Robotics Project Report

BAKEL BAKEL

November, 2024

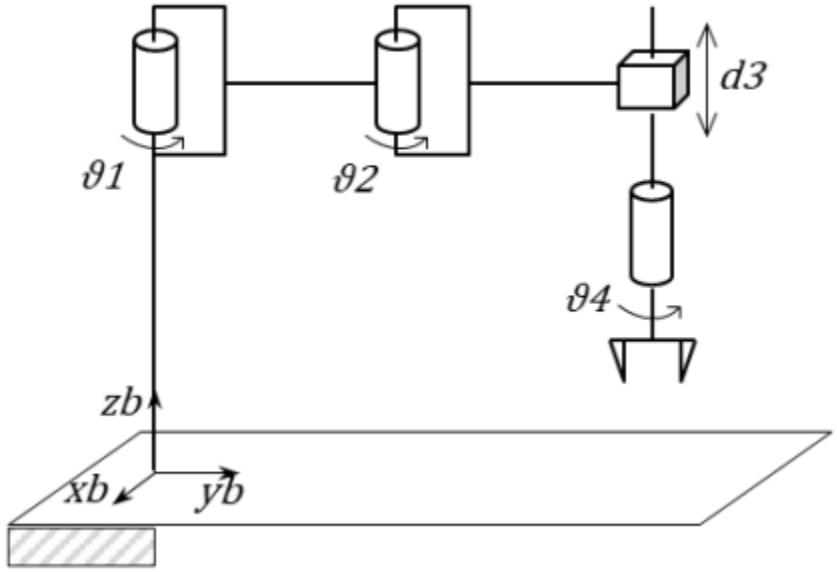
Contents

List of Figures

Chapter 1

Kinematics

Given the SCARA robot manipulator below,



I aim to determine:

1. The solution to the forward kinematic problem
2. The solution to the inverse kinematic problem

1 Solutions to Kinematic Problems

1.1 Forward Kinematic Problem

The forward kinematics problem of a robotic system can be addressed through various methodologies and conventions. In this course, the problem was solved with the use of the ***homogenous transformation matrix***;

$$T_{\mathrm{e}}^{\mathrm{b}}(\mathbf{q}) = \begin{bmatrix} \mathbf{n}_{\mathrm{e}}^{\mathrm{b}}(\mathbf{q}) & \mathbf{s}_{\mathrm{e}}^{\mathrm{b}}(\mathbf{q}) & \mathbf{a}_{\mathrm{e}}^{\mathrm{b}}(\mathbf{q}) & \mathbf{p}_{\mathrm{e}}^{\mathrm{b}}(\mathbf{q}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

and the ***Denavit–Hartenberg (D-H) Convention***, both of which provide systematic frameworks for determining the position and orientation of a robot's end-effector. For this project, I applied all the methods covered in class and extended the scope by incorporating an additional method known as ***Screw Theory***, to analyze and solve the forward kinematics.

By utilizing these methods, the forward kinematic analysis benefits from both systematic parameterization (Homogeneous Matrix and D-H Convention) and advanced geometric interpretation (Screw Theory). This comprehensive approach ensures robustness and flexibility in solving complex robotic configurations.

1.1.1 Homogeneous Transformation Matrix

This method involves the representation of spatial transformations (rotation and translation) in a unified 4x4 matrix format. Each link and joint of the robot is described by a sequence of transformations, which, when combined, determine the final position and orientation of the end-effector in the base frame.

Key Features:

1. Encodes rotation using a 3x3 rotation matrix.
2. Encodes translation as a 3x1 vector.
3. Allows easy composition of transformations using matrix multiplication.

Solution using Homogeneous Transformation Matrix

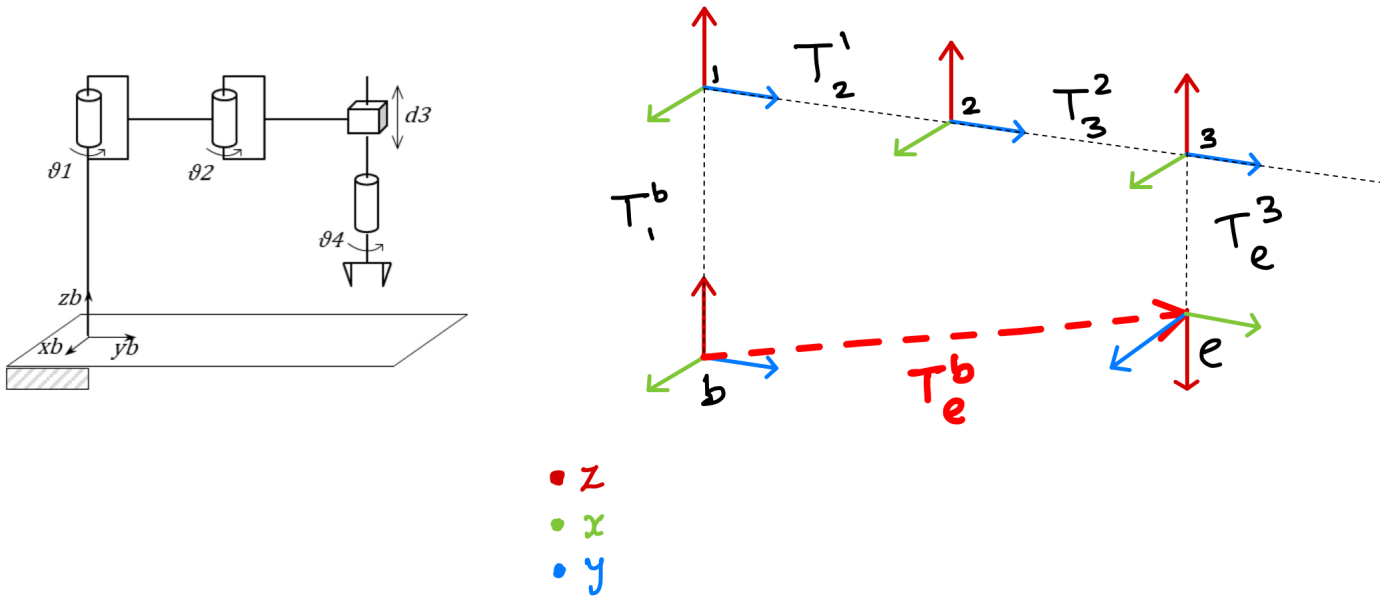


Figure 2: Diagram of Frames of the SCARA Robot

Equation 1.2 gives the formula for the forward kinematics of the scara robot as obtained from Figure 2

$$T_e^b = T_1^b \cdot T_2^1 \cdot T_3^2 \cdot T_e^3 \tag{1.2}$$

$$T_1^b = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

$$T_2^1 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 & a_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

$$T_3^2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & a_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

$$T_e^3 = \begin{bmatrix} \cos\theta_4 & -\sin\theta_4 & 0 & 0 \\ \sin\theta_4 & \cos\theta_4 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.6}$$

Finally, the forward kinematics can be determined by

$$T_e^b = T_1^b \cdot T_2^1 \cdot T_3^2 \cdot T_e^3 \tag{1.7}$$

Solve this by hand? As an engineer, I will do no such! I love mathematical rigour but I am an efficient engineer. I have decided to flex the power of python to solve symbolically the said matrics equation. The **Sympy** Library was used for this task.

Multiplication code.py ● | Workspace Trust

C: > Users > beged > OneDrive > Documents > ERASMUS MUNDUS > SEAS4.0 > Academic > Main course Work > Robotics > Bakel_Bakel_Project > Multiplication code.py >

```python
# Importing the init_printing function from the sympy library
# This is used to enable better printing of symbolic expressions, making them easier to read.
from sympy import init_printing

# Initializing pretty printing with Unicode characters for enhanced readability
# This ensures that the symbolic matrix equations and expressions appear in a clean,
# mathematical format.
init_printing(use_unicode=True)
from IPython.display import display

# Importing the symbols function from sympy
# The symbols function allows the creation of symbolic variables that can be used in equations.
# The cos and sin functions are used for symbolic trigonometric calculations.
from sympy import symbols,cos, sin, pprint, simplify

# Importing the Matrix class from sympy's matrices module
# This class is used to create and manipulate matrices
from sympy.matrices import Matrix
import math

#Define all the symbols needed
theta1,theta2,theta4,do,a1,a2,d3 = symbols ("theta1,theta2,theta4,do,a1,a2,d3")

```

Multiplication code.py ● | Workspace Trust

C: > Users > beged > OneDrive > Documents > ERASMUS MUNDUS > SEAS4.0 > Academic > Main course Work > Robotics > Bakel_Bakel_Project > Multiplication code.py > ...

```python
#Define all the transformation matrix
Tb_1 = Matrix([[cos(theta1),-sin(theta1), 0 , 0 ],
               [sin(theta1), cos(theta1), 0 , 0 ],
               [     0     ,     0      , 1 , do],
               [     0     ,     0      , 0 , 1 ]])

T1_2 = Matrix([[cos(theta2),-sin(theta2), 0 , 0 ],
               [sin(theta2), cos(theta2), 0 , a1],
               [     0     ,     0      , 1 , 0 ],
               [     0     ,     0      , 0 , 1 ]])

T2_3 = Matrix([[     1     ,     0      , 0 , 0 ],
               [     0     ,     1      , 0 , a2],
               [     0     ,     0      , 1 , d3],
               [     0     ,     0      , 0 , 1 ]])

T3_e = Matrix([[cos(theta4),-sin(theta4), 0 , 0 ],
               [sin(theta4), cos(theta4), 0 , 0 ],
               [     0     ,     0      , -1 , 0 ],
               [     0     ,     0      , 0 , 1 ]])

#Multiplying all the transformation matrix to get the final matrix
Tb_e = Tb_1 * T1_2 * T2_3 * T3_e

# Simplifying each component of the resulting transformation matrix
Tb_e_simplified = simplify(Tb_e)

# Displaying the simplified transformation matrix
pprint(Tb_e_simplified)

pprint(simplify(Tb_1 * T1_2))
pprint(simplify(Tb_1 * T1_2* T2_3))
pprint(simplify(Tb_1 * T1_2* T2_3* T3_e))
```

Upon running the code, the following matrix was gotten. This is the solution to our forward kinematic problem.

$$\begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_4) & -\sin(\theta_1 + \theta_2 + \theta_4) & 0 & -a_1 \cdot \sin(\theta_1) - a_2 \cdot \sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2 + \theta_4) & \cos(\theta_1 + \theta_2 + \theta_4) & 0 & a_1 \cdot \cos(\theta_1) + a_2 \cdot \cos(\theta_1 + \theta_2) \\ 0 & 0 & -1 & d_3 + do \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

VOILAA!!!

1.1.2 Denavit–Hartenberg (D-H) Convention

This is a standardized convention for modeling the forward kinematics of serial-chain robots. It simplifies the process by defining each joint's coordinate frame through four parameters: link length, link twist, link offset, and joint angle.

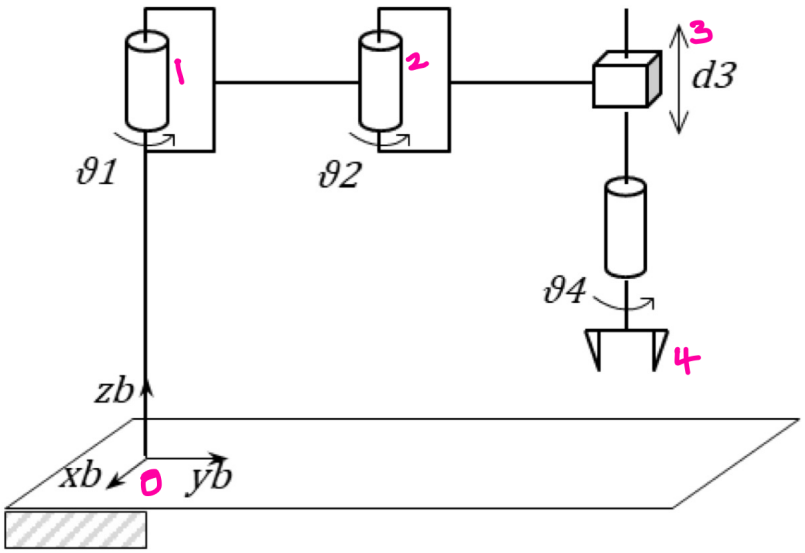
Key Features:

i. Reduces the complexity of defining coordinate systems.

ii. Provides a compact parameterization of the robot's kinematic chain.

iii. Supports systematic derivation of transformation matrices for each joint.

a. Firstly, number the joints/points of reference where the frames will be located.



b. The joint 1 is revolute and following DH Convention, the Z axis will be given as in the image below. The x and y axis of Frame 1 can be obtained correctly as given in the image below.

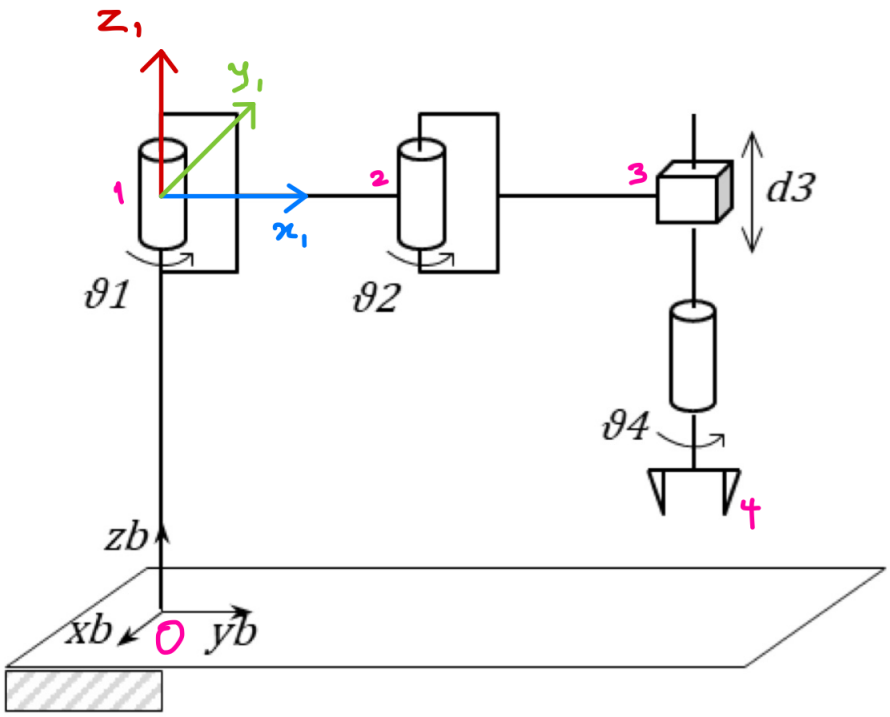


Figure 1.1: Determination of Frame 1

c. According to the Denavit-Hartenberg (DH) convention, it is often convenient to align the base frame with Frame 1 for simplicity in deriving the transformation matrices. However, in the context of this problem, the base frame is explicitly defined as part of the problem statement and is distinct from Frame 1.

While I could alter the base frame to match Frame 1 in my solution, this approach is not advisable for practical reasons, especially in an industrial environment:

i. Adherence to Given Specifications: In real-world engineering problems, the coordinate frames are often pre-defined by the system, client requirements, or the physical environment. Modifying the base frame for convenience could lead to misinterpretations or deviations from the intended design.

ii. Compatibility with Other Systems: Industrial setups often involve integration with existing systems or machines. The base frame might be used as a reference for other components, sensors, or control systems. Changing it in calculations could lead to inconsistencies or errors when interpreting results in the actual system.

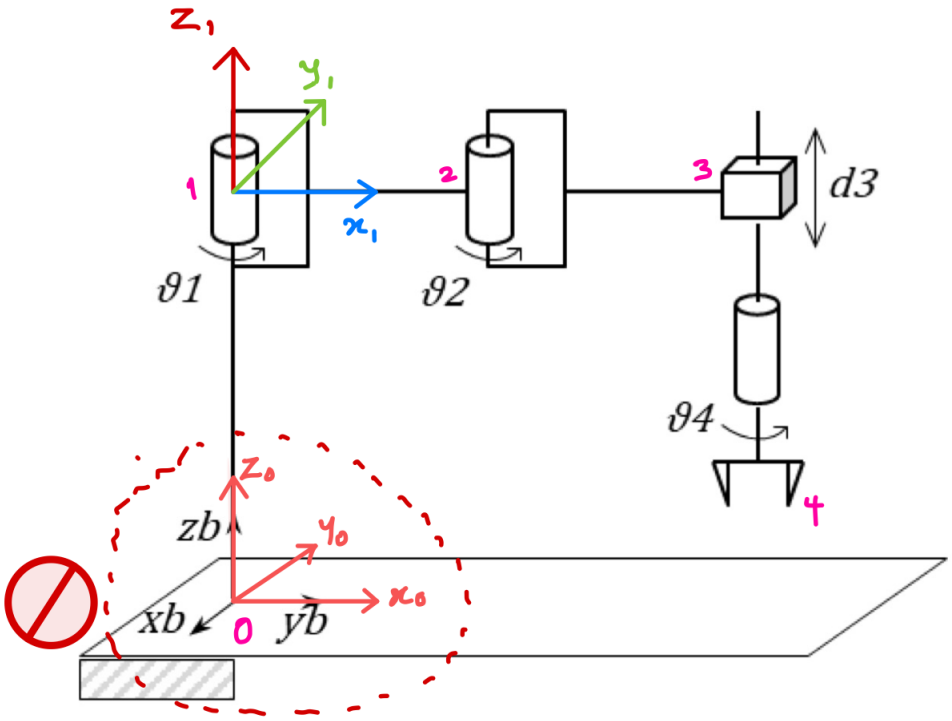


Figure 1.2: Illustration of Retaining the Given Base Frame for Consistency with Problem Specifications.

d. The joint 2 is also revolute so the axis is determined as follows;

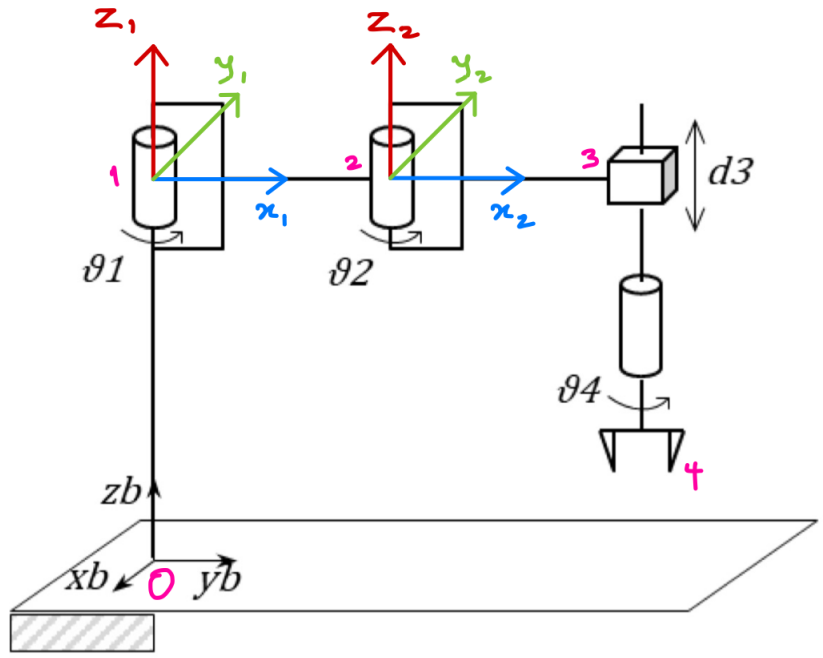


Figure 1.3: Determination of Frame 2

e. The joint 3 however is a prismatic so the axis is determined as follows;

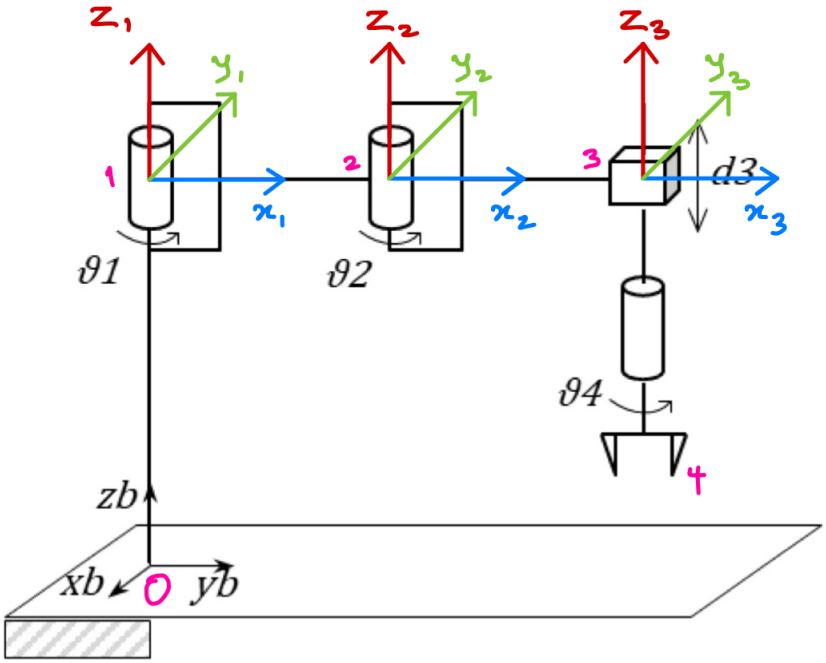


Figure 1.4: Determination of Frame 3

e. For joint 4, there are some school of thought that insist on putting the frame at the center of the revolute joint and still have a separate frame for the end effector. However, with no clear distance between the joint and the end effector, there is no point in having two separate frames for this configuration and the frame of the revolute joint and the end effector will not only be identical but coincidental.

Secondly, from the nature of the question and the position of the end effector, it is intuitive to assign the (z_4) facing downwards. But since this doesnt affect much aside our perception of the end effector (which we can easily interpret), I will follow the DH convention that advices the end effector frame to be identical to that of the last frame (Frame 3).

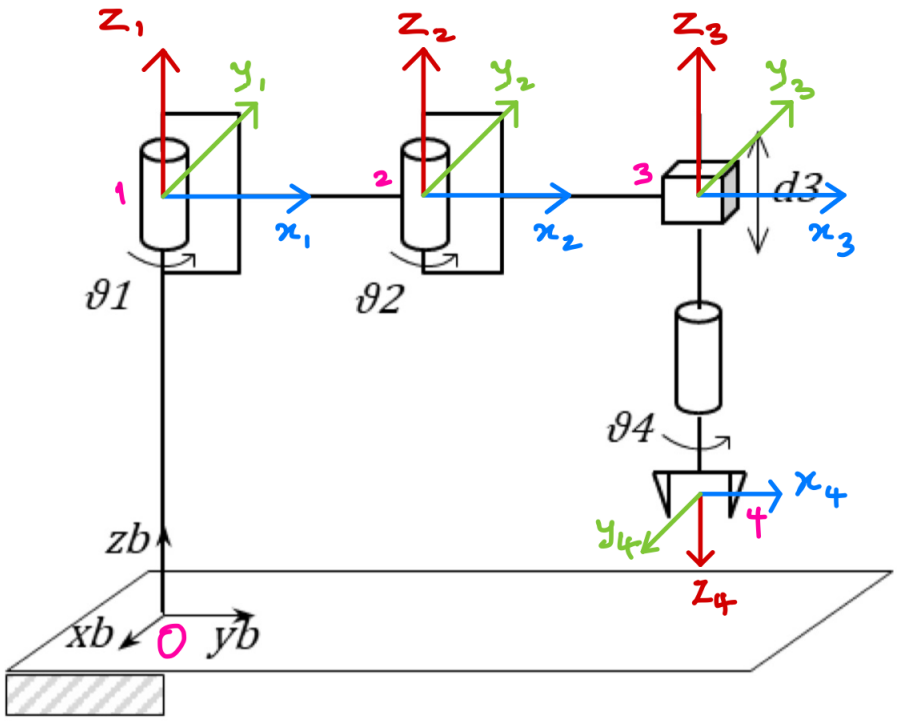


Figure 1.5: Schematics showing all the DH Frames of the SCARA robot

Table 1.1: Table showing DH Parameters

Link	a_i	α_i	d_i	θ_i
1	0_1	0	d_0	$\frac{\pi}{2} + \theta_1$
2	a_1	0	0	θ_2
3	a_2	0	0	0
4	0	0	d_3	θ_4

1.1.3 Screw Theory

Screw Theory offers a geometric and algebraic framework to describe motion and kinematics in terms of twists (velocity screws) and wrenches (force screws). It represents the motion of a rigid

body as a combination of rotational and translational components about a screw axis.

Key Features:

i. Models motion using Plücker coordinates for lines.

ii. Efficiently handles instantaneous kinematics, including singularities and constraints.

iii. Extends naturally to spatial motion analysis, offering insights into both the kinematic and dynamic aspects of robotic systems.

1.1.4 Verification with Corke's Robotics Toolbox

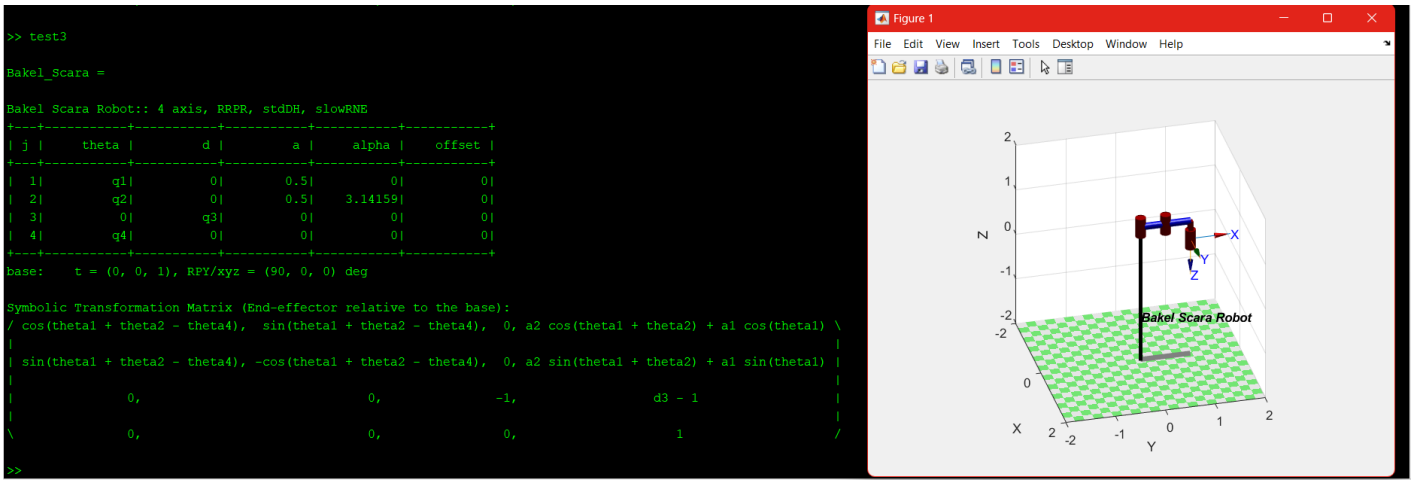


Figure 1.6: Schematics showing all the DH Frames of the SCARA robot

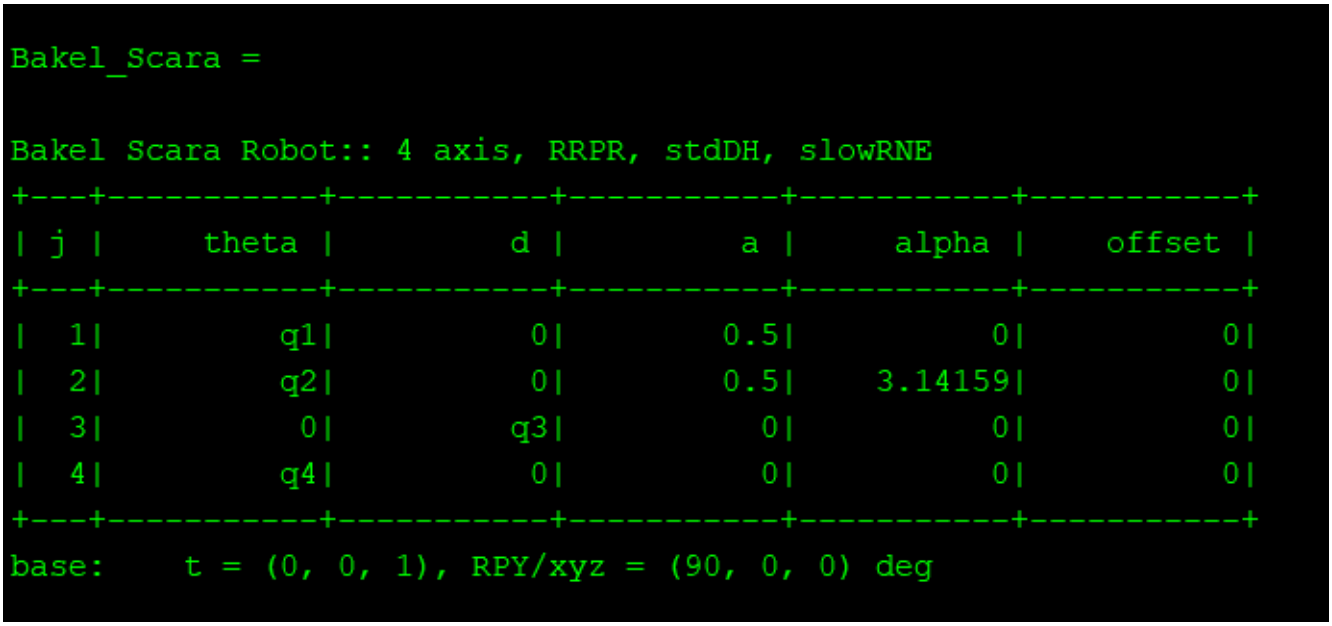


Figure 1.7: Schematics showing all the DH Frames of the SCARA robot

```
Symbolic Transformation Matrix (End-effector relative to the base):
/ cos(theta1 + theta2 - theta4),  sin(theta1 + theta2 - theta4),  0, a2 cos(theta1 + theta2) + a1 cos(theta1) \
|                                                                                                            |
| sin(theta1 + theta2 - theta4), -cos(theta1 + theta2 - theta4),  0, a2 sin(theta1 + theta2) + a1 sin(theta1) |
|                                                                                                            |
|              0,                              0,              -1,                 d3 - 1                  |
|                                                                                                            |
\              0,                              0,               0,                   1                    /
```

Figure 1.8: Schematics showing all the DH Frames of the SCARA robot

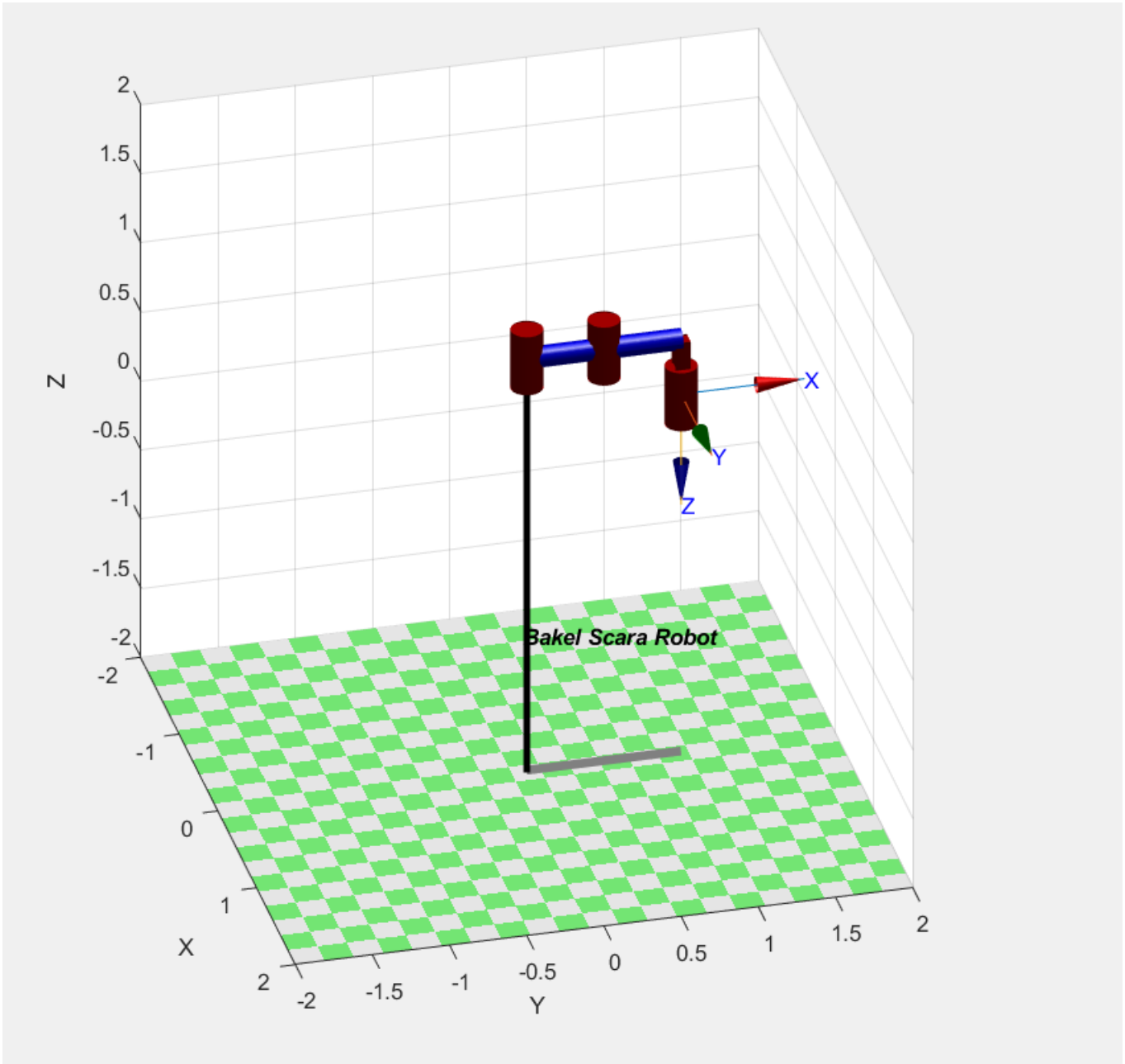


Figure 1.9: Schematics showing all the DH Frames of the SCARA robot

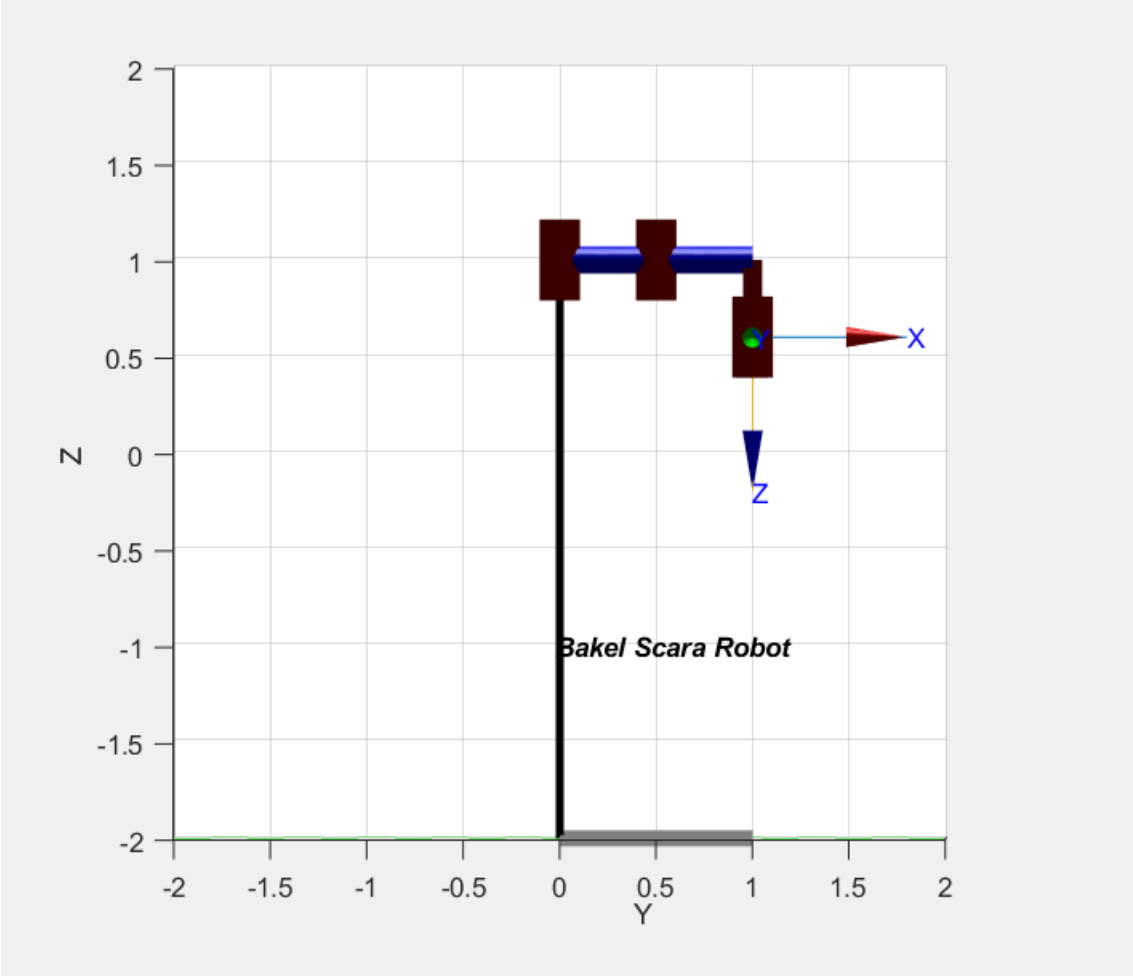


Figure 1.10: Schematics showing all the DH Frames of the SCARA robot

1.2 Inverse Kinematic Problem

The inverse kinematic problem is formulated as follows:

Given the position and orientation of the end effector, find the joint variables of the robot.

The given SCARA robot has 4 DOF, so for its position and orientation to be fully represented, we need 4 variables which are; x, y, z for position, and ψ for orientation

$$\psi = \theta_1 + \theta_2 + \theta_3 \tag{1.8}$$

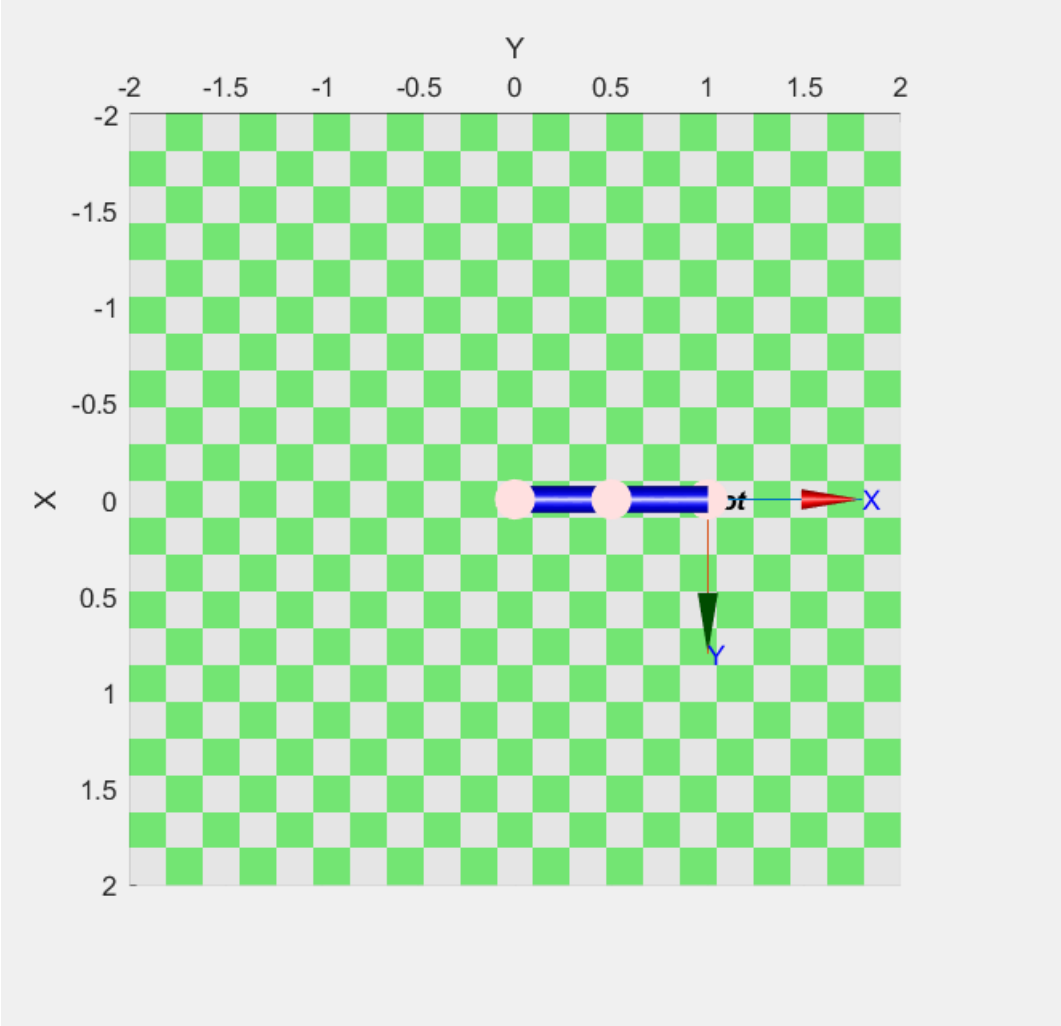


Figure 1.11: Top View (XY Plane) of the robot

$$
\begin{aligned}
&\text{As } z_2 \text{ is olory } z_4 \\
&p_2\,(x_2, y_2) \geq (x_1) \\
&r^2 = x^2 + y^2 \\
&r^2 = a_1^2 + a_2^2 - 2a_1 a_2 \cos x \\
&1\theta_2 = \pi - \alpha \\
&\cos\theta_2 = -\cos\alpha \\
&r^2 = a_1^2 + a_2^2 + 2a_1 a_2 \cos\theta_2 \\
&\sin^2\theta_2 + \cos^2\theta_2 = 1 \\
&\sin\theta_2 = \sqrt{1 - \cos^2\theta_2} \\
&\beta = \tan^{-1}\frac{a_2 \sin\theta_2}{a_1 + a_2 \cos_2} \\
&\gamma = \tan^{-1}\frac{4}{x} \\
&\theta_1 = \gamma - \frac{\beta}{r} \\
&\theta_1 = \tan^{-1}\frac{1}{x} - \tan^{-1}\frac{a_2 \sin\theta_2}{a_1 + a_2 \cos\theta_2}
\end{aligned} \tag{1.9}
$$

$$
\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{a_2 \sin\theta_2}{a_1 + a_2 \cos\theta_2}\right) \tag{1.10}
$$

$$
\theta_2 = \cos^{-1}\left(\frac{x^2 + y^2 - (a_1^2 + a_2^2)}{2a_1 a_2}\right) \tag{1.11}
$$

1.2.1 Verification with Corke's Robotics Toolbox

```
Symbolic Inverse Kinematics Solutions:
theta1:
atan2(py, px)

theta2:
pi - acos((a1^2 + a2^2 - px^2 - py^2)/(2*a1*a2))

d3:
1 - pz

theta4:
phi - pi - atan2(py, px) + acos((a1^2 + a2^2 - px^2 - py^2)/(2*a1*a2))
```

Figure 1.12: Verification of Inverse Kinematics from Corke's Robotics Toolbox

1.3 Representing my end-effector in terms of Euler angles

It is possible to convert our SCARA 4-DOF configuration and the transformation matrix into Euler angles.

Euler angles describe the orientation of a rigid body (in this case, the end-effector) as a sequence of rotations around different axes. Since the SCARA robot has only 1 rotational DOF (the combined rotation angle), the process simplifies significantly.

Analyzing the Transformation Matrix:
The transformation matrix you provided is:

$$T = \begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_4) & -\sin(\theta_1 + \theta_2 + \theta_4) & 0 & x \\ \sin(\theta_1 + \theta_2 + \theta_4) & \cos(\theta_1 + \theta_2 + \theta_4) & 0 & y \\ 0 & 0 & -1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The rotational part (upper-left 3×3) shows the orientation:

$$R = \begin{bmatrix} \cos(\theta_1 + \theta_2 + \theta_4) & -\sin(\theta_1 + \theta_2 + \theta_4) & 0 \\ \sin(\theta_1 + \theta_2 + \theta_4) & \cos(\theta_1 + \theta_2 + \theta_4) & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

The translational part shows the position:

$$P = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Steps to Extract Euler Angles: 1. Choose an Euler Angle Convention:

Euler angles can be defined in different conventions (e.g., ZYX, XYZ). For a SCARA robot, the typical convention is Z-X-Z or Z-Y-Z since rotations occur primarily around the Z-axis. 2. Decompose the Rotation Matrix: 2. Decompose the Rotation Matrix:

For the $\mathrm{Z} - \mathrm{Y} - \mathrm{Z}$ convention: - The angles are ϕ (first Z-axis rotation), θ (Y-axis rotation), and ψ (second Z-axis rotation).

From the rotation matrix R, the Euler angles can be extracted as follows:

$$\phi = \arctan 2\left(R_{2,3}, R_{3,3}\right) \tag{1.12}$$

$$\theta = \arcsin\left(-R_{1,3}\right) \tag{1.13}$$

$$\psi = \arctan 2\left(R_{1,2}, R_{1,1}\right) \tag{1.14}$$

3. Simplify for the SCARA Robot:

For the SCARA robot, since the rotation matrix is planar (no significant Y-axis rotation), the Euler angles simplify: - The primary orientation is defined by $(\theta_1 + \theta_2 + \theta_4)$. - There are no out-of-plane rotations.

Thus, the Euler angles reduce to:

$$\phi = \theta_1 + \theta_2 + \theta_4$$

$\theta = 0$ (no Y-axis rotation, as the SCARA is planar) $\psi = 0$ (no additional Z-axis rotation beyond the combined angle).

Final Result: The Euler angles for this SCARA robot are:

$$\text{Euler Angles: } \phi = \theta_1 + \theta_2 + \theta_4, \quad \theta = 0, \quad \psi = 0$$

This representation simplifies because the robot's motion is constrained to the plane, and all rotation occurs around the Z-axis.

Chapter 2

Working Space

The workspace can be defined mathematically as:

$$p = p(q), \quad q_{im} \leq q_i \leq q_{iM}, \quad i = 1, \ldots, n \tag{2.1}$$

where q represents the joint variables $(\theta_1, \theta_2, d_3)_{\mathrm{r}}$ and their limits are as specified in Figure 4.

1. Position Components: Using forward kinematics, the end-effector position (x, y, z) is derived as:

$$x = a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2) \tag{2.2}$$

$$y = a_1 \sin(\theta_1) + a_2 \sin(\theta_1 + \theta_2) \tag{2.3}$$

$$z = d_0 + d_3 \tag{2.4}$$

Here, (x, y) defines the planar reach due to the rotational joints, while z incorporates the prismatic joint's effect.

2. Range of Motion: - Rotational Joints (θ_1, θ_2) : The range of motion for θ_1 $(-90^\circ$ to $90^\circ)$ and θ_2 $(-90^\circ$ to $45^\circ)$ determines the extents of the toroidal workspace in the XY-plane. - Prismatic Joint (d_3) : The vertical range from $d_{3m} = 0.25m$ to $d_{3M} = 1m$ produces the cylindrical height in the Z-direction.

3. Effect of End-Effector Orientation (θ_4) : - While θ_4 is a degree of freedom of the SCARA robot, it does not influence the reachable workspace, as it only governs the angular orientation of the end-effector. This distinction highlights that the reachable workspace is purely positional, while the dexterous workspace includes orientation capabilities.

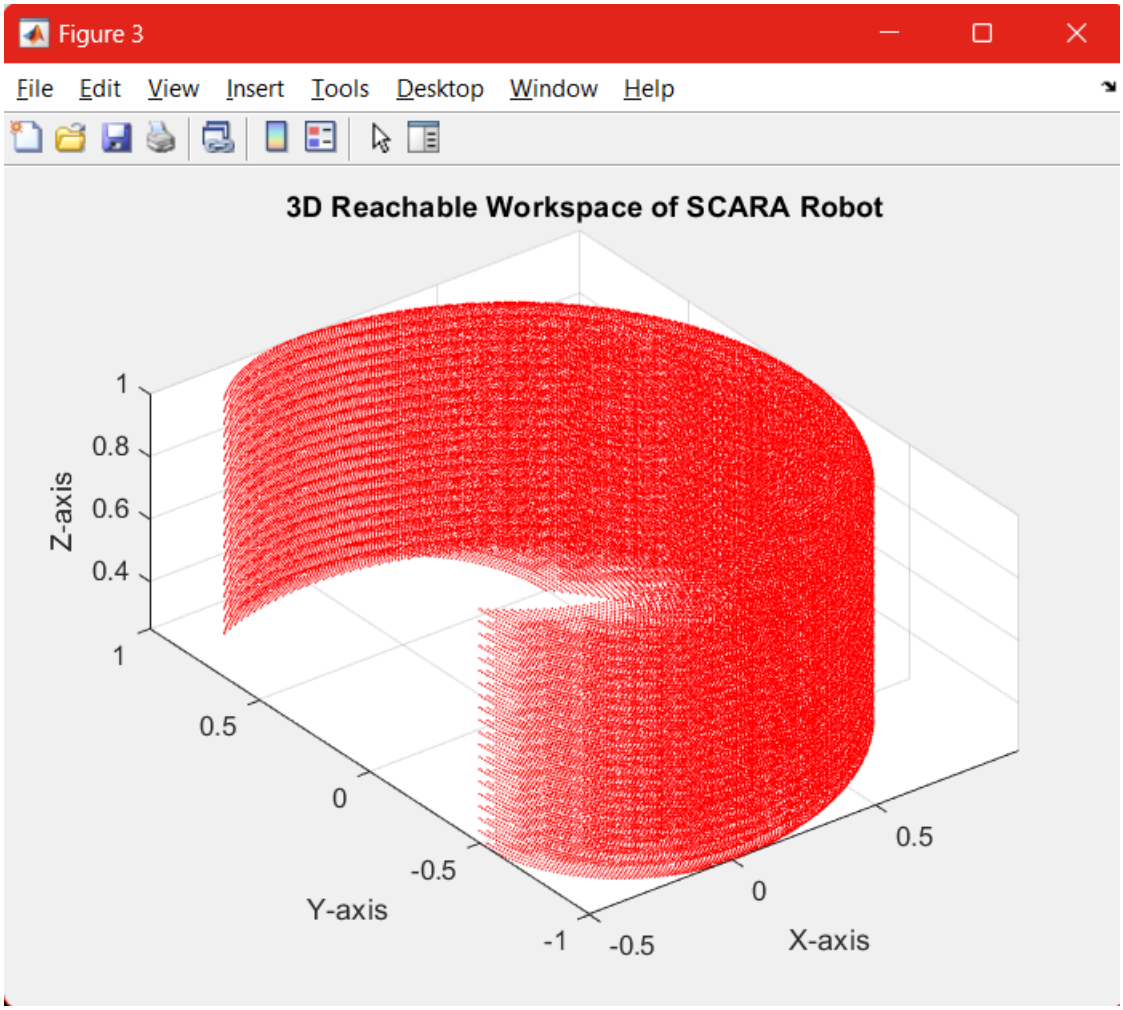


Figure 2.1: Verification of Inverse Kinematics from Corke's Robotics Toolbox

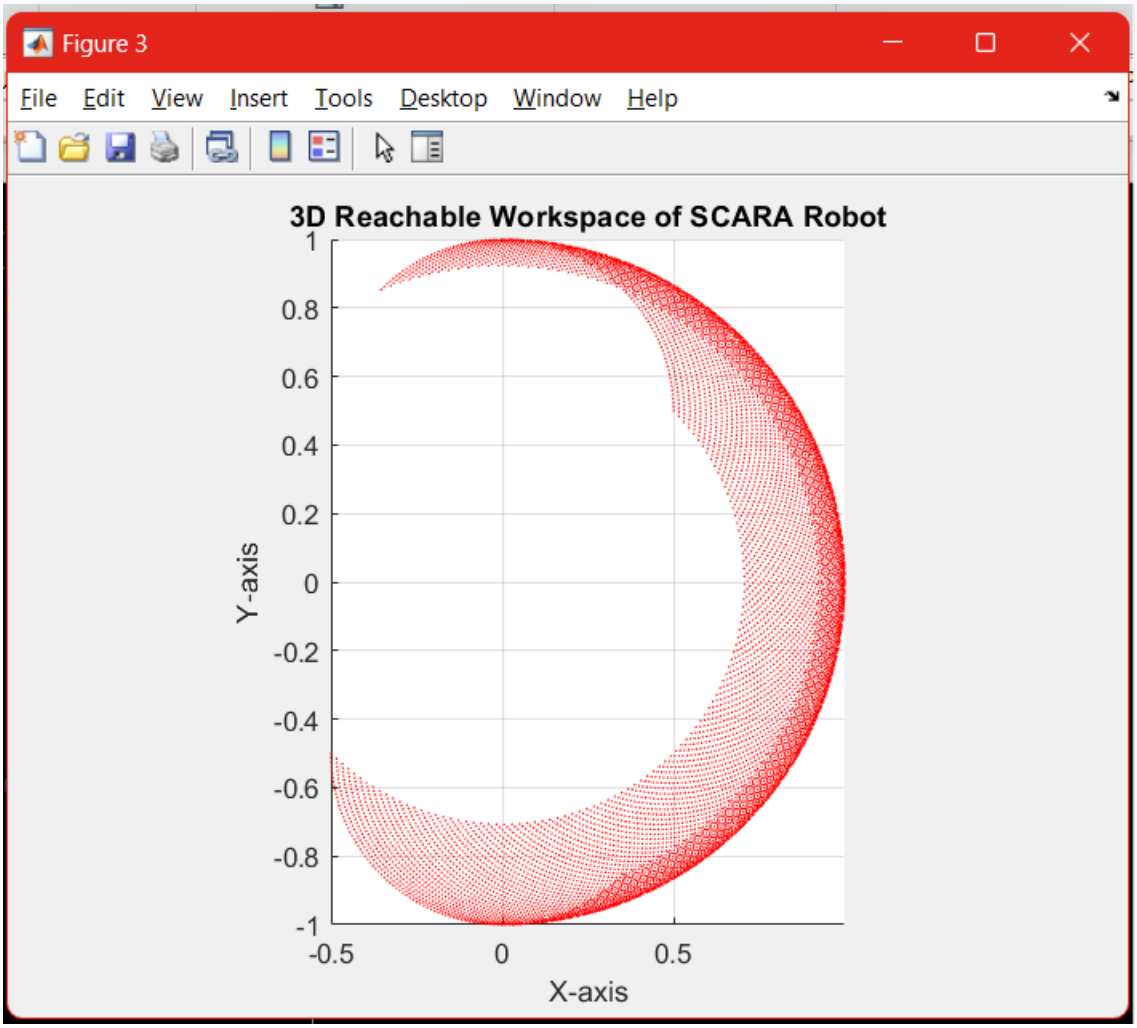


Figure 2.2: Verification of Inverse Kinematics from Corke's Robotics Toolbox

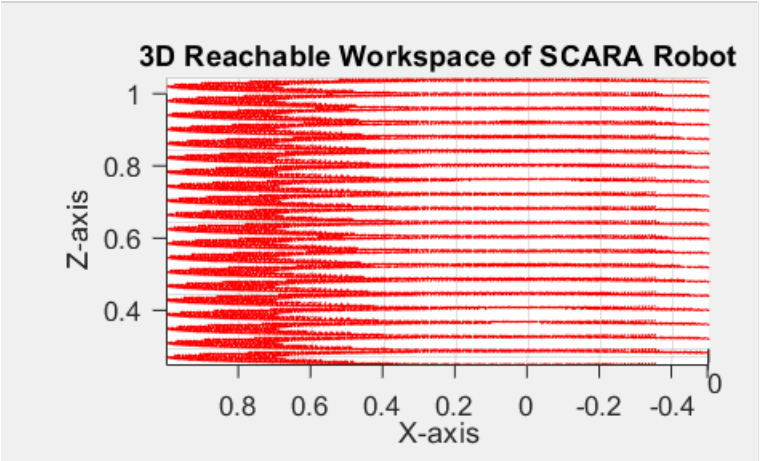


Figure 2.3: Verification of Inverse Kinematics from Corke's Robotics Toolbox

The reachable workspace of the SCARA robot is visualized in the first three figures. The

workspace analysis takes into account the range of motion of the robot's joints, as described by the joint variable limits in the fourth figure and the mathematical description of the reachable workspace shown in the last figure. Below is a detailed discussion of the workspace visualization and its relation to the robot's joint parameters and mathematical framework.

Workspace Visualization 1. Figure 1 (3D Workspace): - This figure depicts the 3D reachable workspace of the SCARA robot. The workspace forms a cylindrical volume due to the combination of rotational motion in the XY-plane and linear motion along the Z-axis. - The inclusion of the prismatic joint (d_3) in the Z-direction expands the workspace into the third dimension. The continuous red points represent all positions that the robot's endeffector can achieve within its joint limits.

2. Figure 2 (Top View - XY Plane): - The workspace in the XY-plane appears as a toroidal or annular region. This is a direct result of the rotational joints θ_1 and θ_2, with their respective ranges creating a circular area when combined with the link lengths ($a_1 = a_2 = 0.5$ m). - The inner and outer radii of the toroidal region correspond to the minimum and maximum reach of the robot, given by:

$$r_{\min} = |a_1 - a_2|, \quad r_{\max} = a_1 + a_2$$

3. Figure 3 (Side View - XZ Plane): - This view illustrates the linear movement introduced by the prismatic joint (d_3), which shifts the workspace along the Z -axis. The prismatic joint range ($d_{3m} = 0.25$ m to $d_{3M} = 1$ m) causes the cylindrical workspace to extend vertically within this range.

The 3D workspace analysis demonstrates how the SCARA robot's joint configurations and ranges contribute to its reachable volume. The cylindrical workspace in Figure 1 combines the toroidal XYplane workspace (Figure 2) with the linear Z-axis motion (Figure 3). The joint limits and mathematical formulation (Figures 4 and 5) align well with the observed workspace, confirming the robot's kinematic design and functionality.

It is also evident that the end-effector orientation (θ_4) does not alter the physical boundaries of the reachable workspace, as it solely adjusts the angular orientation of the end-effector at a given position. This distinction is critical when analyzing the robot's capabilities for applications that require precise positioning versus those requiring specific orientations.

This analysis provides a comprehensive understanding of the SCARA robot's operational capabilities, useful for tasks requiring precise motion planning and workspace optimization.

Chapter 3

Jacobian

3.1 Geometric Jacobian

The Geometric Jacobian will be computed using the following procedure:

$$\begin{bmatrix} \mathbf{J}_{\mathbf{P}_i} \\ \mathbf{J}_{\mathbf{O}_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \\ \mathbf{0} \end{bmatrix}, & \text{for a } \textit{prismatic} \text{ joint} \\ \begin{bmatrix} \mathbf{z}_{i-1} \times (\mathbf{p} - \mathbf{p}_{i-1}) \\ \mathbf{z}_{i-1} \end{bmatrix}, & \text{for a } \textit{revolute} \text{ joint} \end{cases}$$

For our SCARA robot, the Jacobian will be in the form below.

$$\mathbf{J}(\mathbf{q}) = \begin{bmatrix} \mathbf{Z}_0 \times (\mathbf{p} - \mathbf{p}_0) & \mathbf{Z}_1 \times (\mathbf{p} - \mathbf{p}_1) & \mathbf{Z}_2 & \mathbf{Z}_3 \times (\mathbf{p} - \mathbf{p}_3) \\ \mathbf{Z}_0 & \mathbf{Z}_1 & 0 & \mathbf{Z}_3 \end{bmatrix} \tag{3.1}$$

In order to solve the geometrical jacobian we have to recall the position of each joint which will require the homogenous transformation matrix calculated earlier on in Chapter 2, Section 1.1.1. The homogenous transformation matrix of each joint can be given as follows;

$$T_1^b = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 & 0 \\ 0 & 0 & 1 & d_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.2}$$

$$T_1^b.T_2^1 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & -a_1 \cdot \sin(\theta_1) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1 \cdot \cos(\theta_1) \\ 0 & 0 & 1 & d_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$T_1^b.T_2^1.T_3^1 = \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 & -a_1 \cdot \sin(\theta_1) - a_2 \cdot \sin(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 & a_1 \cdot \cos(\theta_1) + a_2 \cdot \cos(\theta_1+\theta_2) \\ 0 & 0 & 1 & d_3 + d_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$T_1^b.T_2^1.T_3^2.T_e^3 = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_4) & -\sin(\theta_1+\theta_2+\theta_4) & 0 & -a_1 \cdot \sin(\theta_1) - a_2 \cdot \sin(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2+\theta_4) & \cos(\theta_1+\theta_2+\theta_4) & 0 & a_1 \cdot \cos(\theta_1) + a_2 \cdot \cos(\theta_1+\theta_2) \\ 0 & 0 & -1 & d_3 + d_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

Where equation 3.1, 3.2, 3.3 and 3.4 resolves to the matrix at each joints. From here, we can get the (P_i) of each joint i. From the above set of equations, it is easy to deduce that;

$$\mathbf{Z}_0 = \mathbf{Z}_1 = \mathbf{Z}_2 = \mathbf{Z}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.6}$$

$$\mathbf{p} = \begin{bmatrix} -a_1 \cdot \sin(\theta_1) - a_2 \cdot \sin(\theta_1 + \theta_2) \\ a_1 \cdot \cos(\theta_1) + a_2 \cdot \cos(\theta_1 + \theta_2) \\ d_3 + d_o \\ 1 \end{bmatrix} \tag{3.7}$$

$$\mathbf{P}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.8}$$

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \\ d_0 \\ 1 \end{bmatrix} \tag{3.9}$$

$$\mathbf{p}_2 = \begin{bmatrix} -a_1 \cdot \sin(\theta_1) \\ a_1 \cdot \cos(\theta_1) \\ d_o \\ 1 \end{bmatrix} \tag{3.10}$$

$$\mathbf{p}_3 = \begin{bmatrix} -a_1 \cdot \sin(\theta_1) - a_2 \cdot \sin(\theta_1 + \theta_2) \\ a_1 \cdot \cos(\theta_1) + a_2 \cdot \cos(\theta_1 + \theta_2) \\ d_3 + d_o \\ 1 \end{bmatrix} \tag{3.11}$$

Linear Velocity Component (J_p) :

$$J_p = \begin{bmatrix} Z_0 \times (p - p_0) & Z_1 \times (p - p_1) & Z_2 & Z_3 \times (p - p_3) \end{bmatrix}$$

1. For $Z_0 \times (p - p_0)$:

$$p - p_0 = p_3 - p_0 = \begin{bmatrix} -a_1 \sin\theta_1 - a_2 \sin(\theta_1 + \theta_2) \\ a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \\ d_3 + d_0 \end{bmatrix}$$

$$Z_0 \times (p - p_0) = \begin{bmatrix} -(a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2)) \\ -(a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2)) \\ 0 \end{bmatrix}$$

2. For $Z_1 \times (p - p_1)$:

$$p - p_1 = p_3 - p_1 = \begin{bmatrix} -a_1 \sin\theta_1 - a_2 \sin(\theta_1 + \theta_2) \\ a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \\ d_3 \end{bmatrix}$$

$$Z_1 \times (p - p_1) = \begin{bmatrix} -a_2 \cos(\theta_1 + \theta_2) \\ -a_2 \sin(\theta_1 + \theta_2) \\ 0 \end{bmatrix}$$

3. For Z_2 :

$$Z_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

4. For $Z_3 \times (p - p_3)$:

$$p - p_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Z_3 \times (p - p_3) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Angular Velocity Component (J_o) :

$$J_o = \begin{bmatrix} Z_0 & Z_1 & 0 & Z_3 \end{bmatrix}$$

$$J_o = \left[\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right]$$

Final Geometric Jacobian:

$$J(q) = \begin{bmatrix} -(a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2)) & -a_2 \cos(\theta_1 + \theta_2) & 0 & 0 \\ -(a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2)) & -a_2 \sin(\theta_1 + \theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

This Jacobian matrix $J(q)$ captures the relationship between the joint velocities $\dot{q} = \left[\dot{\theta}_1, \dot{\theta}_2, \dot{d}_3, \dot{\theta}_4\right]^T$ and the end-effector velocities.

3.1.1 Verification with Corke's Robotics Toolbox

3.2 Analytical Jacobian

The analytical Jacobian relates the joint velocities $\dot{q} = \left[\dot{\theta}_1, \dot{\theta}_2, \dot{d}_3, \dot{\theta}_4\right]$ to the linear velocity ($\dot{p}$) and angular velocity (ω) of the end-effector. It is derived using the position equations from forward kinematics.

Forward Kinematics of the SCARA Robot The position of the end-effector is given by:

$$x = a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2)$$
$$y = a_1 \sin(\theta_1) + a_2 \sin(\theta_1 + \theta_2)$$
$$z = d_0 + d_3$$

The orientation of the end-effector is described by θ_4, the rotation angle about the Z -axis. 1. Partial Derivatives for Linear Velocity (J_p) :

The linear velocity $\dot{p} = [\dot{x}, \dot{y}, \dot{z}]$ is derived by taking partial derivatives of x, y, and z with respect to the joint variables θ_1, θ_2, d_3, and θ_4. - For x :

$$\frac{\partial x}{\partial \theta_1} = -a_1 \sin(\theta_1) - a_2 \sin(\theta_1 + \theta_2)$$
$$\frac{\partial x}{\partial \theta_2} = -a_2 \sin(\theta_1 + \theta_2)$$
$$\frac{\partial x}{\partial d_3} = 0$$
$$\frac{\partial x}{\partial \theta_4} = 0$$

- For y :

$$\frac{\partial y}{\partial \theta_1} = a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2)$$
$$\frac{\partial y}{\partial \theta_2} = a_2 \cos(\theta_1 + \theta_2)$$
$$\frac{\partial y}{\partial d_3} = 0$$
$$\frac{\partial y}{\partial \theta_4} = 0$$

- For z:

$$\frac{\partial z}{\partial \theta_1} = 0$$
$$\frac{\partial z}{\partial \theta_2} = 0$$
$$\frac{\partial z}{\partial d_3} = 1$$
$$\frac{\partial z}{\partial \theta_4} = 0$$

2. Partial Derivatives for Angular Velocity (J_o) :

The angular velocity is influenced only by the rotational joints ($\theta_1, \theta_2, \theta_4$). Since all rotations are about the Z-axis:

$$\omega = \left[0, 0, \dot{\theta}_1 + \dot{\theta}_2 + \dot{\theta}_4\right]^T$$

Therefore:

$$J_o = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

3. Combine J_p and J_o :

Using the partial derivatives calculated above, we construct the full analytical Jacobian:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial d_3} & \frac{\partial x}{\partial \theta_4} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial d_3} & \frac{\partial y}{\partial \theta_4} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial d_3} & \frac{\partial z}{\partial \theta_4} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Substituting the partial derivatives:

$$J = \begin{bmatrix} -a_1 \sin(\theta_1) - a_2 \sin(\theta_1 + \theta_2) & -a_2 \sin(\theta_1 + \theta_2) & 0 & 0 \\ a_1 \cos(\theta_1) + a_2 \cos(\theta_1 + \theta_2) & a_2 \cos(\theta_1 + \theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

Chapter 4

Trajectory Planning