Università di Genova

UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF INFORMATICS,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELLING AND CONTROL OF MANIPULATORS

Second Assignment

Manipulator Geometry and Direct Kinematics

Author:

Germena Stefania
Schmidt Irene

Student ID:

s7903986
s4821412

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Simone Borelli
Luca Tarasi

December 17, 2025

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 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 transformation matrices, by filling the structures in the *BuildTree()* function. Be careful to define the z-axis coinciding with the joint rotation axis, such that the positive rotation is the same as showed in the CAD model you received; and the x-axis, when possible, pointing to the next link. 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 computes the model transformation 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 computes the transformation matrix from the base to a given frame. Using this method, compute the following transformation matrices: ${}^{b}_{e}T$, ${}^{6}_{2}T$. Explain the method used and comment on the results obtained.

Q1.4 Run the MATLAB section Q1.4 in the given script *main.m*, which computes the configurations of the manipulators moving from an initial joint position $q_i = [\frac{\pi}{4}, -\frac{\pi}{4}, 0, -\frac{\pi}{4}, 0, 0.15, \frac{\pi}{4}]^\top$ to $q_f = [\frac{5\pi}{12}, -\frac{\pi}{4}, 0, -\frac{\pi}{4}, 0, 0.18, \frac{\pi}{5}]^\top$. The script generates plots using methods from the given class *plotManipulators*, which must not be modified. Check that it behaves reasonably and show the obtained plots in the report.

Q1.5 Implement the method of *kinematicModel* called *getJacobianOfLinkWrtBase()* which takes as argument the link number, and returns the Jacobian of the corresponding link with respect to the base. Compute the Jacobian of **link 6** with respect to base for the final configuration q_f given in Q1.4. Explain the method used and comment on the result obtained.

Q1.6 Implement the method of *kinematicModel* called *updateJacobian()* which updates the Jacobian matrix of the manipulator for the end-effector. Compute the Jacobian for the final configuration q_f given in Q1.4. Explain the method used and comment on the results obtained.

Q1.7 Given the following joint position, $q = [0.7, -0.1, 1, -1, 0, 0.03, 1.3]^\top$ expressed in radians and meters, and the following joint velocities $\dot{q} = [0.9, 0.1, -0.2, 0.3, -0.8, 0.5, 0]^\top$ expressed in rad/s and m/s, compute the velocities of the end effector with respect to the base projected on the end effector frame, ${}^{e}v_{e/b}$ and ${}^{e}\omega_{e/b}$, using **forward kinematics**.

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

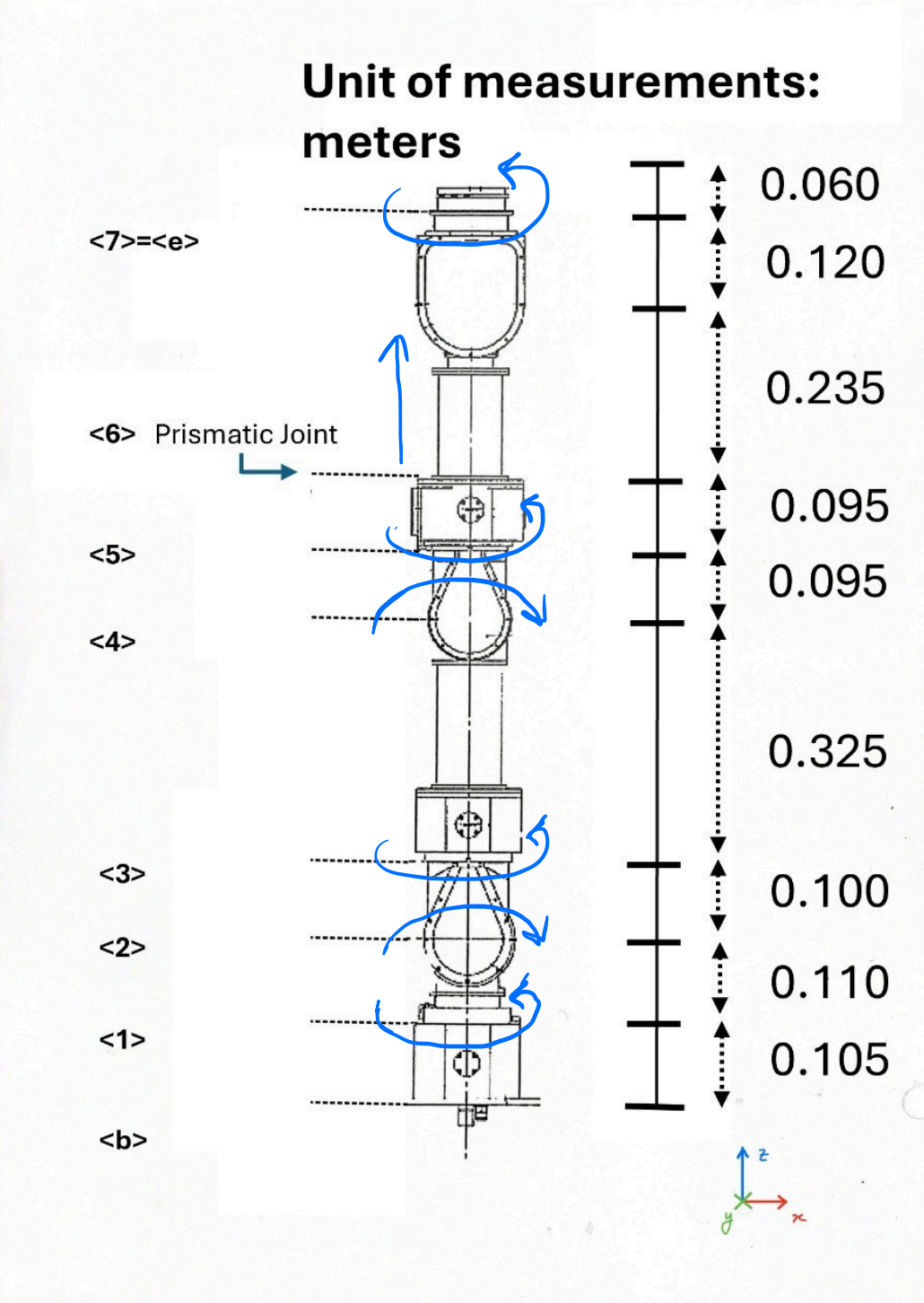

Figure 1: CAD model of the robot. The directions of motion of each joint are indicated by the blue arrows.

2 Exercise 1

Q1.1 The definition of all the transformation matrices has been computed in the *BuildTree()* function. In order to compute the matrices, the reference frames were defined with the z-axis coinciding with the joint rotation axis and the x-axis pointing, when possible, to the following link.

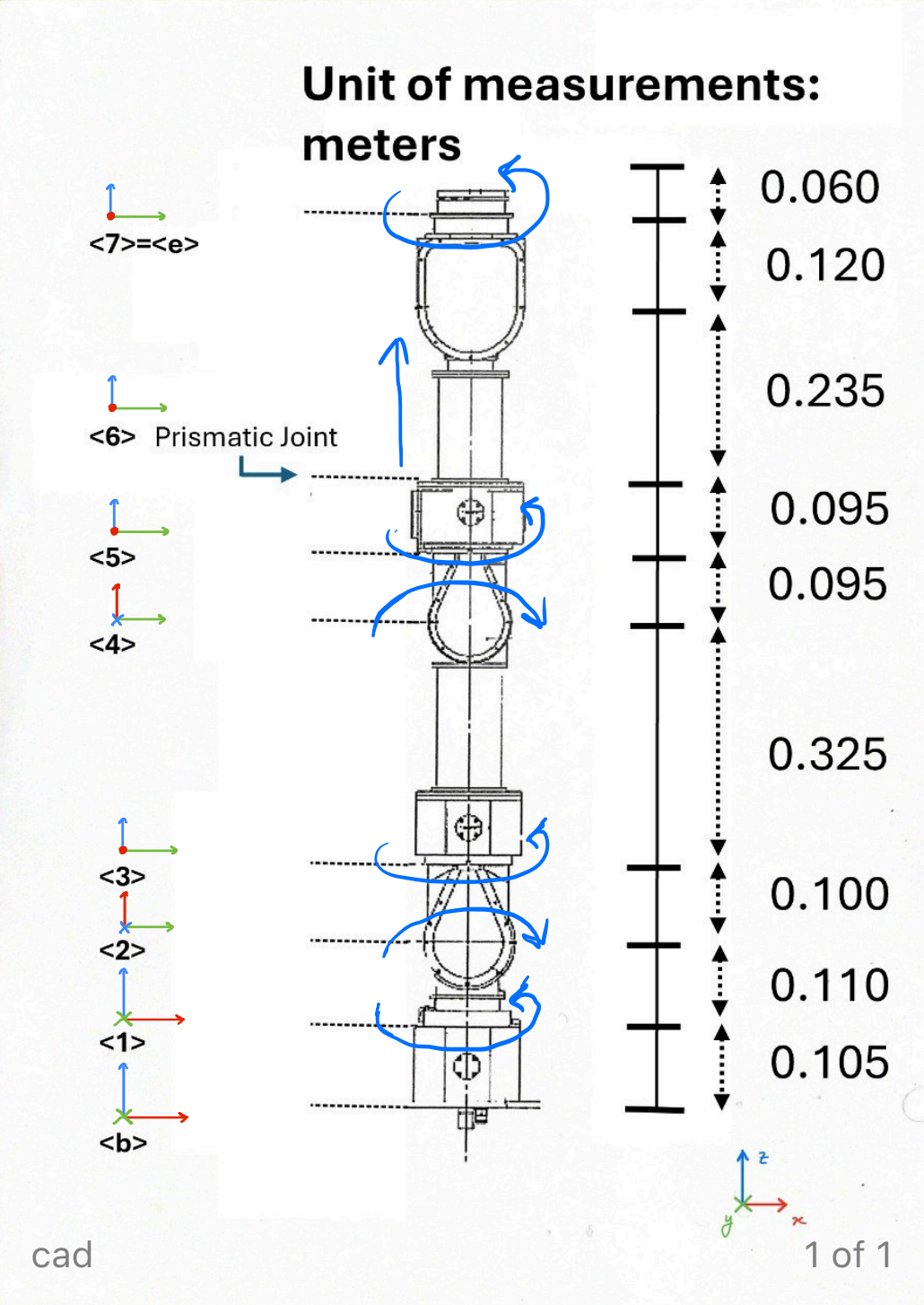

Figure 2: CAD model of the robot. The directions of motion of each joint are indicated by the blue arrows.

Q1.2 The method `updateDirectGeometry(q)` updates the transformation matrices, computing ${}^{i}T_j$ as a function of the joint variables q. The first passage consists of verifying the type of joint. If the joint is revolute, so `jointType(i) == 0`, the transformation consists of a rotation about z of $q(i)$, and the matrix is the following:

$$T(q_i) = \begin{bmatrix} \cos q_i & -\sin q_i & 0 & 0 \\ \sin q_i & \cos q_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

If the joint is prismatic; `jointType(i) == 1`, then it translates over z of $q(i)$, and the relative matrix is:

$$T(q_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_i \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The total transformation between link i and link j is then obtained by combining the constant transformation model ${}^{i}T_{j_0}$, which depends on the geometry of the observed link, with the one depending on the joint type $q(i)$.

$${}^{i}T_j = {}^{i}T_{j_0}\, T(q_i).$$

Q1.3 The method `getTransformWrtBase(k)` computes the transformation matrix that describes the pose of a generic frame k with respect to the base. This is done by multiplying each of the transformation matrices computed in `updateDirectGeometry(q)` up to the generic frame k.

The `getTransformWrtBase(k)` function computes the transformation matrix from the base frame to the frame of the 7th joint. In case of a tool attached to the end effector frame, it would be necessary to have an additional transformation matrix. However, since in this case the end effector frame is indicated as the last frame, we compute the transformation matrix from the base to the end effector frame.

The transformation matrix from the base to the end effector:

$${}^{b}\mathbf{T}_{ee} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1.1850 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

A similar concept is defined for the transformation matrix from the $6th$ joint to the $2nd$ joint. In this instance, the inverse of the transformation matrix from the $2nd$ joint to the $6th$ joint needs to be computed. The `getTransform6wrt2(k)` function is used; however, the starting frame is the second joint, and the final frame is the $6th$ joint.

The MATLAB algorithm computes the following:

$$T_{26} = T_{23}T_{34}T_{45}T_{56}$$

The inverse of this matrix is calculated using the formula:

$$\mathbf{T}^{-1} = \begin{bmatrix} & R^T & & -R^T P_{26} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

From which it follows the transformation matrix:

$${}^{6}\mathbf{T}_2 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -0.6150 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Q1.4

In this exercise, the manipulator is moved from an initial configuration

$$q_i = \begin{bmatrix} \frac{\pi}{4} & -\frac{\pi}{4} & 0 & -\frac{\pi}{4} & 0 & 0.15 & \frac{\pi}{4} \end{bmatrix}^\top$$

to a final configuration

$$q_f = \begin{bmatrix} \frac{5\pi}{12} & -\frac{\pi}{4} & 0 & -\frac{\pi}{4} & 0 & 0.18 & \frac{\pi}{5} \end{bmatrix}^\top.$$

The motion of the manipulator generates a sequence of robot poses that illustrate the transition between the two configurations. With each point seen in Figure 3 and Figure 4 representing the joints. To visualize the motion between the initial and final configurations, joint–space trajectory is computed using the forward kinematics at each step. A total of 100 samples are used to interpolate linearly between the two joint vectors. The generated figures show the evolution of the robot's structure during the motion between q_i and q_f. In order to show that the motion is reasonable, the difference component by component between the current state vector memorized in `geometricModel`, and the vector of the final configuration is shown using

```
disp(geometricModel.q - qf')
```

which being zero demonstrates that the final configuration that is reached, in fact, it corresponds to the given qf.

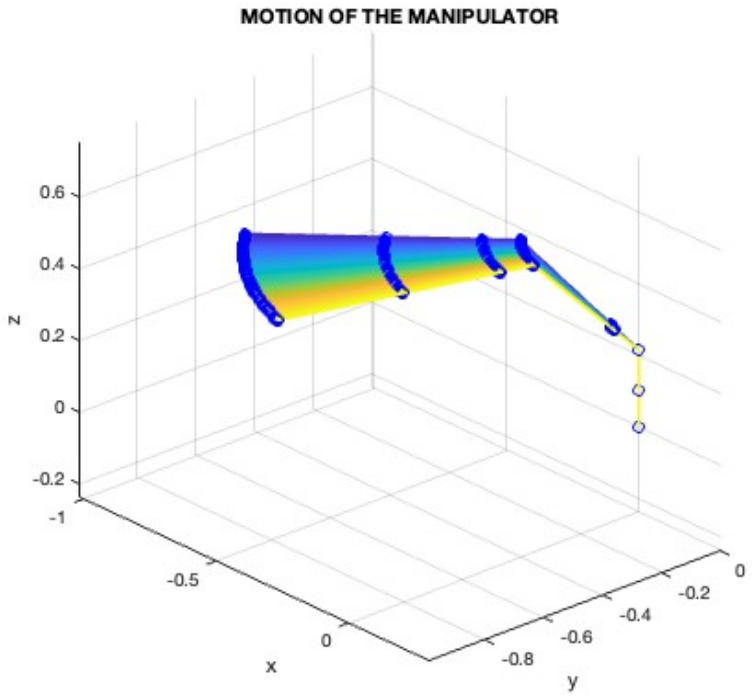

Figure 3: Motion of the manipulator.

This produced the following transformation matrix from the base to the end effector, in the final configuration q_f:

$${}^b\mathbf{T}_{ee} = \begin{bmatrix} 0.5 & -0.5 & -0.7071 & -0.7039 \\ -0.5 & 0.5 & -0.7071 & -0.7039 \\ 0.7071 & 0.7071 & 0 & 0.5155 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

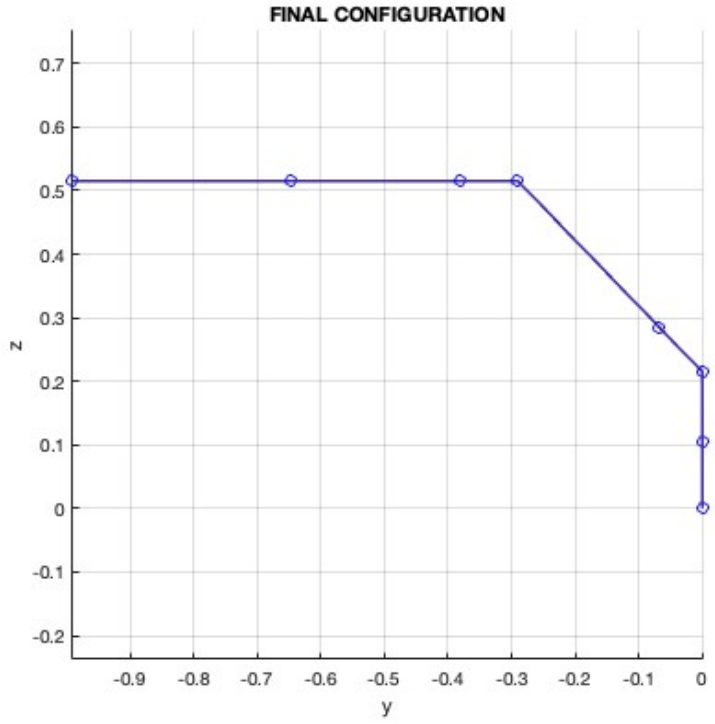

Figure 4: Final configuration.

Q1.5

The method `getJacobianOfLinkWrtBase()` returns the Jacobian of the corresponding link with respect to the base.

The geometric Jacobian describes how the joint velocities contribute to the instantaneous angular and linear velocity of the link i. It is defined as:

$$J_i = \begin{bmatrix} J_i^{\omega} \\ J_i^{L} \end{bmatrix}$$

where J_i^L is the linear velocity component and J_i^{ω} is the angular velocity component.

In order to compute the Jacobian, the transformation matrices of each joint with respect to the base, and the joint types are required. The different joint types alter the type of angular and linear velocity calculated in the Jacobian.

If the joint type is prismatic, there is only a linear velocity contribution, so its angular velocity will be $[000]^T$. While its linear velocity is taken from the translational part of the respective joint transformation matrix.

If the joint type is revolute, it contributes to both linear and angular velocities. Where its angular velocity takes the rotation axis of the joint. This is taken from the rotation matrix of the respective joint transformation matrix with respect to the base reference frame. While its linear velocity is calculated as follows:

$$J_{n/i}^L = [\underline{k}_i X]\underline{r}_{n/i} \tag{1}$$

Where $\underline{r}_{n/i}$ represents the translation from the current joint to the desired frame. This is required to understand how each joint affects the end effector.

The position of the target link is extracted, and the Jacobian is computed based on the joint type and transformation matrix of each joint with respect to the base, as previously discussed. Each column in the Jacobian represents each joint of the manipulator.

The Jacobian of link 6 with respect to the base, evaluated at the final configuration q_f, is given by:

$$\mathbf{J}_6^b = \begin{bmatrix} 0 & -0.9659 & -0.1830 & -0.9659 & -0.2588 & 0 & 0 \\ 0 & 0.2588 & -0.6830 & 0.2588 & -0.9659 & 0 & 0 \\ 1 & 0 & 0.7071 & 0 & 0 & 0 & 0 \\ 0.6477 & 0.0778 & 0.2527 & 0 & 0 & -0.2588 & 0 \\ -0.1735 & 0.2903 & -0.0677 & 0 & 0 & -0.9659 & 0 \\ 0 & 0.6705 & 0 & 0.3700 & 0 & 0 & 0 \end{bmatrix}.$$

Q1.6 In this exercise, the method `updateJacobian()` is implemented within the *kinematicModel* class that computes and updates the Jacobian matrix of the end-effector with respect to the base frame. The Jacobian of the end effector projected in the base frame calculates the angular and linear velocities of each joint in the final configuration q_f. It utilizes the method previously discussed in question 5.

The Jacobian of the last link with respect to the base is computed using the previous method `getJacobianOfLinkWrtB`

The Jacobian obtained is the following:

$$\mathbf{J}_{ee}^{b} = \begin{bmatrix} 0 & -0.9659 & -0.1830 & -0.9659 & -0.2588 & 0 & -0.2588 \\ 0 & 0.2588 & -0.6830 & 0.2588 & -0.9659 & 0 & -0.9659 \\ 1 & 0 & 0.7071 & 0 & 0 & 0 & 0 \\ 0.9906 & 0.0778 & 0.4952 & 0 & 0 & -0.2588 & 0 \\ -0.2654 & 0.2903 & -0.1327 & 0 & 0 & -0.9659 & 0 \\ 0 & 1.0255 & 0 & 0.7250 & 0 & 0 & 0 \end{bmatrix}.$$

Q1.7 Given the joint configuration

$$\mathbf{q} = \begin{bmatrix} 0.7 & -0.1 & 1 & -1 & 0 & 0.03 & 1.3 \end{bmatrix}^{\top},$$

expressed in radians and meters, and the joint velocities

$$\dot{\mathbf{q}} = \begin{bmatrix} 0.9 & 0.1 & -0.2 & 0.3 & -0.8 & 0.5 & 0 \end{bmatrix}^{\top},$$

expressed in rad/s and m/s.

End-effector velocity expressed in the base frame, is the twist representing the spatial velocity of the end-effector with respect to the base frame. The top three entries correspond to the angular velocity $\boldsymbol{\omega}$, while the bottom three entries correspond to the linear velocity $\mathbf{v}$. It is obtained by multiplying the Jacobian $\mathbf{J}_{ee}^{b}$ by the joint velocity vector $\dot{\mathbf{q}}$.

$$\mathbf{V}_{e/b}^{b} = \begin{bmatrix} \boldsymbol{\omega}_{e/b}^{b} \\ \mathbf{v}_{e/b}^{b} \end{bmatrix} [\dot{\mathbf{q}}] = \begin{bmatrix} -0.4008 \\ 0.7457 \\ 0.2820 \\ 0.4432 \\ -0.3112 \\ 0.4208 \end{bmatrix}.$$

In order to compute the velocity of the end effector, projected in the end effector frame, it is necessary to either compute the adjoint or use the rotational matrix (from the transformation matrix).

The first computation requires the transformation matrix representing the end-effector pose with respect to the base frame in the final configuration dictated by q. This transformation matrix from the base to the end effector frame is obtained by composing all joint transformations along the manipulator chain using forward kinematics:

$${}^{b}\mathbf{T}_{e} = \begin{bmatrix} 0.1343 & -0.9885 & 0.0689 & 0.0072 \\ 0.4970 & 0.0071 & -0.8677 & -0.5263 \\ 0.8573 & 0.1508 & 0.4922 & 0.9209 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Then the inverse is computed, which expresses the base frame with respect to the end-effector frame. This matrix is needed to compute the adjoint matrix, allowing the projection of the velocities from the base frame to the end-effector frame:

$${}^{e}\mathbf{T}_{b} = \begin{bmatrix} 0.1343 & 0.4970 & 0.8573 & -0.5289 \\ -0.9885 & 0.0071 & 0.1508 & -0.1281 \\ 0.0689 & -0.8677 & 0.4922 & -0.9105 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The adjoint formula is as follows:

$$Adj(T) = \begin{bmatrix} R & 0 \\ TR & R \end{bmatrix}.$$

Applying the adjoint transformation associated with ${}^{e}\mathbf{T}_{b}$, the end-effector velocity is now projected in the end effector frame:

$$\mathbf{V}^{e}_{e/b} = \left[Adj({}^{e}\mathbf{T}_{b})\right]\left[\mathbf{V}^{b}_{e/b}\right] = \begin{bmatrix} 0.5585 \\ 0.4440 \\ -0.5359 \\ 0.7385 \\ -1.1688 \\ 0.3444 \end{bmatrix}.$$

The same results are obtained when calculating the velocity of the end effector relative to the base, using the rotation matrix to transform the projected frame into the end effector frame. The subsequent formula is employed to project the velocity of the end effector in relation to the base frame in the end effector frame. The rotation matrix is derived from the transformation matrix from the base to the end effector. The inverse of the rotation matrix is required to project the velocity in the end effector frame.

This provides an angular velocity, expressed in the end effector frame as:

$${}^{e}\omega_{ee/0} = {}^{0}R^{T}_{ee}\,{}^{0}\omega_{ee/0} \tag{2}$$

The respective linear velocity is calculated differently due to the linear velocity taking into account the translation. This yields the following formula:

$${}^{e}v_{ee/0} = {}^{0}R^{T}_{ee}([{}^{0}\omega_{ee/0}X]P_{ee/o}) + {}^{0}R^{T}_{ee}\,{}^{0}v_{ee/0} \tag{3}$$

Where $P_{ee/0}$ is taken from the transformation matrix from the base to the end effector.