Università di Genova

UNIVERSITÀ DEGLI STUDI DI GENOVA

DIBRIS

DEPARTMENT OF INFORMATICS,
BIOENGINEERING, ROBOTICS AND SYSTEM ENGINEERING

MODELLING AND CONTROL OF MANIPULATORS

Third Assignment

Jacobian Matrices and Inverse Kinematics

Author:

Germena Stefania
Schmidt Irene

Student ID:

s7903986
s4821412

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Luca Tarasi
Simone Borelli

December 15, 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
${}^{a}O_{b}$	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 Inverse Kinematics (IK) control of a robotic manipulator.

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

- Download the .zip file called from the Aulaweb page of this course.
- Implement the code to solve the exercises on MATLAB by filling in the predefined files.
- Write a report motivating your answers, following the predefined format on this document.
- **Do NOT put your code in the report!**

1.1 Exercise 1

Given the geometric model of an industrial manipulator in Figure 1, you have to add a tool frame. The tool frame is rigidly attached to the robot end-effector according to the following specifications: $\Gamma_{t/e} = [\pi/10, 0, \pi/6], {}^{e}O_t = [0.3, 0.1, 0]^\top (m)$ where $\Gamma_{t/e}$ represents the YPR values from end effector frame to tool frame.

To complete this task you should modify the class *geometricModel* by adding a new method called *getToolTransformWrtBase*.

1.2 Exercise 2

Implement an inverse kinematic control loop to control the tool of the manipulator. You should be able to complete this exercise by using the MATLAB classes implemented for the previous assignment (*geometricModel*,*kinematicModel*), and also you need to implement a new class *cartesianControl* (see the template attached). The initial joint position is $q = [\pi/2, -\pi/4, 0, -\pi/4, 0, 0.15, \pi/4]^\top$, expressed in radians and meters. The procedure can be split into the following phases

Q2.1 Compute the cartesian error between the robot end-effector frame ${}^{b}_{t}T$ and the goal frame ${}^{b}_{g}T$.

The goal frame must be defined knowing that:

- The goal position with respect to the base frame is ${}^{b}O_g = [0.2, -0.7, 0.3]^\top$ (m)
- The goal frame is rotated by the YPR angles $\Gamma_g = [0, 1.57, 0]^\top$ (rad) with respect to the base frame.

Q2.2 Compute the desired angular and linear reference velocities of the tool with respect to the base: ${}^{b}\nu^{*}_{t/b} = \begin{bmatrix} \kappa_a & 0 \\ 0 & \kappa_l \end{bmatrix} \cdot {}^{b}e$, such that $\kappa_a = 0.8$,$\kappa_l = 0.8$ is the gain.

Q2.3 Compute the desired joint velocities $\dot{\tilde{q}}$

Q2.4 Simulate the robot motion by implementing the function: *"KinematicSimulation()"* for integrating the joint velocities in time.

Q2.5 Compute the end effector and tool linear and angular velocities with respect to the base frame projected on the base frame.

Remark 1: All the methods must be implemented for a generic serial manipulator. For instance, joint types, and the number of joints should be parameters.

Remark 2: The class *plotManipulators* is provided for plot generation. You must not modify its implementation, or its calls in *main.m.*

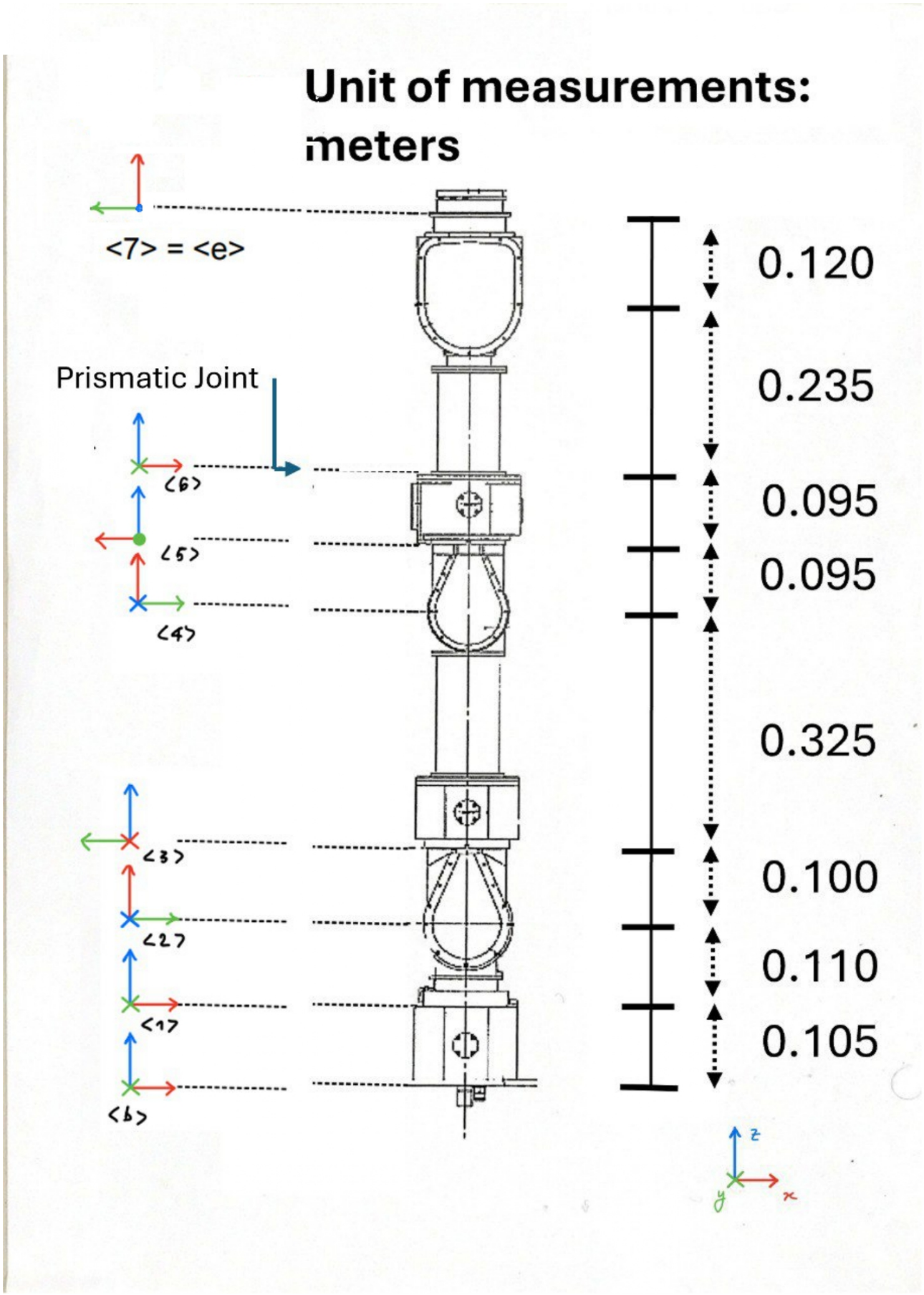


Figure 1: Manipulator CAD

2 Exercise 1

The manipulator geometric model is defined by the sequence of homogeneous transformations connecting consecutive links. The forward kinematics from the base frame $\{b\}$ to the end-effector frame $\{e\}$ is given by:

$$ {}^{b}T_{e} = \prod_{i=1}^{n} {}^{i-1}T_{i}(q_i) \tag{1} $$

In this case a tool frame $\{t\}$ is rigidly attached to the end-effector according to the specifications:

$$ \Gamma_{t/e} = \left[\frac{\pi}{10}, 0, \frac{\pi}{6}\right], \quad {}^{e}\mathbf{O}_{t} = \begin{bmatrix} 0.3 & 0.1 & 0 \end{bmatrix}^{T} \tag{2} $$

The corresponding homogeneous transformation ${}^{e}T_{t}$ is constructed by combining the rotation matrix obtained from the Yaw-Pitch-Roll representation and the translation. The rotation matrix is formulated from the given angles using the function

$$ \mathbf{R}(\psi, \theta, \phi) = \mathbf{R}_{z}(\psi)\mathbf{R}_{y}(\theta)\mathbf{R}_{x}(\phi) \tag{3} $$

The obtained rotation matrix is as follows:

$$ {}^{ee}\mathbf{R}_{t} = \begin{bmatrix} 0.9511 & -0.2676 & 0.1545 \\ 0.3090 & 0.8236 & -0.4755 \\ 0 & 0.5000 & 0.8660 \end{bmatrix} \tag{4} $$

The computation of the homogeneous transformation matrix from the base to the tool is:

$$ {}^{b}T_{t} = {}^{b}T_{e}\ {}^{e}T_{t} \tag{5} $$

$$ {}^{b}\mathbf{T}_{t}(q=0) = \begin{bmatrix} -0.3090 & -0.8236 & 0.4755 & -0.1 \\ 0 & -0.5 & -0.8660 & 0 \\ 0.9511 & -0.2676 & 0.1545 & 1.4850 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

In this way the tool pose can be computed within the forward kinematics framework.

3 Exercise 2

Results were taken after the kinematic simulation was computed, as to obtain the final configuration at the defined goal pose.

3.1 Exercise 2.1

The second exercise consists of implementing an inverse kinematic loop using a Cartesian control loop to move the manipulator tool from an initial configuration as follows:

$$ q_0 = \begin{bmatrix} \pi/2 & -\pi/4 & 0 & -\pi/4 & 0 & 0.15 & \pi/4 \end{bmatrix}^{\top} $$

towards a goal frame defined by:

- Goal position in base frame: $bO_g = [0.2, -0.7, 0.3]^{\top}$ (m)
- Goal orientation (YPR angles): $\Gamma_g = [0, \pi/2, 0]^{\top}$ (rad)

The translation in the transformation matrix includes the goal position. The goal orientation is converted from yaw-pitch-roll angles into a rotational matrix. This rotational matrix is used to create the rotational component of the transformation matrix. The transformation matrix from the base to the goal frame is as follows:

$$ {}^{b}\mathbf{T}_{g} = \begin{bmatrix} 0 & 0 & 1.0 & 0.2 \\ 0 & 1.0 & 0 & -0.7 \\ -1.0 & 0 & 0 & 0.3 \\ 0 & 0 & 0 & 1.0 \end{bmatrix} \tag{7} $$

The transformation matrix from the base to the tool in the initial configuration is as follows:

$$ {}^{b}\mathbf{T}_{t} = \begin{bmatrix} 0 & 0.5000 & 0.8660 & 0 \\ -0.4540 & 0.7716 & -0.4455 & -1.1369 \\ -0.8910 & -0.3932 & 0.2270 & 0.2327 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8} $$

3.2 Exercise 2.2

Cartesian control evaluates the errors in terms of position and orientation between the goal frame and the robot tool frame.

The linear error is computed by subtracting the current tool position in the base frame bO_t, from the goal position bO_g:

$$e_p = bO_g - bO_t \tag{9}$$

The linear error is projected onto the base frame, which is necessary because the Jacobian is defined with respect to the base frame. The rotation error instead is computed by extracting the current tool rotation matrices, which are taken from the most up-to-date configuration, and the goal rotation matrices from their respective homogeneous transformation matrices.

$$^{t}R_g = {}^{b}R_t^{\top\, b}R_g \tag{10}$$

Then, by using the RotToAngleAxis function, we compute $\mathbf{h}$, the unit vector along the rotation axis, and the rotation angle θ. The error rotation axis, $\mathbf{h}$, denotes the preferred axis around which the tool frame should rotate to align with the target frame. The error rotation angle, θ, indicates the degree to which the tool frame must rotate around the error rotation axis to align with the goal frame. These elements aid in rectifying the orientational error.

$$R \xrightarrow{\text{RotToAngleAxis}} (\mathbf{h}, \theta) \tag{11}$$

The combined linear and orientational error, along with the controller gains, produces the desired tool velocities:

$$\omega_{\text{base}} = R_t\, K_a\, e_\omega \tag{12}$$

$$v_{\text{base}} = K_l\, e_p \tag{13}$$

The linear Cartesian error at initial configuration is as follows:

$$\mathbf{e_p} = \begin{bmatrix} 0.2000 \\ 0.4369 \\ 0.0673 \end{bmatrix} \tag{14}$$

The orientation Cartesian error at initial configuration is as follows:

$$\mathbf{e}_\omega = \begin{bmatrix} 0.4603 \\ 0.1233 \\ 0.5138 \end{bmatrix} \tag{15}$$

The following controller gains were used:

$$K_a = \text{diag}(0.8, 0.8, 0.8), \quad K_l = \text{diag}(0.8, 0.8, 0.8)$$

The controller gains assists in the convergence of the system. This represents how the error will decay.

$\omega_{\text{base}}$is the desired angular velocity of the tool in the base frame. In order to project the angular velocity in the base frame, it was necessary to multiply by the rotation matrix ${}^{b}R_t$ (the rotation matrix from the base frame to the tool frame, which was taken from the transformation matrix of the base frame to the tool frame). This ensures that the error of the angular velocity of the tool is projected in the base frame, which is frame that the Jacobian is calculated in. $v_{\text{base}}$is the desired linear velocity of the tool in the base frame.

This results to the Cartesian velocity vector as:

$$\dot{x}_{desired} = \begin{bmatrix} \omega_{\text{base}} \\ v_{\text{base}} \end{bmatrix} \in \mathbb{R}^6 \tag{16}$$

The desired joint velocities produced are what is desired for the tool velocity. The desired joint velocities at the given initial configuration are as follows:

$$\dot{x}_{desired} = \begin{bmatrix} 0.3683 \\ 0.0987 \\ 0.4110 \\ 0.1600 \\ 0.3496 \\ 0.0539 \end{bmatrix} \tag{17}$$

While at the final configuration the desired joint velocities were as follows:

$$\dot{x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.0050 \\ 0.0086 \\ -0.0011 \end{bmatrix} \tag{18}$$

These represent the desired reference velocity of the tool with respect to the base.

3.3 Exercise 2.3

The desired joint velocities using the reference velocities and the pseudo-inverse of the Jacobian, represent how much each motor must spin to achieve the desired joint velocities. This yields the following formula:

$$\dot{q} = J^{\dagger}\dot{x} \tag{19}$$

The required Jacobian is the Jacobian of the tool with respect to the base. In order to compute this Jacobian, these are required: 1. The position of the tool with respect to the end effector. This is taken from the given translation in exercise. In order to project it in the base frame, it is essential to use the rotation matrix that represents the rotation from the base to the end effector frame. The choice of this rotation matrix is because the translation of the tool with respect to the end effector is in the end effector frame.

2. Now to perform matrix multiplication. This requires the translation from the end-effector to the tool represented in the base frame. Once the translation is calculated and represented in the base frame which requires the rotation matrix from the base to the end-effector frame. We can now compute the skew-symmetric matrix of this vector. We can now compute a matrix that can be multiplied by the Jacobian of the end effector with respect to the base.

This yields the following formula:

$$J_{\text{tool/base}} = \begin{bmatrix} I & 0) \\ [r_{t/e}X]\top & I \end{bmatrix} J_{\text{ee/base}} \tag{20}$$

In order to compute the inverse of the Jacobian, the pseudo-inverse is required. The pseudo-inverse is solved using SVD decomposition. The SVD decomposition ensures that the joint velocity vector found has the minimal error between the desired Cartesian velocity and the actual joint space velocity. An essential part of solving with SVD is for the cases of singularity. A singularity occurs when at an instantaneous configuration of a robot, it loses a degree of freedom. If the robot approaches a singularity the SVD decomposition allows us to set the corresponding joint effect to zero. This can guaranty that there is no loss of control of the robot. The damping SVD was used to solve for the pseudo-inverse of the Jacobian, as it ensures that if a singularity occurs, it does not just set the joint effect to zero, but rather allows the joint to slow down gracefully. This prevents a jerking motion compared to truncated SVD.

The desired motor velocities at the initial configuration were as follows:

$$\dot{q} = \begin{bmatrix} -0.5187 \\ 0.1554 \\ 1.3148 \\ -0.0701 \\ -1.0284 \\ -0.1987 \\ 0.4535 \end{bmatrix} \tag{21}$$

3.4 Exercise 2.4

In order to compute the simulation of the robot reaching the goal frame. It is necessary to continuously feed the updated joint positions, which are calculated using the KinematicSimulation function, into the updateDirect-Geometry function.

It is also necessary to update the Jacobian and the cartesianControl functions, and to recalculate the Jacobian of the tool with respect to the base as well as the pseudo-inverse of the Jacobian.

The reason for this is because the robot is moving so the calculations need to be continuously updated to correspond to its current motion.

If the goal frame is out of the dexterous workspace the robot will never reach the requested pose. It is essential to ensure that the goal frame is within the workspace to avoid non convergence.

The simulation shown represents the trajectory of the robot from its initial configuration to the goal frame in 20 seconds. The initial position is defined by the following joint positions:

$$q_{initial} = [\pi/2, -\pi/4, 0, -\pi/4, 0, 0.15, \pi/4] \tag{22}$$

Once the requested pose was reached the desired joint velocities at the final configuration was as follows:

$$\dot{\underline{x}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.0050 \\ 0.0086 \\ -0.0011 \end{bmatrix} \tag{23}$$

The desired motor velocities at the final configuration revealed the following:

$$\dot{\underline{q}} = \begin{bmatrix} -0.0009 \\ 0.0007 \\ 0.0004 \\ -0.0011 \\ 0.0004 \\ -0.0097 \\ -0.0004 \end{bmatrix} \tag{24}$$

This represented the expected motor velocities to reach the desired joint velocity.

3.5 Exercise 2.5

The actual velocity is the Jacobian multiplied by the desired velocity, acquired from the equation 24. The equation used is as follows:

$$\dot{q}_{actual} = J\dot{q}_{desired} \tag{25}$$

The actual velocity for the tool was calculated using the Jacobian of the tool with respect to the base. The actual velocity of the end effector was calculated using the Jacobian of the end effector with respect to the base.

The actual velocity of the tool with respect to the base is as follows:

$$\dot{\underline{q}}_{tool} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.0050 \\ 0.0086 \\ -0.0011 \end{bmatrix} \tag{26}$$

The tool's actual velocity relative to the base corresponds with the desired velocity. The data further reveals that a singularity was not attained, and the robot remained within the dexterous workspace. If the robot's end position was in a singularity, the actual velocity would have been lower than the desired velocity. The computation of SVD aims to approximate the desired configuration; however, if the final configuration is singular, SVD calculations will hinder the computation of unbounded joint velocities to maintain stability and prevent surpassing the mechanical limits of the actuators.

To show that the tool reached the goal frame it is possible to multiply the inverse transformation matrix from the base to the tool by the transformation matrix from the base to the goal frame. This resulted in the following transformation matrix:

$$\mathbf{T}_{\text{error}} = \left({}^{b}T_{t}\right)^{T} {}^{b}T_{\text{g}} = \begin{bmatrix} 1 & 0 & 0 & 0.0014 \\ 0 & 1 & 0 & 0.0108 \\ 0 & 0 & 1 & -0.0062 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{27}$$

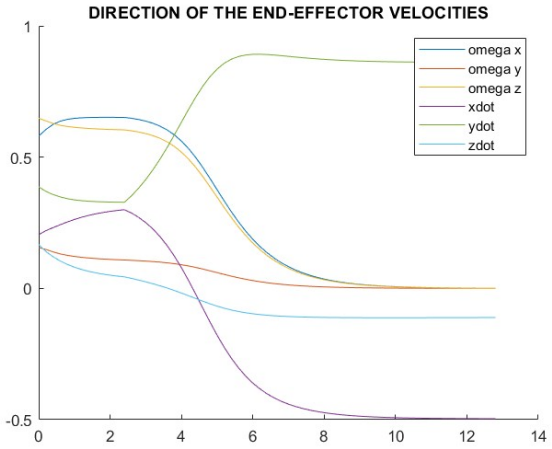
This transformation matrix closely resembles an identity matrix; the off-diagonal values deviate by 10^{-3} and 10^{-2}, indicating a minimal positional error. The intended and actual velocities were the same, confirming that the robot performs the kinematic command, while the calculated error matrix indicates the proximity of the tool frame to the goal frame.

A minor discrepancy is anticipated due to the mathematical computation of the damped SVD, which gradually reduces the robot's motion when a singular value falls below 0.001. The gain values of the controller influence the achievement of the precise position, as they dictate the speed at which the desired position is achieved. As the error becomes smaller, the desired velocity decreases, and the damped SVD generates a minimal value that gradually drives the robot towards the intended position, although never precisely achieving it. Consequently, both the damped SVD and the controller gain contribute to this acceptable positional error.

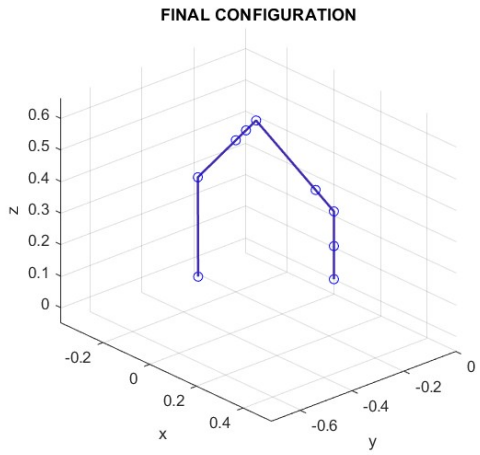
The actual velocity of the end effector with respect to the base is as follows:

$$\underline{\dot{q}}_{ee} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -0.0050 \\ 0.0086 \\ -0.0011 \end{bmatrix} \tag{28}$$

The "DIRECTION OF THE END-EFFECTOR VELOCITIES" figure represents how the angular and linear components of the desired end effector velocity vector were altered during the robot's trajectory to the goal frame.

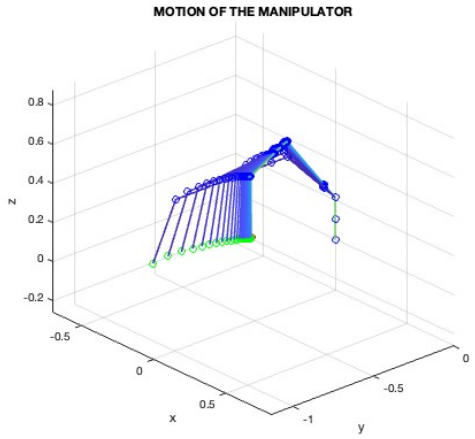


The final configuration of the robot is as follows:



This represents the manipulator's final configuration in the dextrous workspace. Here it is clearly seen where all respective joints and the tool frame are located within the dextrous workspace.

The following represents the trajectory of the joints starting at the given initial configuration and ending at the desired goal frame:



The green dot represents the trajectory of the tool throughout the motion. It is also noticeable that the first joint had no effect during the entire trajectory of the motion.