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:

Ashoori Sarvenaz
Hayee Hafiz Abdul
Khadka Chhetri Rubin

Student ID:

s6878764
s6029926
s6558048

Group:

4

Professors:

Enrico Simetti
Giorgio Cannata

Tutors:

Andrea Tiranti
Luca Tarasi
George Kurshakov

December 10, 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.

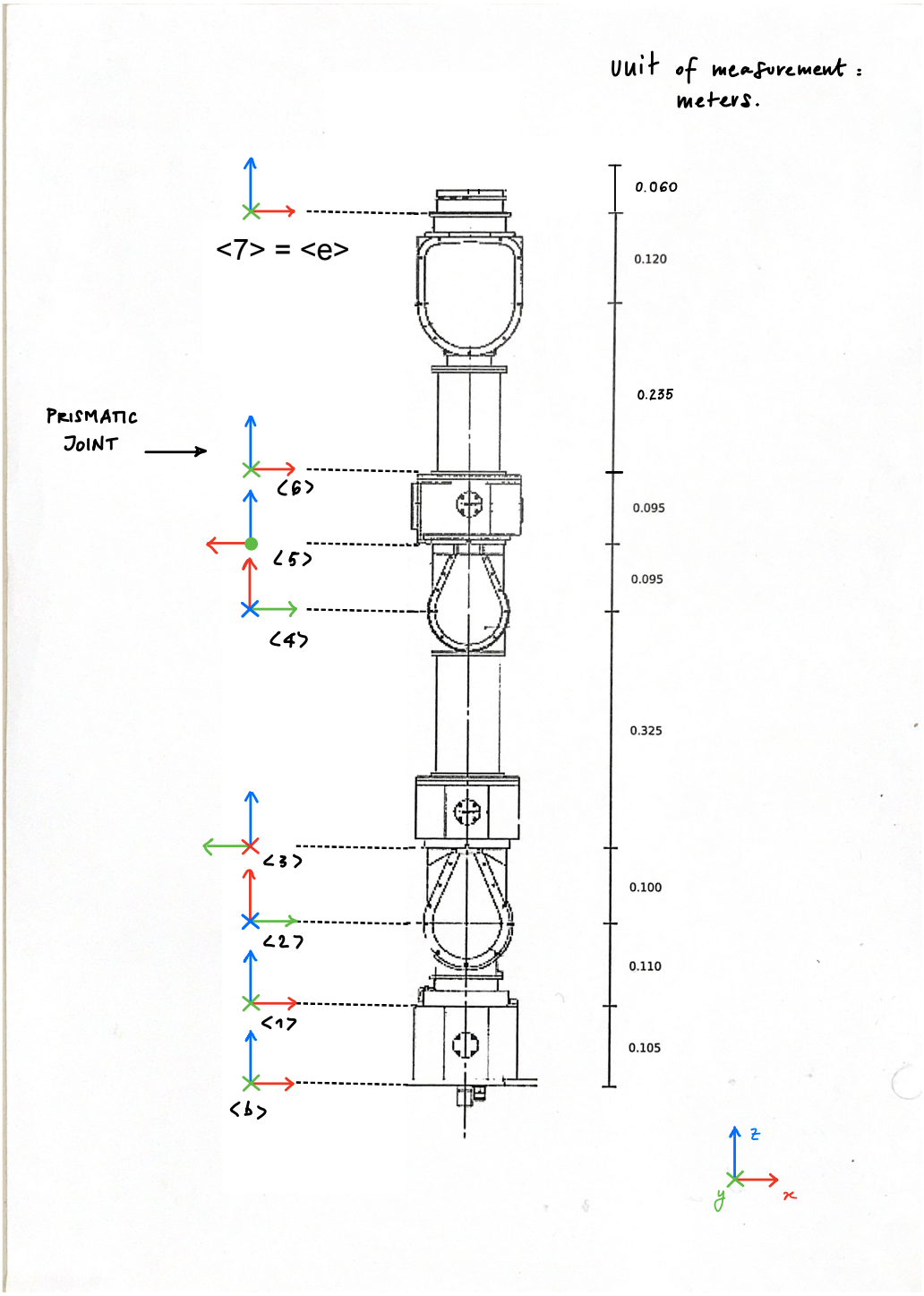

Figure 1: CAD model of the robot

2 Exercise 1

This assignment focuses on the geometric modeling and direct kinematics of a 7-DOF manipulator. The main objectives are:

- Defining the transformation matrices for the manipulator in its zero configuration.
- Computing the transformation matrices as a function of joint positions.
- Calculating transformations from the base to specific joint frames to analyze the spatial relationships between different parts of the manipulator.
- Deriving the Jacobian matrix for the end-effector.

2.1 Q 1.1: Transformation Matrices in Zero Configuration

To analyze the geometry of the manipulator in its zero configuration, transformation matrices between consecutive frames are derived from the provided CAD model. The zero configuration refers to the default pose of the manipulator, with all joint variables q set to zero. These matrices define the spatial relationships between adjacent links and are essential for kinematic analysis.

In MATLAB, the function `BuildTree()` is used to define these transformation matrices between consecutive frames in the zero configuration. However, the transformation matrices must be manually filled in the function before it can be used for other kinematic calculations.

2.1.1 Methodology for Deriving Transformation Matrices

The transformation matrices ${}^{i}_{j}T_0$ represent the transformation of frame j relative to frame i in zero configuration, incorporating both rotation and translation. Note that frame j corresponds to the next frame relative to frame i (i.e., frame j is the frame $i+1$). The derivation follows these steps:

1. **Frame Orientation and Offsets:** The provided CAD model of the 7-DOF robot, shown in Figure 1, specifies the orientations and distances (offsets) between consecutive frames. This information is used to calculate the relative rotation matrices and position vectors between the frames.
2. **Rotation Matrices:** The rotation matrix ${}^{i}_{j}R$ defines the relative orientation between frames i and j, describing how the coordinate axes of frame j are oriented relative to frame i.
3. **Position Vectors:** The position vector ${}^{i}_{j}r$ represents the displacement between frames along the x, y, or z-axes. These vectors are derived from the distances specified in the CAD model.
4. **Homogeneous Transformation Matrices:** The transformation matrix ${}^{i}_{j}T_0$ is a 4x4 matrix combining rotation and translation:

$$ {}^{i}_{j}T_0 = \begin{bmatrix} {}^{i}_{j}R & {}^{i}_{j}r \\ 0 & 1 \end{bmatrix} $$

   where:

   - ${}^{i}_{j}R$ is the 3x3 rotation matrix,
   - ${}^{i}_{j}r$ is the 3x1 position vector,
   - The last row $[0\ 0\ 0\ 1]$ ensures that the matrix is homogeneous.

2.1.2 Initial Transformation Matrices

The transformation matrices ${}^{i}_{j}T_0$ for each joint in the zero configuration are as follows:

Transformation of Frame 1 Relative to Frame b

$$ {}^{b}_{1}T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.105 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 1 is aligned with the x-axis of the base frame.

- The y-axis of frame 1 is aligned with the y-axis of the base frame.
- The z-axis of frame 1 is aligned with the z-axis of the base frame.
- The translation is along the z-axis by 0.105 units.

Transformation of Frame 2 Relative to Frame 1

$$ {}^{1}_{2}T_{0} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0.110 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 2 is aligned with the positive z-axis of frame 1.
- The y-axis of frame 2 is aligned with the positive x-axis of frame 1.
- The z-axis of frame 2 is aligned with the positive y-axis of frame 1.
- The translation is along the z-axis by 0.110 units.

Transformation of Frame 3 Relative to Frame 2

$$ {}^{2}_{3}T_{0} = \begin{bmatrix} 0 & 0 & 1 & 0.100 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 3 is aligned with the positive z-axis of frame 2.
- The y-axis of frame 3 is aligned with the negative y-axis of frame 2.
- The z-axis of frame 3 is aligned with the positive x-axis of frame 2.
- The translation is along the x-axis by 0.100 units.

Transformation of Frame 4 Relative to Frame 3

$$ {}^{3}_{4}T_{0} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0.325 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 4 is aligned with the positive z-axis of frame 3.
- The y-axis of frame 4 is aligned with the negative y-axis of frame 3.
- The z-axis of frame 4 is aligned with the positive x-axis of frame 3.
- The translation is along the z-axis by 0.325 units.

Transformation of Frame 5 Relative to Frame 4

$$ {}^{4}_{5}T_{0} = \begin{bmatrix} 0 & 0 & 1 & 0.095 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 5 is aligned with the positive z-axis of frame 4.
- The y-axis of frame 5 is aligned with the negative x-axis of frame 4.
- The z-axis of frame 5 is aligned with the negative y-axis of frame 4.
- The translation is along the x-axis by 0.095 units.

Transformation of Frame 6 Relative to Frame 5

$$ {}^{5}_{6}T_0 = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0.095 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 6 is aligned with the negative x-axis of frame 5.
- The y-axis of frame 6 is aligned with the negative y-axis of frame 5.
- The z-axis of frame 6 is aligned with the positive z-axis of frame 5.
- The translation is along the z-axis by 0.095 units.

Transformation of Frame 7 Relative to Frame 6

$$ {}^{6}_{7}T_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.355 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

- The x-axis of frame 7 is aligned with the x-axis of frame 6.
- The y-axis of frame 7 is aligned with the y-axis of frame 6.
- The z-axis of frame 7 is aligned with the z-axis of frame 6.
- The translation is along the z-axis by 0.355 units.

These transformation matrices describe the relative positions and orientations between adjacent frames in the 7-DOF manipulator. They form the foundation for subsequent tasks such as forward kinematics and Jacobian computation, which are essential for analyzing the manipulator's motion capabilities.

2.2 Q 1.2: Transformation Matrices Based on Joint Positions

After obtaining the transformation matrices in the zero configuration, the next step is to update these matrices as a function of the joint positions, q. This is essential for solving the forward kinematics problem, as it enables the computation of the position and orientation of the end-effector based on the current joint angles or displacements.

To update the transformation matrices, we adjust each matrix for every joint as it moves from its zero configuration ($q = 0$) to its current configuration, defined by q. The transformation for each joint depends on its type (rotational or prismatic) and its corresponding joint position.

2.2.1 Methodology for Updating Transformation Matrices

The objective is to compute the transformation matrix ${}^{i}_{j}T(q)$ for each link in the manipulator, based on the current joint configuration q. This involves updating the rotation matrix and position vector from their initial configuration ${}^{i}_{j}T_0$ by incorporating the effect of the joint displacements. The process varies depending on whether the joint is rotational or prismatic, which dictates how the transformation matrix is modified. The procedure is as follows:

1. **Extract Rotation Matrix and Position Vector:** The first step is to extract the rotation matrix ${}^{i}_{j}R_0$ and position vector ${}^{i}_{j}r_0$ from the initial transformation matrix ${}^{i}_{j}T_0$. These components define the relative orientation and displacement between the frames, respectively. For details on the form of ${}^{i}_{j}T_0$, please refer to Section 2.1.1.
2. **Update for Rotational Joints (Type 0):** When the current joint q is a rotational joint, the displacement is defined by an angular displacement q, which corresponds to a rotation around the z-axis. The process to update the rotation matrix and position vector is as follows:

- The rotation matrix ${}^{i}_{j}R_0$, extracted from the zero configuration transformation matrix ${}^{i}_{j}T_0$, is updated by multiplying it with the rotation matrix $R_z(q)$, which represents a rotation by an angle q around the z-axis:

$$ {}^{i}_{j}R(q) =^{i}_{j} R_0 \cdot R_z(q) \tag{1}$$

where, $R_z(q)$ is given by:

$$R_z(q) = \begin{bmatrix} \cos(q) & -\sin(q) & 0 \\ \sin(q) & \cos(q) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This updates the orientation of the joint based on the angular displacement q.

- The position vector ${}^{i}_{j}r_0$ remains unchanged, as there is no translation for a rotational joint.

$$ {}^{i}_{j}r(q) =^{i}_{j} r_0 \tag{2}$$

Thus, the updated transformation for the rotational joint becomes:

$${}^{i}_{j}T(q) = \begin{bmatrix} {}^{i}_{j}R(q) & {}^{i}_{j}r(q) \\ 0 & 1 \end{bmatrix}$$

where:

- ${}^{i}_{j}R(q)$ is the updated rotation matrix
- ${}^{i}_{j}r(q) =^{i}_{j} r_0$ is the unchanged position vector.

3. **Update for Prismatic Joints (Type 1):** When the current joint q is a prismatic joint, the displacement is defined by a linear displacement q, which corresponds to translation along a specific axis (typically the z-axis). The process to update the rotation and position vectors is as follows:

- The rotation matrix ${}^{i}_{j}R_0$, extracted from the zero configuration transformation matrix ${}^{i}_{j}T_0$, remains unchanged, as there is no rotation for a prismatic joint:

$$ {}^{i}_{j}R(q) =^{i}_{j} R_0 \tag{3}$$

- The position vector ${}^{i}_{j}r_0$ is updated to account for the translation along the Z-axis. The new position vector is given by:

$$ {}^{i}_{j}r(q) =^{i}_{j} r_0 + q \cdot k \tag{4}$$

where:

  - k is the unit vector along the Z-axis
  - q is the linear displacement of the prismatic joint.

Thus, the updated transformation for the prismatic joint becomes:

$${}^{i}_{j}T(q) = \begin{bmatrix} {}^{i}_{j}R(q) & {}^{i}_{j}r(q) \\ 0 & 1 \end{bmatrix}$$

where:

- ${}^{i}_{j}R(q) =^{i}_{j} R_0$ is the unchanged rotation matrix
- ${}^{i}_{j}r(q)$ is the updated position vector.

2.2.2 Updated Transformation Matrices

The program is flexible and not hard-coded, allowing the input values like joint types and joint positions, to be easily adjusted or changed depending on the specific manipulator type. For this assignment, the following input values are provided:

- **Joint Type:** The joint type is specified as:

$$\text{jointType} = [0\ 0\ 0\ 0\ 0\ 1\ 0]$$

In this array, 0 represents a **rotational** joint, and 1 indicates a **prismatic** joint.

- **Joint Positions (q):** The joint positions are given by:

$$q = \left[\frac{\pi}{4}, -\frac{\pi}{4}, 0, -\frac{\pi}{4}, 0, 0.15, \frac{\pi}{4}\right]$$

  These values correspond to the angles of the rotational joints and the displacement for the prismatic joint.

- **Initial Transformation Matrices:** The initial transformation matrices for each joint are provided in Section 2.1.2.

Using the provided inputs, the updated transformation matrices for each joint are computed through the `updateDirectGeometry` function in MATLAB. The resulting updated matrices for each joint are as follows:

Transformation of Frame 1 Relative to Frame b with Joint Angle $q = \frac{\pi}{4}$

$${}^{b}_{1}T(q) = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1.0000 & 0.1050 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The x- and y-axes of Frame 1 are rotated by $45°$ relative to Frame b, while the z-axis remains unchanged. Additionally, Frame 1 is translated along the z-axis by 0.1050 units relative to Frame b.

Transformation of Frame 2 Relative to Frame 1 with Joint Angle $q = -\frac{\pi}{4}$

$${}^{1}_{2}T(q) = \begin{bmatrix} -0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 \\ 0.7071 & 0.7071 & 0 & 0.1100 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The x- and y-axes of Frame 2 are rotated by $-45°$ relative to the x- and y-axes of Frame 1, while the z-axis remains unchanged. Additionally, Frame 2 is translated along the z-axis by 0.1100 units relative to Frame 1.

Transformation of Frame 3 Relative to Frame 2 with Joint Angle $q = 0$

$${}^{2}_{3}T(q) = \begin{bmatrix} 0 & 0 & 1.0000 & 0.1000 \\ 0 & -1.0000 & 0 & 0 \\ 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The x- and y-axes of Frame 3 are rotated by 90° relative to Frame 2, with the z-axis of Frame 3 aligned along the x-axis of Frame 2. Additionally, Frame 3 is translated along the x-axis by 0.1000 units relative to Frame 2.

Transformation of Frame 4 Relative to Frame 3 with Joint Angle $q = -\frac{\pi}{4}$

$${}^{3}_{4}T(q) = \begin{bmatrix} 0 & 0 & 1.0000 & 0 \\ 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0.3250 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The x- and y-axes of Frame 4 are rotated by $45°$ relative to Frame 3 in the xy-plane, while the z-axis remains unchanged. Additionally, Frame 4 is translated along the z-axis by 0.3250 units relative to Frame 3.

Transformation of Frame 5 Relative to Frame 4 with Joint Angle $q = 0$

$${}^{4}_{5}T(q) = \begin{bmatrix} 0 & 0 & 1.0000 & 0.0950 \\ -1.0000 & 0 & 0 & 0 \\ 0 & -1.0000 & 0 & 0 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The x- and z-axes of Frame 5 are rotated by $90°$ relative to Frame 4, with the z-axis of Frame 5 aligned with the x-axis of Frame 4. Additionally, Frame 5 is translated along the x-axis by 0.0950 units relative to Frame 4.

Transformation of Frame 6 Relative to Frame 5 with Joint Displacement $q = 0.015$

$$ {}^{5}_{6}T(q) = \begin{bmatrix} -1.0000 & 0 & 0 & 0 \\ 0 & -1.0000 & 0 & 0 \\ 0 & 0 & 1.0000 & 0.2450 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} $$

The x- and y-axes of Frame 6 are rotated by $180°$ relative to Frame 5 around the z-axis, while the z-axis remains unchanged. Additionally, Frame 6 is translated along the z-axis by 0.2450 units relative to Frame 5.

Transformation of Frame 7 Relative to Frame 6 with Joint Angle $q = \frac{\pi}{4}$

$$ {}^{6}_{7}T(q) = \begin{bmatrix} 0.7071 & -0.7071 & 0 & 0 \\ 0.7071 & 0.7071 & 0 & 0 \\ 0 & 0 & 1.0000 & 0.3550 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} $$

The x- and y-axes of Frame 7 are rotated by $45°$ relative to Frame 6 in the xy-plane, while the z-axis remains unchanged. Additionally, Frame 7 is translated along the z-axis by 0.3550 units relative to Frame 6.

2.2.3 MATLAB implementation

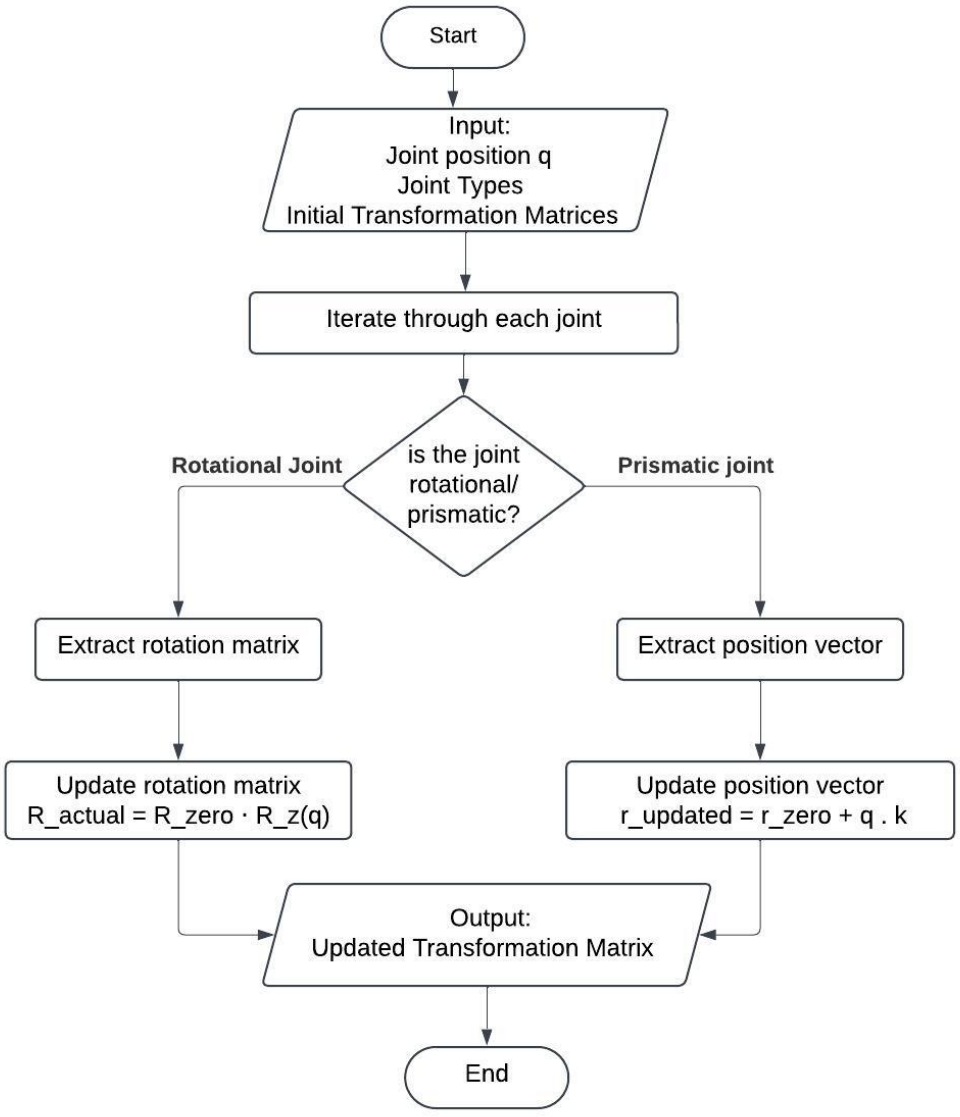

Figure 2: MATLAB implementation of update geometry function

2.3 Q 1.3: Transformation Matrices from Base to a Given Joint Frame

Building on the updated transformation matrices computed in Section 2.2.2, the next step is to calculate the transformation matrices between specific frames. Determining the transformation of a desired frame with respect to the base frame provides essential insights into the spatial relationship and positioning of that frame in the manipulator's workspace. Additionally, computing the transformation between two arbitrary frames helps in understanding their relative positioning and orientation.

For this task, the following transformations are calculated:

- ${}^{b}_{e}T$: Transformation of the end-effector frame e with respect to the base frame b.
- ${}^{5}_{3}T$: Transformation of frame 3 with respect to frame 5.

To achieve these transformations, two key functions are implemented. These functions are designed to compute specific types of transformations, and their details are explained in the following subsections.

2.3.1 Base-to-Frame Transformation

The transformation matrix of the k-th frame with respect to the base frame ${}^{b}_{k}T$ provides a comprehensive representation of the frame's position and orientation in the manipulator's workspace. To compute this, the individual transformation matrices ${}^{i}_{j}T$ are sequentially multiplied along the path from the base frame to the desired frame. This process is mathematically expressed as:

$$ {}^{b}_{k}T = {}^{b}_{1}T \cdot {}^{1}_{2}T \cdot \ldots \cdot {}^{k-1}_{k}T. \tag{5} $$

The result of this multiplication is the cumulative transformation matrix that describes how the frame k is positioned and oriented relative to the base frame.

- **Matlab Implementation:** The calculation of ${}^{b}_{k}T$ is implemented in the `getTransformWrtBase()` function. This function iterates through the frames from the base to the desired frame, multiplying the corresponding transformation matrices at each step.

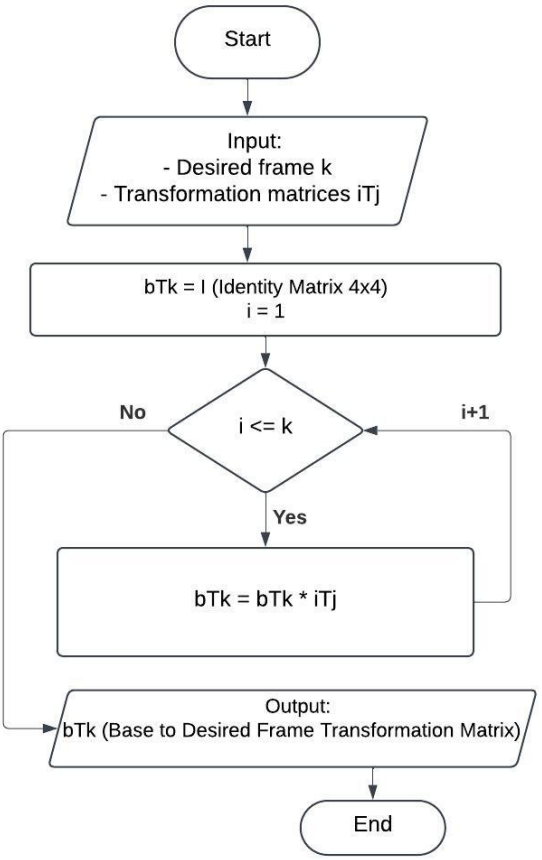

Figure 3: Matlab Implementation of Base to Frame Transformation

- **Result:** The transformation from the base frame to frame e (the end-effector frame) is computed as:

$$ {}^{b}_{e}T = \begin{bmatrix} -0.5000 & -0.5000 & -0.7071 & -0.7039 \\ 0.5000 & 0.5000 & -0.7071 & -0.7039 \\ 0.7071 & -0.7071 & 0.0000 & 0.5155 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} $$

The end-effector frame is rotated relative to the base frame by 45° in the xy-plane, with a tilt of the z-axis into the xy-plane. Additionally, the end-effector is translated by -0.7039 units along both the x- and y-axes, and 0.5155 units along the z-axis.

2.3.2 Frame-to-Frame Transformation

The transformation matrix ${}^{5}_{3}T$ represents the spatial relationship between frame 3 and frame 5. It describes the position and orientation of frame 3 relative to frame 5. This is extra exercise designed for understanding the frames relationship. Hence, this part is hard-coded in MATLAB (code is provided in Appendix A). It is calculated using the following relation:

$$ {}^{5}_{3}T = \left({}^{b}_{5}T\right)^{-1} \cdot {}^{b}_{3}T \tag{6} $$

The steps involved are:

1. Both ${}^{b}_{5}T$ and ${}^{b}_{3}T$ are computed using the base-to-frame transformation method discussed in Section 2.3.1.
2. Then the transformation matrix ${}^{b}_{5}T$ is inverted.
3. Next the ${}^{b}_{3}T$ (the transformation from the base frame to frame 3) is multiplied with the inverse of ${}^{b}_{5}T$.

The resulting matrix ${}^{5}_{3}T$ is:

$$ {}^{5}_{3}T = \begin{bmatrix} 0 & 0.7071 & -0.7071 & 0.2298 \\ -1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0 & 0.7071 & 0.7071 & -0.3248 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix} $$

2.4 Q 1.4: Jacobian Computation for the End-Effector

The Jacobian computation for the end-effector is a critical step in understanding the relationship between joint velocities and the velocity of the end-effector. This section discusses the theory, MATLAB implementation, and results of this computation.

2.4.1 Methodology

The Jacobian matrix $\mathbf{J}$ for a serial manipulator maps the joint velocities $\dot{\mathbf{q}}$ to the spatial velocity $\dot{\mathbf{x}}$ of the end-effector. The spatial velocity consists of the angular velocity $\boldsymbol{\omega}$ and the linear velocity $\mathbf{v}$ of the end-effector. The Jacobian is expressed as:

$$ \mathbf{J} = \begin{bmatrix} \mathbf{J}^{\omega} \\ \mathbf{J}^{v} \end{bmatrix}, $$

where:

- $\mathbf{J}^{\omega} \in \mathbb{R}^{3\times n}$ represents the contributions to angular velocity.
- $\mathbf{J}^{v} \in \mathbb{R}^{3\times n}$ represents the contributions to linear velocity.

The relationship between joint velocities $\dot{\mathbf{q}}$ and spatial velocity $\dot{\mathbf{x}}$ is given by:

$$ \dot{\mathbf{x}} = \begin{bmatrix} \boldsymbol{\omega} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} \mathbf{J}^{\omega} \\ \mathbf{J}^{v} \end{bmatrix} \dot{\mathbf{q}} \tag{7} $$

where:

- $\dot{\mathbf{q}} \in \mathbb{R}^{n}$ is the vector of joint velocities.
- $\boldsymbol{\omega} \in \mathbb{R}^{3}$ is the angular velocity of the end-effector.

- $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity of the end-effector.

The Jacobian matrix $\mathbf{J}$ for an n-joint manipulator has the general form:

$$\mathbf{J} = \begin{bmatrix} J_1^\omega & J_2^\omega & \cdots & J_n^\omega \\ J_1^v & J_2^v & \cdots & J_n^v \end{bmatrix},$$

For a manipulator with 7 joints, the Jacobian matrix $\mathbf{J}$ will take the following form:

$$\mathbf{J} = \begin{bmatrix} J_1^\omega & J_2^\omega & J_3^\omega & J_4^\omega & J_5^\omega & J_6^\omega & J_7^\omega \\ J_1^v & J_2^v & J_3^v & J_4^v & J_5^v & J_6^v & J_7^v \end{bmatrix}$$

where:

- J_i^ω represents the angular velocity contribution of joint i.
- J_i^v represents the linear velocity contribution of joint i.

1. **For Rotational Joints:** For a rotational joint i, the contribution to the Jacobian matrix is as follows:
   - The linear velocity contribution $\mathbf{J}_i^v$ is given by the cross product of the joint axis $\mathbf{z}_i$ and the vector from the joint position ${}_i^b r$ to the end-effector position ${}_e^b r$:
   $$\mathbf{J}_i^v = \mathbf{z}_i \times {}_i^e r \tag{8}$$
   where:
   $${}_i^e r = {}_e^b r - {}_i^b r \tag{9}$$
   - The angular velocity contribution $\mathbf{J}_i^\omega$ is simply the joint axis $\mathbf{z}_i$:
   $$\mathbf{J}_\omega^{(i)} = \mathbf{z}_i. \tag{10}$$
2. **For Prismatic Joints:** For a prismatic joint i, the contribution to the Jacobian matrix is as follows:
   - The linear velocity contribution $\mathbf{J}_i^v$ is along the joint axis $\mathbf{z}_i$, since prismatic joints produce linear motion along the axis:
   $$\mathbf{J}_i^v = \mathbf{z}_i \tag{11}$$
   - The angular velocity contribution $\mathbf{J}_i^\omega$ is zero because prismatic joints do not produce rotational motion:
   $$\mathbf{J}_i^\omega = \mathbf{0} \tag{12}$$

2.4.2 MATLAB Implementation

In this section, we present the MATLAB implementation for calculating the Jacobian matrix for a serial manipulator. The function `updateJacobian` computes the Jacobian matrix by iterating through all joints, using their respective transformations, and considering their types (revolute or prismatic). The function ensures that both the linear and angular velocity contributions of each joint are accurately determined.

The overall process is summarized in the flowchart shown below, encapsulating the logic implemented in the `updateJacobian` function.

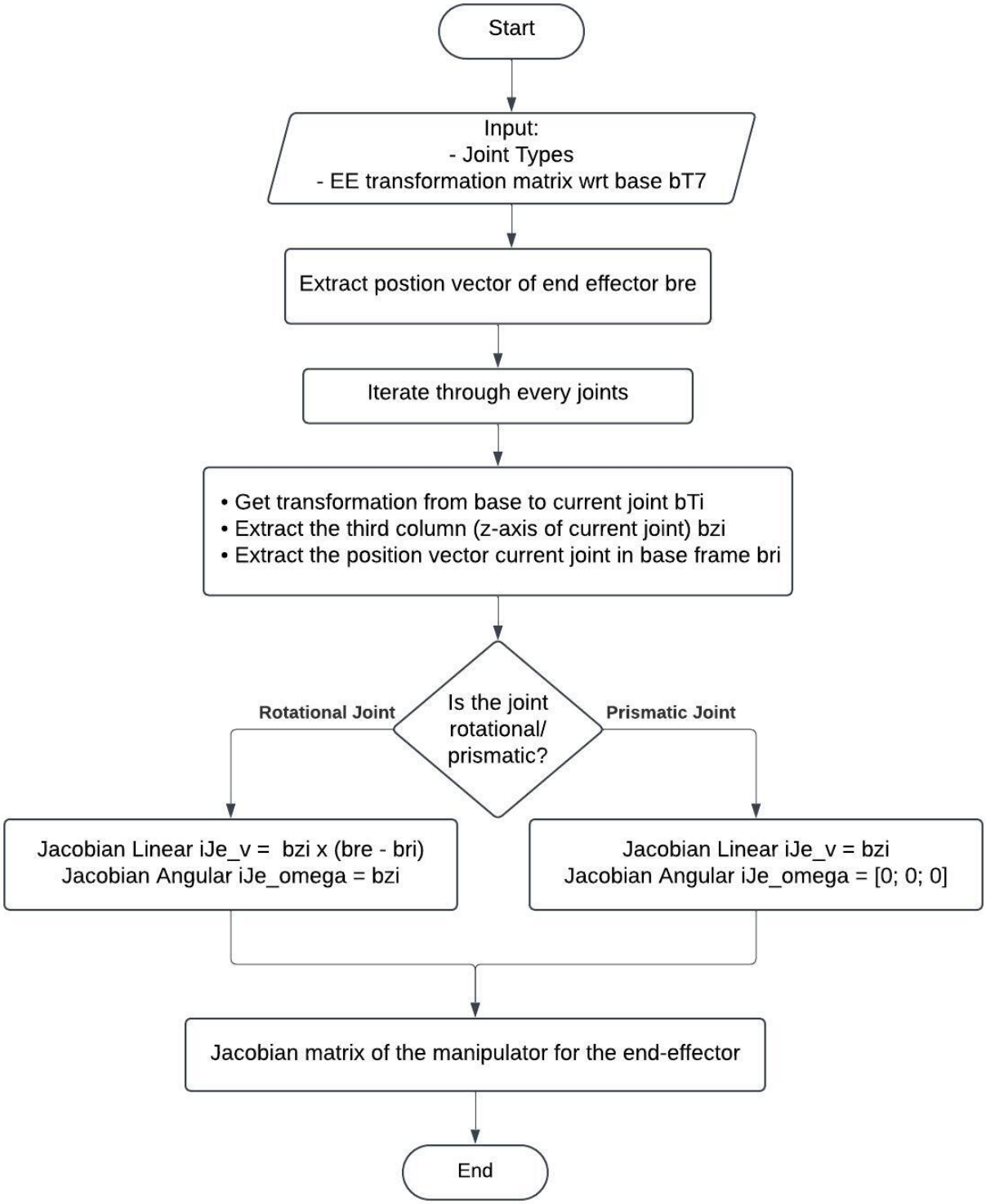

Figure 4: Matlab Implementation of Jacobian matrix computation

2.4.3 Results

The Jacobian matrix obtained is:

$$\mathbf{J} = \begin{bmatrix} 0 & -0.7071 & -0.5000 & -0.7071 & -0.7071 & 0 & -0.7071 \\ 0 & 0.7071 & -0.5000 & 0.7071 & -0.7071 & 0 & -0.7071 \\ 1.0000 & 0 & 0.7071 & 0 & 0 & 0 & 0 \\ 0.7039 & 0.2125 & 0.3475 & 0.0000 & 0.0000 & -0.7071 & 0 \\ -0.7039 & 0.2125 & -0.3475 & 0.0000 & -0.0000 & -0.7071 & 0 \\ 0 & 0.9955 & -0.0000 & 0.6950 & -0.0000 & 0.0000 & 0 \end{bmatrix}$$

- Joint 1 affects angular velocity around the z-axis and linear velocity along the x-direction.
- Joint 2 affects angular velocity around the x- and y-axes, as well as linear velocity along the y- and z-axes.
- Joint 3 influences angular velocity around the z-axis and linear velocity along the x- and y-axes.
- Joint 4 contributes to angular velocity and linear velocity along the z-direction.
- Joint 5 affects angular velocities but does not affect linear velocities.
- Joint 6 has a negative effect on linear velocity along the x- and y-axes, with no angular velocity contribution.
- Joint 7 contributes to angular velocity around the x-axis without affecting the linear velocities.

2.5 Final Robot Configuration

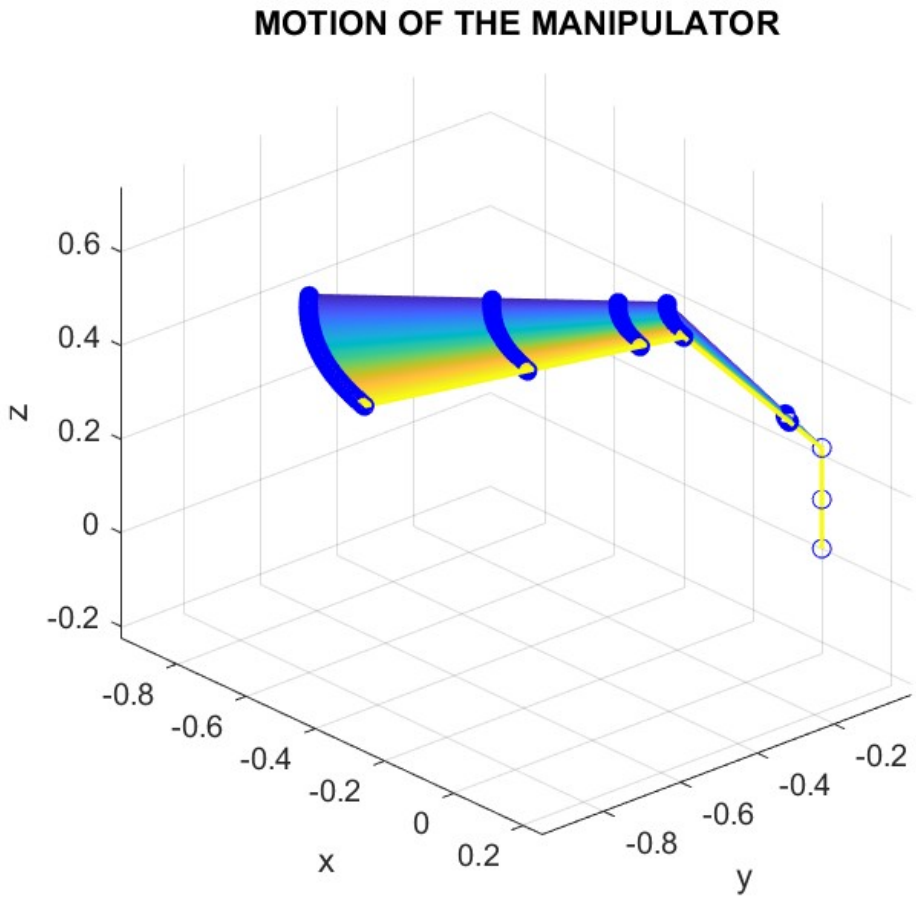

Figure 5: Motion of the Manipulator

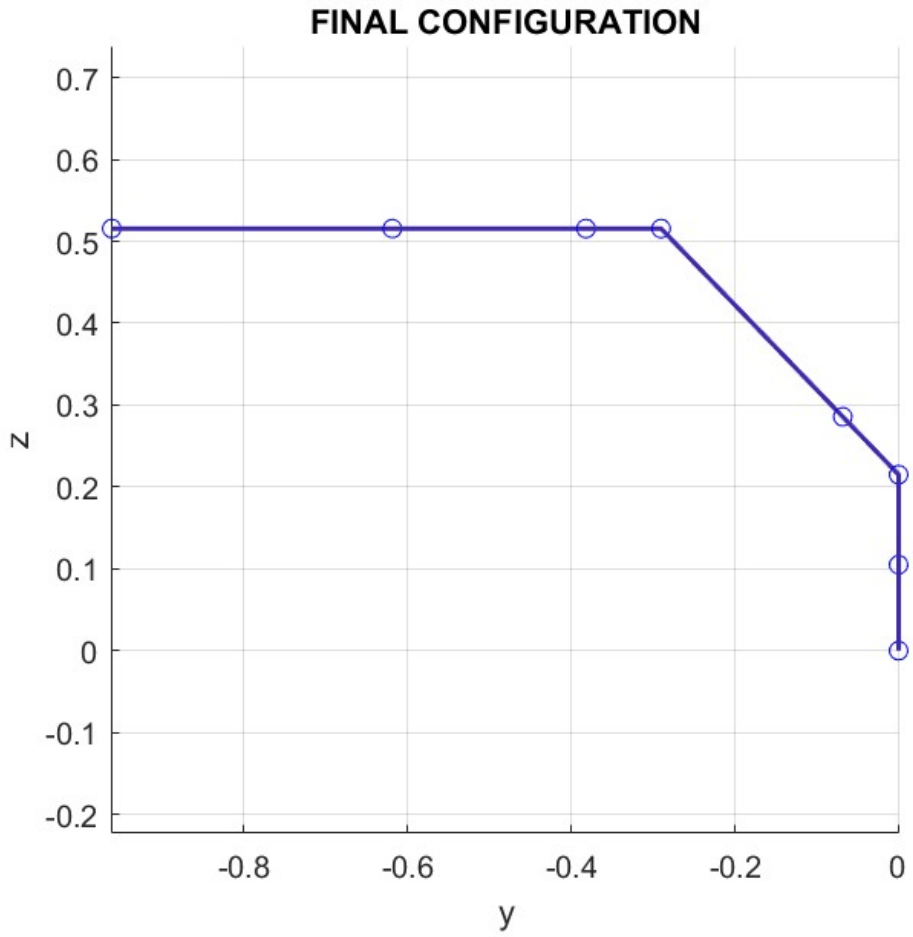

Figure 6: Final Robot configuration

3 Appendix

This section includes the MATLAB function codes and additional support materials.

3.1 Appendix A - Transformation from frame 5 to frame 3

```
function [T_3_5] = getTransform3Wrt5(self)
    %% Get the transformation matrix of frame 3 wrt frame 5
    % Step 1: Get the transformation matrix from base to frame 3
    bT3 = self.getTransformWrtBase(3);

    % Step 2: Get the transformation matrix from base to frame 5
    bT5 = self.getTransformWrtBase(5);

    % Step 3: Compute the transformation from frame 3 to frame 5 using matrix division
    T_3_5 = bT5 \ bT3;  % Equivalent to bT5_inv * bT3 but more efficient
end
```