). Then, to compute the transformation matrix ${}^b_kT$, we multiply all the matrices following the joint order along the chain:

$$ {}^b_nT = {}^b_1T \cdot ... \cdot {}^{n-1}_nT \tag{3} $$

Using this method, we obtain the ${}^b_eT$ matrix, which describe the transformation of frame $< e >$ with respect to frame $< b >$:

$$ {}^b_eT = \begin{pmatrix} -0.5 & -0.5 & -0.7071 & -0.7039 \\ 0.5000 & 0.5000 & -0.7071 & -0.7039 \\ 0.7071 & -0.7071 & 0 & 0.5155 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$



4.1 getTransformWrt() function

Starting from the function *getTransformWrtBase()*, we have implemented the *getTransformWrt()* function, that takes as input the values s and k, where s is the index of the referred frame and k maintain the same meaning of reffering frame. Then we compute the same check on the k value; for the s value we want to ensure that this corresponds to a frame which comes before along the chain and that it is a number corresponding to an existing frame (so $s > 1$). By doing so, the computation of the transformation matrix ${}^s_kT$ is obtained by multiplying all the matrices from s to k index following the joint order in the chain (since the method is built on the same logic of the previous one, the flowchart is quite the same). To compute the transformation matrix ${}^5_3T$, we used the implemented function *getTransformWrt()*, in order to obtain the ${}^3_5T$ transformation matrix. Then, it is sufficient to use the *inv()* matlab function to obtain the inverse matrix. The inverse of a transformation matrix has the following structure:

$$ {}^a_bT = {}^b_a\ T^{-1} = \begin{pmatrix} {}^b_aR^T & -{}^b_aR^T{}^aO_b \\ 0_{3x1} & 1 \end{pmatrix} $$

So, from the computed ${}^3_5T$ matrix, which describe the transformation of frame $< 5 >$ with respect to frame $< 3 >$:

$$ {}^3_5T = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0.7071 & 0 & 0.7071 & 0.0672 \\ -0.7071 & 0 & 0.7071 & 0.3922 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

we obtain the ${}^5_3T$ matrix, which describe the transformation of frame $< 3 >$ with respect to frame $< 5 >$:

$$ {}^5_3T = \begin{pmatrix} 0 & 0.7071 & -0.7071 & 0.2298 \\ -1 & 0 & 0 & 0 \\ 0 & 0.7071 & 0.7071 & -0.3248 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

5 Exercise 1.4 - updateJacobian() method implementation

The Jacobian matrix is a linear operator that allows us to correlate the velocities of the joint variables (angular velocity for revolut joints and linear velocity for prismatic joints) with the generalized velocity of the end effector. In particular, by observing the structure of the Jacobian matrix we can understand how the velocity of a certain joint influences a certain component of the generalized velocity of the end effector. For example, if the element J_{ij} is equal to 0 it means that the velocity of the $j-$th joint does not affect the $i-$th component of the end-effector velocity (having $i \in \{1,2,3\}$ we refer to the angular velocity $\omega_{e/b}$ components; instead having $i \in \{4,5,6\}$ we identify components of the linear velocity $v_{e/b}$).

$$\mathbf{J}(\underline{\mathbf{q}}) = \begin{bmatrix} \frac{\partial \omega_x}{\partial \dot{q}_1} & \frac{\partial \omega_x}{\partial \dot{q}_2} & \dots & \frac{\partial \omega_x}{\partial \dot{q}_n} \\ \frac{\partial \omega_y}{\partial \dot{q}_1} & \frac{\partial \omega_y}{\partial \dot{q}_2} & \dots & \frac{\partial \omega_y}{\partial \dot{q}_n} \\ \frac{\partial \omega_z}{\partial \dot{q}_1} & \frac{\partial \omega_z}{\partial \dot{q}_2} & \dots & \frac{\partial \omega_z}{\partial \dot{q}_n} \\ \frac{\partial v_x}{\partial \dot{q}_1} & \frac{\partial v_x}{\partial \dot{q}_2} & \dots & \frac{\partial v_x}{\partial \dot{q}_n} \\ \frac{\partial v_y}{\partial \dot{q}_1} & \frac{\partial v_y}{\partial \dot{q}_2} & \dots & \frac{\partial v_y}{\partial \dot{q}_n} \\ \frac{\partial v_z}{\partial \dot{q}_1} & \frac{\partial v_z}{\partial \dot{q}_2} & \dots & \frac{\partial v_z}{\partial \dot{q}_n} \end{bmatrix}$$

On the other hand, if the element J_{ij} is different from 0, its value explains how much (and in which direction looking at the sign) the velocity of the j-th joint affects the i-th component of the generalized velocity. The Jacobian matrix is obtained by multiplying two matrices together, the Basic-Robot Jacobian and the Rigid-Body Jacobian. The first composes the Jacobian starting from the base frame up to the n-th frame, the second instead expresses the Jacobian relation between the n-th frame and the other which is positioned on the end effector.

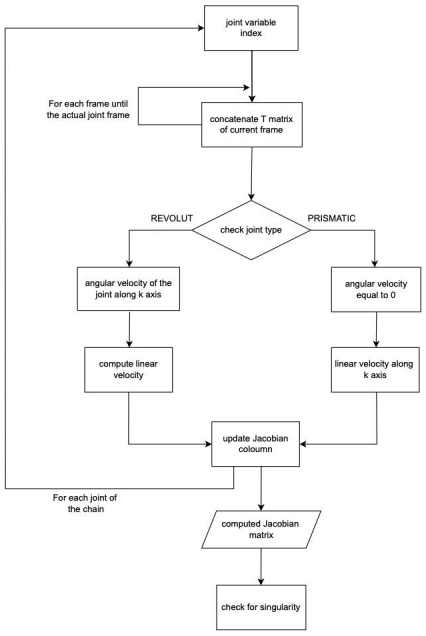
$$\nu_{e/b} = S_{e/n}(\underline{q}) \cdot J_{n/0}(\underline{q}) \cdot \underline{\dot{q}}$$

Where:

- $S_{e/n}(\underline{q})$ is the Rigid-Body Jacobian;
- $J_{n/0}(\underline{q})$ is the Basic Robot Jacobian;
- $\nu_{e/b}$ is the generalized velocity of the end-effector expressed respect to the base frame;
- $\underline{q}$ is the vector containing all the joint variables;
- $\underline{\dot{q}}$ is the vector containing all the joint velocities.

In our case, we did not need the Rigid-Body Jacobian, since without having a tool attached to the n-th frame, we considered in our computations the frame $< e >$ coincident with the frame $< n >$.

The exercise involve the implementation of the method updateJacobian() with the aim of updating the Jacobian matrix with a given $\underline{q}$ joint position vector as input. The function performs a loop taking into account every joint from the first to the n-th. For each joint the transformation matrix is calculated from the base to the current frame, considering the value contained in the $\underline{q}$ input vector. Subsequently, we extract the column relating to the translation between the base frame and the current frame and the unit vector k (around which the rotation occurs for a revolut joint, or along which the translation occurs for a prismatic joint). Then, we proceed with the computation of the contribution that the joint taken into consideration gives to the angular and linear velocity of the final effector, a calculation which is carried out following equations shown below:



$$J^A_{i/i-1} = \begin{cases} k_i & \Gamma_i = r \\ 0 & \Gamma_i = p \end{cases}$$

$$J^l_{n/i} = \begin{cases} k_i \times r_{n/i} & \Gamma_i = r \\ k_i & \Gamma_i = p \end{cases}$$

Considering:

- $J^A_{i/i-1}$ is the angular Jacobian of the i-th frame which respect to the $(i-1)$-th;
- $J^l_{n/i}$ is the linear Jacobian of the n-th frame which respect to the i-th;
- k_i is the unit vector of the i-th joint axis;
- $r_{n/i}$ is the distance vector between the n-th frame end the i-th;
- Γ_i is a vector which contains the value r if the $i-th$ joint is revolut, p if it is prismatic.

After this computation, we can update the coloumn of the Jacobian we are working on, and we can iterate the same process until the last frame.

$$J = \begin{pmatrix} 0 & -0.7071 & -0.5 & -0.7071 & -0.7071 & 0 & -0.7071 \\ 0 & 0.7071 & -0.5 & 0.7071 & -0.7071 & 0 & -0.7071 \\ 1 & 0 & 0.7071 & 0 & 0 & 0 & 0 \\ 0.7039 & 0.2125 & 0.3475 & 0 & 0 & -0.7071 & 0 \\ -0.7039 & 0.2125 & -0.3475 & 0 & 0 & -0.7071 & 0 \\ 0 & 0.9955 & 0 & 0.695 & 0 & 0 & 0 \end{pmatrix}$$

The matrix shown above is the obtained Jacobian for the given $\underline{q}$. As we can see, for a prismatic joint (sixth column of the matrix) the first three rows are null, meaning that it doesn't contributes to the angular velocity of the end effector with respect to the base frame. It's important to check if the manipulator is in a singular configuration (if is in a poor control performance, instability, or complete loss of control in certain directions) by checking the rank of the Jacobian matrix; since J is a 6×7 matrix, we should obtain that:

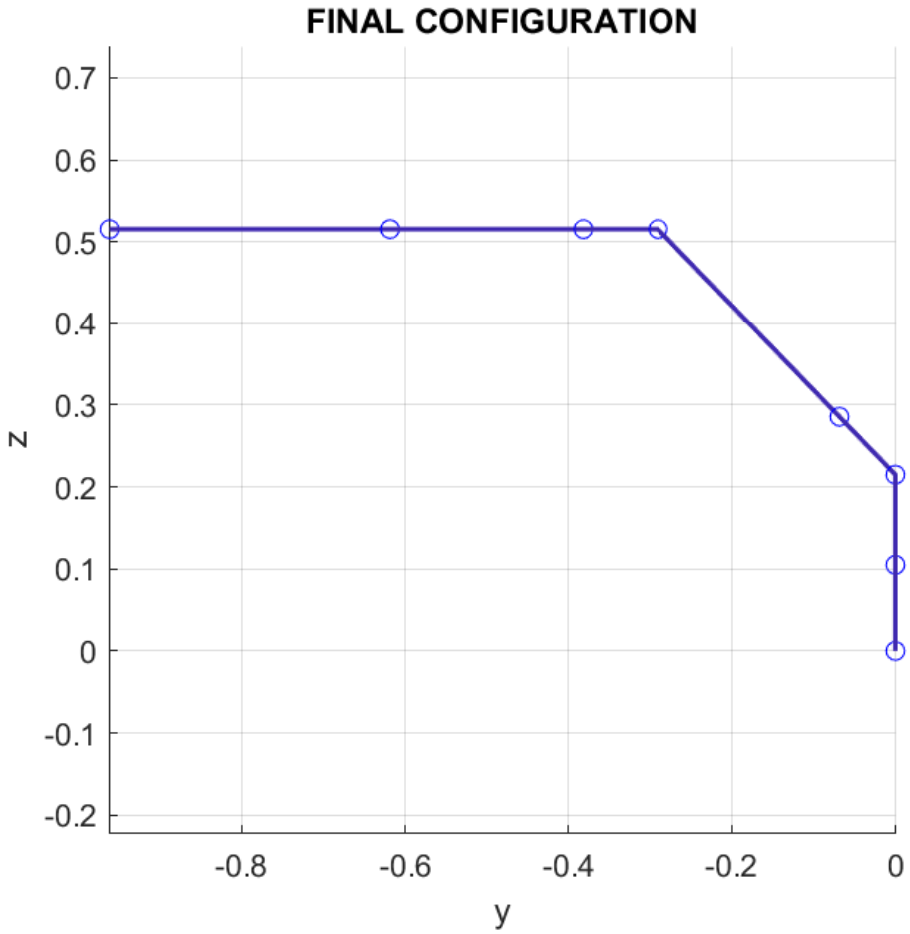
$$rank(J) = min(size(J)) = 6 \tag{4}$$

This means that the manipulator has 6 indipendent directions of motion avalaible to the end-effector in this configuration.

6 Appendix

Appendix A : Final configuration of the manipulator;
Appendix B : Motion of the manipulator applying a rotation about the first joint.

6.1 Appendix A



6.2 Appendix B