UFMF4X-15-M Robotic Fundamentals Coursework

Serial and Parallel Robot Kinematics Report

Name: Jiajun Guo 24065271

Name: Dongqing Shao 24065119

Robotics Msc

Engineering Design and Mathematics
University of the West of England, Bristol

January 16, 2025

Contents

1 PART I

1.1 A1 - FK

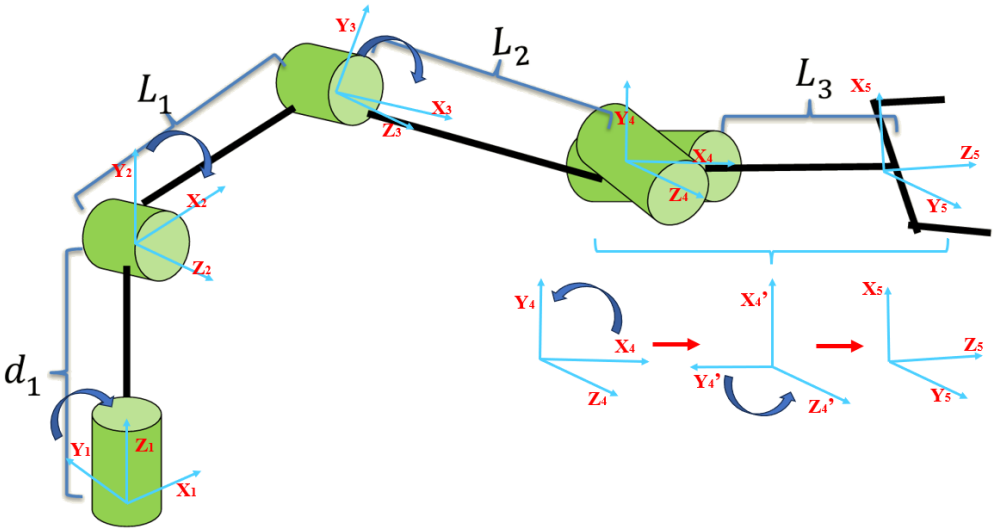


Figure 1.1: DH Transformation schema

The Proximal Denavit-Hartenberg(DH) parameter table is obtained based on the Lynxmotion Robot structure diagram as follows:

Table 1.1: Proximal DH Convention

n	a_{n-1}	α_{n-1}	d_n	θ_n
1	0	0	0	θ_1
2	0	90°	0	θ_2
3	L_1	0	0	θ_3
4	L_2	0	0	$\theta_4(+\frac{pi}{2})$
5	0	90°	0	θ_5

where

- a_{n-1}: the distance from z_{n-1} to z_n measured along x_{n-1}.
- α_{n-1}: the angle between z_{n-1} to z_n measured about x_{n-1}.
- d_n: the distance from x_{n-1} to x_n measured along z_n.
- θ_n: the angle between x_{n-1} to x_n measured about z_n.

As shown in Table1.1, the d_1 and L_3 parameters are missing, hence we have to use the distal DH convention. The distal DH parameter table is shown as follows:

Table 1.2: Distal DH Convention

n	a_n	α_n	d_n	θ_n
1	0	90°	d_1	θ_1
2	L_1	0	0	θ_2
3	L_2	0	0	θ_3
4	0	90°	0	θ_4
5	0	0	L_3	θ_5

where

- a_n: the distance from z_{n-1} to z_n measured along x_n.
- α_n: the angle between z_{n-1} to z_n measured about x_n.
- d_n: the distance from x_{n-1} to x_n measured along z_{n-1}.
- θ_n: the angle between x_{n-1} to x_n measured about z_{n-1}.

Use the distal DH parameters to derive the transformation matrix for each link, ${}^{n}_{n-1}T$: The general form of the matrix is as follows:

$$ {}^{n}_{n-1}T = \begin{bmatrix} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & a_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & a_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.1} $$

Then, Compute the overall transformation matrix by:

$$ {}^{0}_{5}T = {}^{0}_{1}T \cdot {}^{1}_{2}T \cdot {}^{2}_{3}T \cdot {}^{3}_{4}T \cdot {}^{4}_{5}T \tag{1.2} $$

and implement the calculations and simulations in MATLAB. Input the lengths (d_1, L_1, L_2, L_3) and the joint angles ($\theta_1, \theta_2, \theta_3, \theta_4, \theta_5$) and output the 3D Cartesian coordinates of the end-effector. The parameters of the Lynxmotion robot arm are assumed as follows:

Table 1.3: The parameters of the Lynxmotion robot arm

Parameters	Value	Explanation
d_1	0.1	Distance from the first joint to the second joint
L_1	0.2	Length of the link from the second joint to the third joint
L_2	0.2	Length of the link from the third joint to the fourth joint
L_3	0.1	Distance from the fifth joint to the gripper
θ_1	60°	The rotation angle of the first joint (base joint)
θ_2	45°	The rotation angle of the second joint
θ_3	-30°	The rotation angle of the third joint
θ_4	60°	The rotation angle of the fourth joint
θ_5	30°	The rotation angle of the 5th joint (end-effector joint)

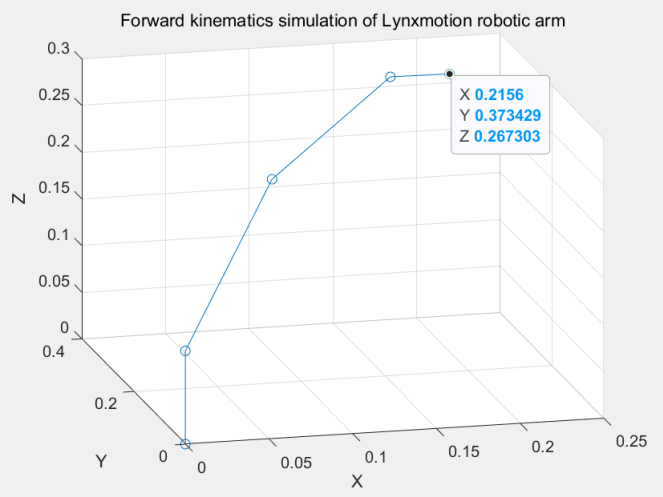
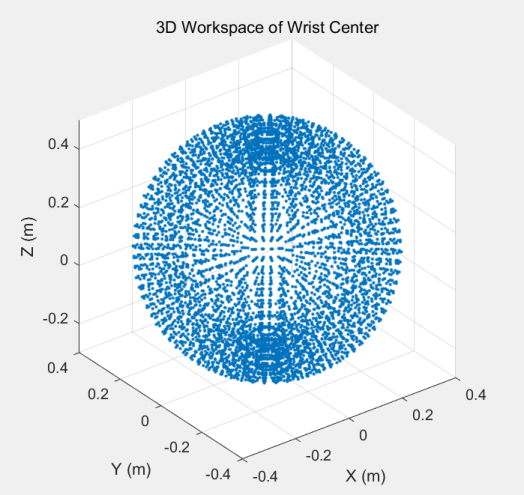


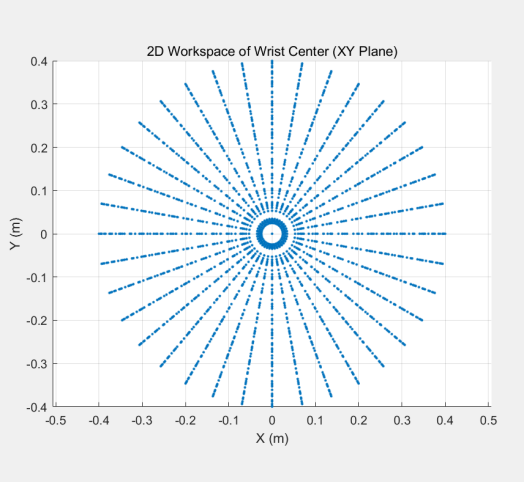
Figure 1.2: Forward kinematics simulation of Lynxmotion robotic arm

Figure 1.2 shows the results of the MATLAB operation, which describe the position and orientation of the end-effector.

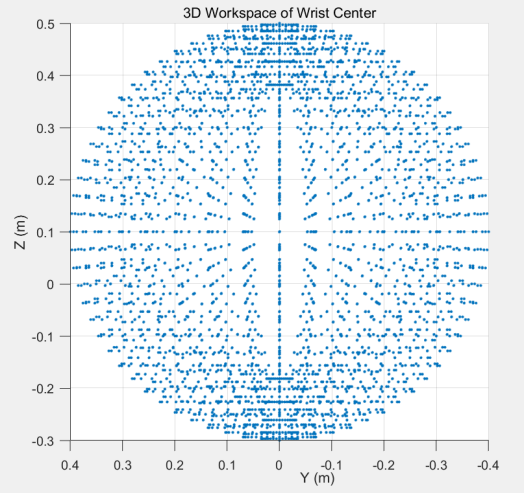
1.2 A2 - Workspace



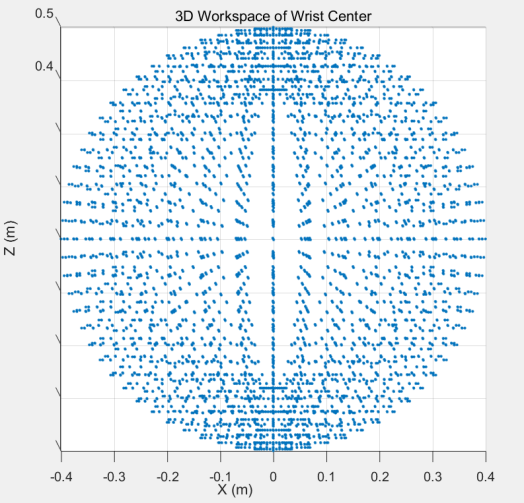
(a) 3D workspace of wrist center



(b) 2D workspace of wrist center XY plane



(c) 2D workspace of wrist center YZ plane



(d) 2D workspace of wrist center XZ plane

Figure 1.3: The workspace of the center of the wrist

The size of the workspace depends on the total length of the link and the extent of the joint. Since θ_5 does not affect the position of the fifth joint and only affects the direction of the end effector, the workspace can be obtained by traversing the motion range of the first four joints in MATLAB and calculating the center position of the fifth joint using forward kinematics. In addition, considering the actual physical limitations, the Angle range of joint 1 and joint 4 of the robotic arm is set from 0°to 360°, and the Angle range of joint 2 and joint 3 is assumed to be -90°to 90°. Furthermore, due to the excessive number of combinations when the step size of the calculation is set to 1°, more than 4 billion workspace points need to be calculated, Matlab also displays an error "The requested 4269446281x3

(95.4GB) array exceeds the preset maximum array size (15.6GB)". Therefore, the step size of the joint Angle range is changed to 10°instead of 1°to reduce the number of iterations.

Figure 1.3a is a point cloud diagram of a 3D workspace drawn using scatter3 in MATLAB, showing the complete 3D workspace in the center of the wrist. The workspace is shaped like a spherical shell, and the arm cannot reach the position directly above or below the base due to the physical limitations of the link length and joint range. Another Figures show three 2D projection of the workspace in the XY, YZ and XZ planes. The dense inner ring corresponds to the position near the base of the arm, while the outer point represents the extended position of the arm when it is fully extended. The robotic arm has a distinct cylindrical extension in the horizontal plane, determined by the maximum connecting rod length of the arm $L_1 + L_2 + L_3$. The gap in the two figures, especially the radial line gap in the XY plane, is caused by the chosen joint angle step of 10°.

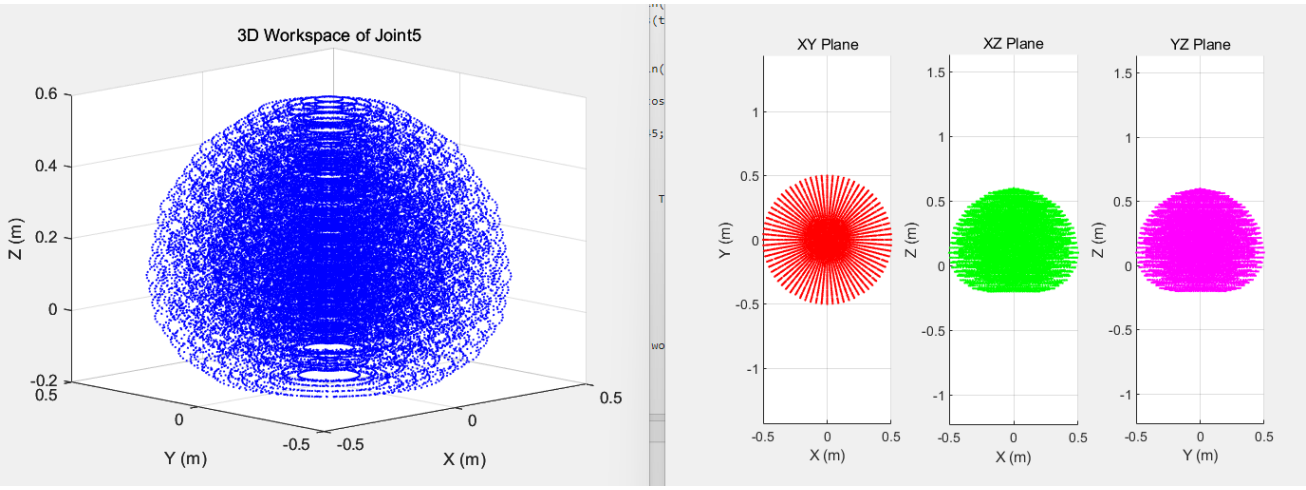


Figure 1.4: Workspace of the end effector with limitations as the real environment

According to Fig. 1.1, a revised constraint has been proposed for joint 2. This is because the X-axis is horizontal as the initial state of the joint, and in reality it can only move to one side, which is [0°, 180°]. This is how the figure 1.4 was obtained. It can be seen that the limitation of each joint can cause large differences between two workspaces.

1.3 A3 - IK

The inverse kinematics model of the manipulator solves the angle or position of each joint of the manipulator by giving the target position $P(x,y,z)$ of the end-effector. Because only the joint Angle needed to reach the target position is solved, the rotation direction of the end-effector (θ_5) is ignored. In addition, if the Target point is out of the arm's workspace, an error will be reported: "Target position is out of reach."

The steps to solve the inverse kinematics model are as follows:

1. The base rotation angle θ_1 is determined by the projection of the target position onto the XY-plane:
$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) \tag{1.3}$$
2. Calculate horizontal distance r and height difference h:
$$r = \sqrt{x^2+y^2}, \quad h = z - d_1 \tag{1.4}$$
3. Calculate θ_3, using trigonometry and the law of cosines:
$$D = \frac{r^2+h^2-L_1^2-L_2^2}{2L_1L_2} \tag{1.5}$$
$$\theta_3 = \tan^{-1}\left(\sqrt{1-D^2}, D\right) \tag{1.6}$$
4. Calculate θ_2:
$$\theta_2 = \tan^{-1}(h,r) - \tan^{-1}\left(L_2\sin(\theta_3), L_1 + L_2\cos(\theta_3)\right) \tag{1.7}$$

Step 3: Calculate Wrist Joint Angles (θ_4, θ_5)

To compute the remaining wrist joint angles, the orientation of the wrist base relative to the target orientation R_{target} need to be determined.

1. Using the forward kinematics model, calculate the rotation matrix of the wrist base R_{wrist} by multiplying the transformation matrices for θ_1, θ_2 and θ_3:
$$R_{\text{wrist}} = T_1 \cdot T_2 \cdot T_3 \tag{1.8}$$
2. The relative orientation of the wrist $R_{\text{wrist\_remaining}}$ is given by:
$$R_{\text{wrist\_remaining}} = R_{\text{wrist}}^T \cdot R_{\text{target}} \tag{1.9}$$
3. The wrist angles θ_4 and θ_5 are then extracted from $R_{\text{wrist\_remaining}}$:
$$\theta_4 = \tan^{-1}\left(\frac{R_{\text{wrist\_remaining}}(2,3)}{R_{\text{wrist\_remaining}}(1,3)}\right) \tag{1.10}$$
$$\theta_5 = \tan^{-1}\left(\sqrt{R_{\text{wrist\_remaining}}(1,3)^2 + R_{\text{wrist\_remaining}}(2,3)^2}, R_{\text{wrist\_remaining}}(3,3)\right) \tag{1.11}$$

where (1,3) represents the elements of row 1 and column 3, and (2,3) represents the elements of row 2 and column 3.

1.3.1 Verification of Results

```
命令行窗口
>> A3_IK
Computed joint angles (in radians):
    0.7854    0.0919    1.0472    0.4317    1.5708

Computed joint angles (in degrees):
   45.0000    5.2644   60.0000   24.7356   90.0000

Verification Results:
Target Position:
    0.2000    0.2000    0.3000

Computed Position:
    0.2707    0.2707    0.3000

Position Error: 0.1 meters
Target Orientation (R_target):
     1     0     0
     0     1     0
     0     0     1

Computed Orientation (R_computed):
    0.7071    0.0000    0.7071
   -0.7071    0.0000    0.7071
    0.0000   -1.0000    0.0000

Orientation Error: 2.1414
```

Figure 1.5: IK Verification Results

The computed joint angles are verified using forward kinematics. The final position and orientation of the end-effector are compared to the target position and orientation. The verification results are shown in Figure1.5.

The position error of 0.1 meters indicates that the numerical inverse kinematics approach provides a relatively accurate solution for the target position. However, the orientation error of 2.1414 measured using the Frobenius norm, highlights a significant deviation from the desired target orientation. This could be due to the limitations in the joint ranges or mechanical constraints of the Lynxmotion arm. On the other hand, numerical optimization convergence issues may require refining the initial guesses or optimization algorithm parameters.

1.3.2 The IK with a simplified orientation

Because of the joint 5 is a revolutionary joint, thus the axis of rotation is aligned with the direction of the connecting rod. There are no joints other than joint 1 that cause the end effector to rotate in the X-Y plane of the base coordinate system. Therefore, after obtaining the value of joint 1, the remaining arm can be simplified to a planar three-link arm.

Based on these, it can be assumed that the position of the end effector is $P(x,y,z)$. Only position is not enough because it cannot solve the equation containing four unknown variables. Thus, the orientation of the end effector can be simplified as only turning on the X-Z plane of the base coordinate system,

and it's angle with X-Y plane of the base coordinate system was set as:

$$\varphi = \theta_2 + \theta_3 + \theta_4 \tag{1.12}$$

Thus, the transformation matrix should be:

$$T = \begin{bmatrix} cos(\varphi) & 0 & sin(\varphi) & x \\ 0 & 1 & 0 & y \\ -sin(\varphi) & 0 & cos(\varphi) & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.13}$$

First, we can solve θ_1 using:

$$\theta_1 = arctan(\frac{y}{x}) \tag{1.14}$$

Then, the following equation can be derived:

$$l_1c_2 + l_2c_{23} + l_3c_{234} = \frac{x}{c_1} \tag{1.15}$$

$$l_1s_2 + l_2s_{23} + l_3s_{234} = z - d_1 \tag{1.16}$$

$$P_{\text{joint 4x}} = x/c_1 - l_3c_{234} = l_1c_2 + l_2c_{23} \tag{1.17}$$

$$P_{\text{joint 4y}} = z - d_1 - l_3s_{234} = l_1s_2 + l_2s_{23} \tag{1.18}$$

$1.17^2 + 1.18^2$:

$$c3 = \frac{P^2_{\text{joint 4x}} + P^2_{\text{joint 4y}} - l_1^2 - l_2^2}{2l_1l_2} \tag{1.19}$$

Therefore,

$$\theta_3 = arccos(c3) \tag{1.20}$$

Take it into 1.17 and 1.18, then:

$$s2 = \frac{(a_1 + a_2cos\theta_3)P_{\text{joint 4y}} - a_2sin\theta_3P_{\text{joint 4x}}}{P^2_{\text{joint 4x}} + P^2_{\text{joint 4y}}} \tag{1.21}$$

$$c2 = \frac{(a_1 + a_2cos\theta_3)P_{\text{joint 4x}} - a_2sin\theta_3P_{\text{joint 4y}}}{P^2_{\text{joint 4x}} + P^2_{\text{joint 4y}}} \tag{1.22}$$

Thus:

$$\theta_2 = arctan(s2, c2) \tag{1.23}$$

$$\theta_4 = \varphi - \theta_3 - \theta_2 \tag{1.24}$$

```
>> p1a3_final_code
theta1 = 0.59 rad
theta2 = 0.00 rad
theta3 = 1.10 rad
theta4 = -0.32 rad
Input: x is 0.30, y is 0.200, and z is 0.350.
Output:x is 0.30, y is 0.200, and z is 0.350.
``
```

Figure 1.6: The results of this IK method

1.4 B1-Task planning

This research plans to design a simplified robotics arm for gripping items from one conveyor belt to the other. Based on the obtained simplified inverse kinematics, multiple extreme positions were tested with several end effector orientations, verifying the accuracy of the algorithm and model.

Table 1.4: Position and orientation set for the end effector

n	x	y	z	ϕ
a	0.4	0.1	0.2	0
b	0.5	0.0	0.1	0
c	0.0	0.0	0.6	$\frac{\pi}{2}$
d	0.1	0.4	0.3	$\frac{\pi}{4}$
e	-0.1	-0.4	0.3	$\frac{\pi}{4}$
f	0.32	0.0	0.2	$-\frac{\pi}{2}$

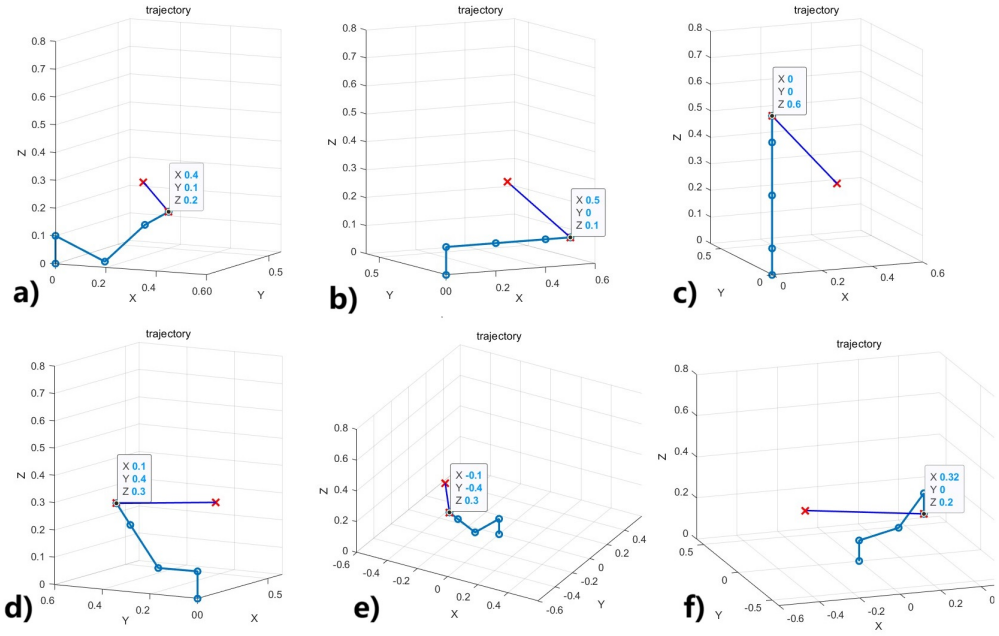


Figure 1.7: Simulation of positions and orientations of the end effector

1.5 B2-Animation of the planned task

This study intends to simulate the completion of the movement of the robotic arm for handling objects, so five points were redesigned for this purpose. At the start and end points, the end effector of the robotic arm needs to be down along the Z-axis. At points 2 and 4, the arm is kept horizontal, while at point 3, the end effector is up along the Z-axis. However, the carried objects and the conveyor belt were simplified in this experiment and will be added when there is enough time.

Table 1.5: Position and orientation set for the end effector

n	x	y	z	ϕ
a	0.32	0.0	0.2	$-\frac{\pi}{2}$
b	0.2	0.0	0.4	0
c	0.0	0.0	0.6	$\frac{\pi}{2}$
d	-0.2	0.0	0.4	0
e	-0.32	-0.0	0.2	$-\frac{\pi}{2}$

The model is simplified because the axis of rotation at the end effector coincides with the final linkage (as discussed in 1.3.2). The attitude matrix T of the end effector can be expressed as a simplified model multiplied by a rotation matrix rotating around the Z-axis at joint 5.

In this section, the essential part of the problem is then turning into obtain the sets of joint coordinates.

From 1.3.2, we can get values of $[\theta_1, \theta_2, \theta_3, \theta_4]$ from positions and orientations set for the end effector.

Then, positions of each joint can be calculated:

$$P_{5x} = (l_1 c_2 + l_2 c_{23} + l_3 c_{234}) c_1 \tag{1.25}$$

$$P_{5y} = (l_1 c_2 + l_2 c_{23} + l_3 c_{234}) s_1 \tag{1.26}$$

$$P_{5z} = l_1 s_2 + l_2 s_{23} + l_3 s_{234} + d_1 \tag{1.27}$$

$$P_5 = [P_{5x}, P_{5y}, P_{5z}] \tag{1.28}$$

Similarly,

$$P_{4x} = (l_1 c_2 + l_2 c_{23}) c_1 \tag{1.29}$$

$$P_{4y} = (l_1 c_2 + l_2 c_{23}) s_1 \tag{1.30}$$

$$P_{4z} = l_1 s_2 + l_2 s_{23} + d_1 \tag{1.31}$$

$$P_4 = [P_{4x}, P_{4y}, P_{4z}] \tag{1.32}$$

$$P_{3x} = (l_1 c_2) c_1 \tag{1.33}$$

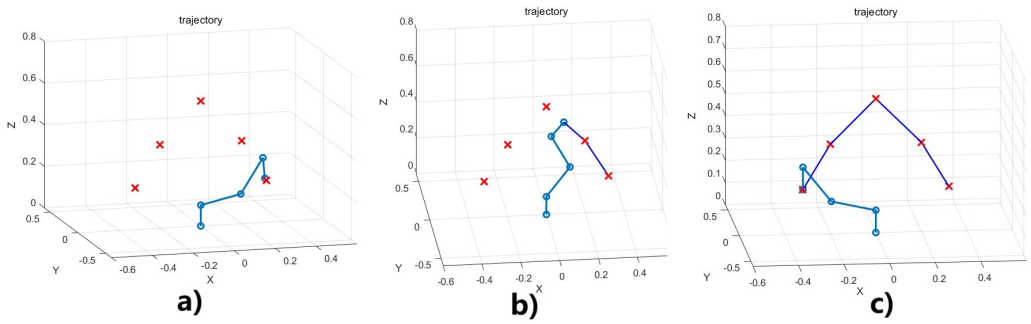


Figure 1.8: Screenshot of animation in the assigned task

$$P_{3y} = (l_1 c_2) s_1 \tag{1.34}$$

$$P_{3z} = l_1 s_2 + d_1 \tag{1.35}$$

$$P_3 = [P_{3x}, P_{3y}, P_{3z}] \tag{1.36}$$

$$P_2 = [0, 0, d_1] \tag{1.37}$$

$$P_1 = [0, 0, 0] \tag{1.38}$$

Connecting these positions of joints, the inverse kinematics can be clearly shown. Using interpolated points and pause in matlab, the animation of the motion is shown in Fig 1.8.

After this, an image of a star (Fig. 1.9) was drawn to ensure it's performance (leisure).

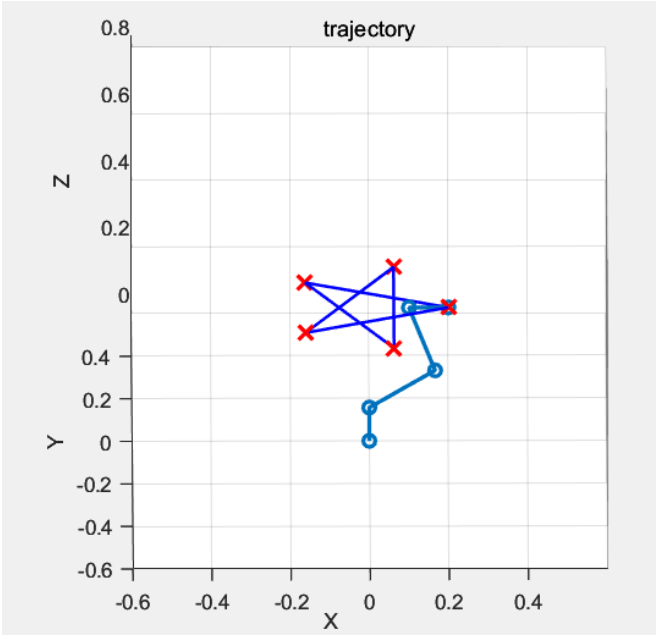


Figure 1.9: A star drawn by the end effector

1.6 B3-Trajectories between points

To visualise the different movement trajectories. This research combines free movement, linear movement and obstacle avoidance in a hypothetical set of points mentioned in Table 1.5. In this case, free movement from point 1 to 2 follows a sinusoidal function. Linear motion is required from 2 to 3 and obstacle avoidance is required from 3 to 4. The obstacle is a sphere with its centre at [-0.2,0.0,0.6] and a radius of 0.15 (shown in Fig. 1.10).

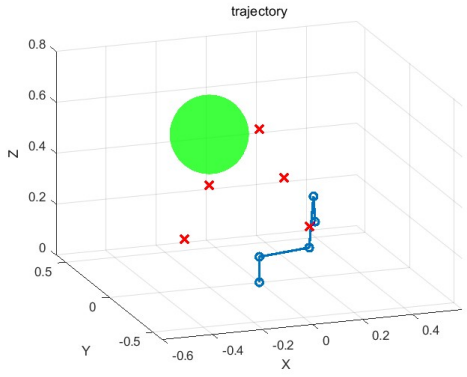


Figure 1.10: Settings of the experimental environment

It is easy to complete the task of free movement and the linear motion (Fig. 1.11). However, the obstacle avoidance movement is especially tough. At first, spline interpolation was used for this trajectory. But the parameters are hard to adjust. Thus, another predict-based trajectory was proposed. Essentially, it's splitting a path into two segments.

The vectors of the line between two points and the vector of the line connecting a point on the line to the obstacle are cross-multiplied, in order to obtain the distance from the center of the obstacle to the line. Comparing this distance to the radius of the obstacle can make sure that the route is safe or not. If an obstacle exists, the end effector should make a linear motion in the for loop until the path between itself and the endpoint is clear. Then a linear motion to the endpoint is made (shown as Fig. 1.12).

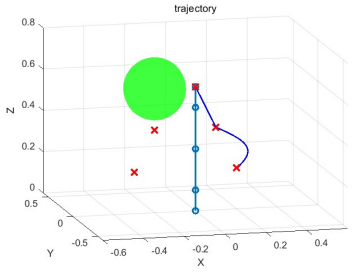


Figure 1.11: A two-in-one trajectory

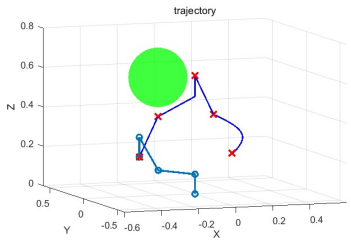


Figure 1.12: The proposed obstacle avoidance trajectory

2 PART II

2.1 IK

This section presents the implementation and simulation of a planar parallel robot, focusing on solving the inverse kinematics for varying platform positions and orientations. The study includes deriving mathematical models, implementing them in MATLAB, and analyzing the results through graphical simulations. The process of solving and simulating the planar parallel robot's kinematics involves the following steps.

2.1.1 Define robot parameters

Table 2.1: Design parameters of planar parallel robot

Parameter – Data Geometry	**Value**
S_A	170mm
L	130mm
r_{platform} (joint circle radius)	130mm
r_{base} (joint circle radius)	290mm

Table 2.2: Angular joint positions

PB_1	$\pi/2$
PB_2	$\pi+\pi/6$
PB_3	$2\pi-\pi/6$
PP_1	$\pi/2$
PP_2	$\pi+\pi/6$
PP_3	$2\pi-\pi/6$

Initialise design parameters such as link lengths and radii of the platform and base, which are shown in Table2.1 and Table2.2.

2.1.2 Set platform input parameters

Table 2.3: Multiple Cartesian input parameters

Position	X_c	Y_c	α
1	0	100	0°
2	50	150	30°
3	-50	50	-30°
4	100	200	45°

Table2.3 gives the specified Cartesian coordinates (X_c, Y_c) and orientation angle α of the center of the platform where the pointer is located, that is the needle position.

2.1.3 Calculate the joint coordinates of the base and platform

Rotation matrices are used to compute the positions of the three platform joints. The platform's center is located at (X_c, Y_c) with an orientation α. The position of the i-th platform joint is calculated as:

$$\begin{bmatrix} x_{PP_i} \\ y_{PP_i} \end{bmatrix} = R_{BC} \begin{bmatrix} r_{\text{platform}} \cos\left(\frac{2\pi(i-1)}{3}\right) \\ r_{\text{platform}} \sin\left(\frac{2\pi(i-1)}{3}\right) \end{bmatrix} + \begin{bmatrix} X_c \\ Y_c \end{bmatrix}, \tag{2.1}$$

where R_{BC} is the rotation matrix:

$$R_{BC} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}. \tag{2.2}$$

2.1.4 Solve inverse kinematics

Derive joint angles for each leg by solving quadratic equations. For each leg, the relationship between the base joint, platform joint, and intermediate joint is defined by the loop closure equation:

$$x_{PP_i} = r_S A\cos(\theta) + L\cos(\psi) \tag{2.3}$$

$$y_{PP_i} = r_S A\sin(\theta) + L\sin(\psi) \tag{2.4}$$

$$(x_{PP_i} - x_{PB_i})^2 + (y_{PP_i} - y_{PB_i})^2 = L^2. \tag{2.5}$$

Rearranging, we define intermediate parameters:

$$e_1 = -2y_{PP_i}S_A, \tag{2.6}$$

$$e_2 = -2x_{PP_i}S_A, \tag{2.7}$$

$$e_3 = x_{PP_i}^2 + y_{PP_i}^2 + S_A^2 - L^2. \tag{2.8}$$

Simplified as:

$$e_1 s_\theta + e_2 c_\theta + e_3 = 0 \tag{2.9}$$

The quadratic equation for $\tan(\theta_i/2)$ becomes:

$$(e_3 - e_2)t^2 + 2e_1 t + (e_3 + e_2) = 0. \tag{2.10}$$

Solving this yields:

$$t_{1,2} = \frac{-e_1 \pm \sqrt{e_1^2 - (e_3 - e_2)(e_3 + e_2)}}{e_3 - e_2}. \tag{2.11}$$

The joint angle is then computed as:

$$\theta_i = 2\tan^{-1}(t). \tag{2.12}$$

Then implement the above equations in MATLAB to simulate the robot's behavior.

2.1.5 Simulate and visualise

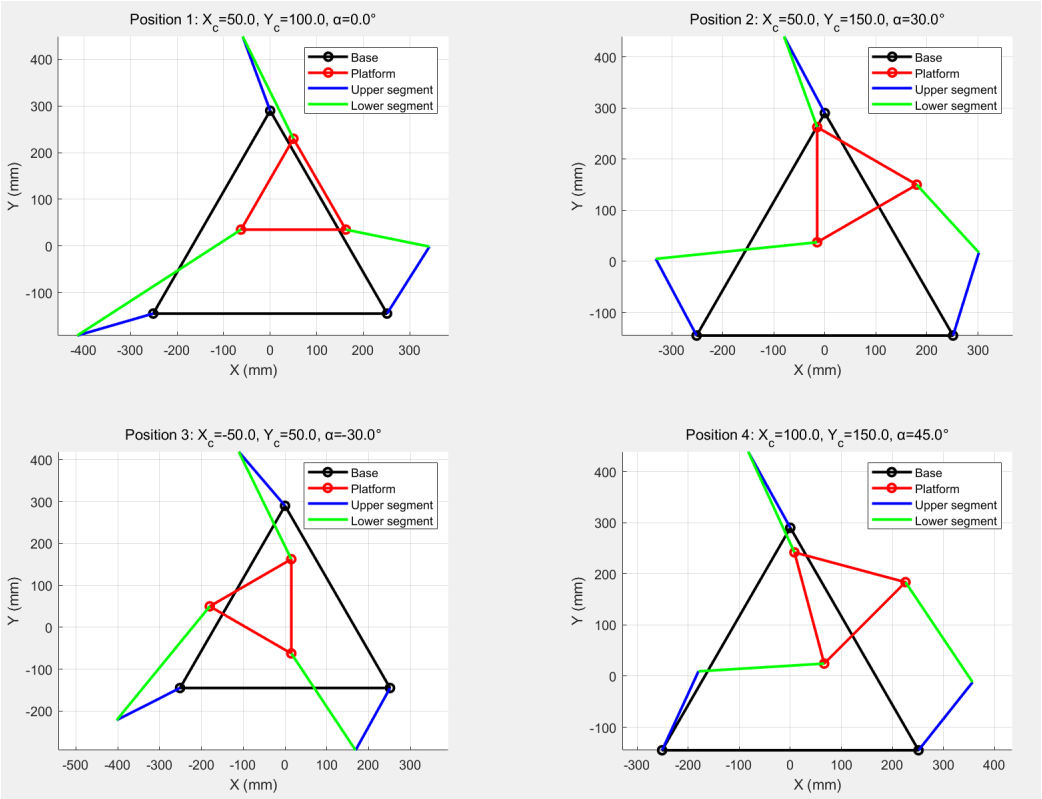


Figure 2.1: Planar Parallel Robot Configurations for Different Input Parameters

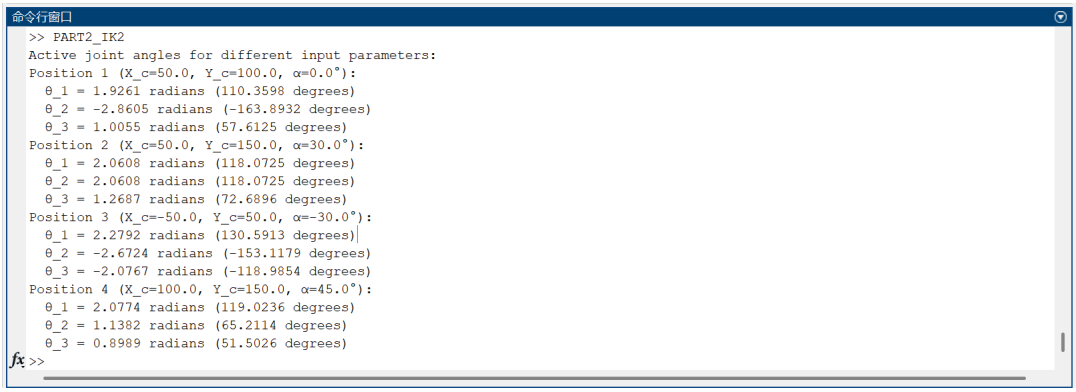


Figure 2.2: The results of each active angle

Table 2.4: Computed Active Joint Angles for Different Platform Positions

Position	θ_1 (deg)	θ_2 (deg)	θ_3 (deg)
1	110.36	-163.893	57.613
2	118.073	118.073	72.69
3	130.591	-153.118	-118.985
4	119.024	65.211	51.503

Matlab plots the robot configuration for multiple platform positions and compares results, and Matlab also outputs the active joint angles of radians and angles with different input parameters. Figure 2.1 shows the robot configurations, highlighting how each leg adapts to the new platform positions and orientations, and Figure2.2 also shows each leg's active joint angles were computed and presented in Table 2.4. After retaining three decimal places for the results, the calculation results of the active joint Angle under different parameters are shown in Table2.4.

The simulation demonstrates that the robot successfully adapts to varying platform positions and orientations. The computed joint angles match expectations, ensuring smooth and feasible configurations. The visualizations further confirm the correctness of the kinematic model.

2.2 Workspace

The task involves plotting the robot's workspace for varying platform orientations α to analyse the workspace of a planar parallel robot used in surgical procedures. The workspace analysis provides a reference for the robot's motion capabilities and constraints, which are important for robot design and safe operation in human environments.

The workspace of a planar parallel robot is the set of all reachable positions and orientations of the

platform center (X_c, Y_c, α), where α is the orientation of the platform. The feasibility of a point in the workspace is determined by solving the inverse kinematics equations and verifying if the joint angles satisfy the robot's geometric and kinematic constraints. For each platform position (X_c, Y_c, α), the joint angles θ_i $(i = 1, 2, 3)$ are determined by solving the loop closure equations:

$$(x_{PP_i} - x_{PB_i})^2 + (y_{PP_i} - y_{PB_i})^2 = L^2, \tag{2.13}$$

where x_{PP_i}, y_{PP_i} are the platform joint positions, x_{PB_i}, y_{PB_i} are the base joint positions, and L is the length of the lower link. The discriminant of the resulting quadratic equation determines whether a real solution exists. For 3D workspace analysis, vary (X_c, Y_c) over a grid and α over a range of orientations. Then solve the inverse kinematics equations for each (X_c, Y_c, α) to determine feasibility, and mark feasible points in the workspace based on the existence of real solutions. For 2D workspace analysis, fix α to specific values, $0°, 30°, 45°, 60°$ and repeat the process.

2.2.1 Results and Analysis

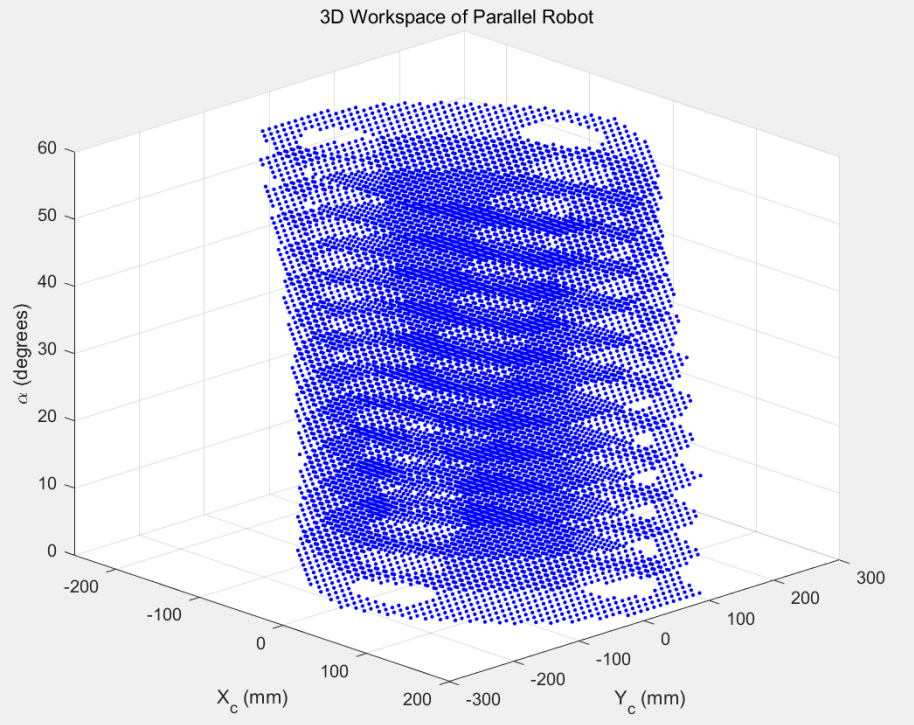


Figure 2.3: 3D Workspace of the Planar Parallel Robot

The 3D workspace of the robot, showing the relationship between X_c, Y_c, and α, is presented in Figure 2.3, which shows the robot's ability to achieve different platform orientations at varying positions within the workspace.

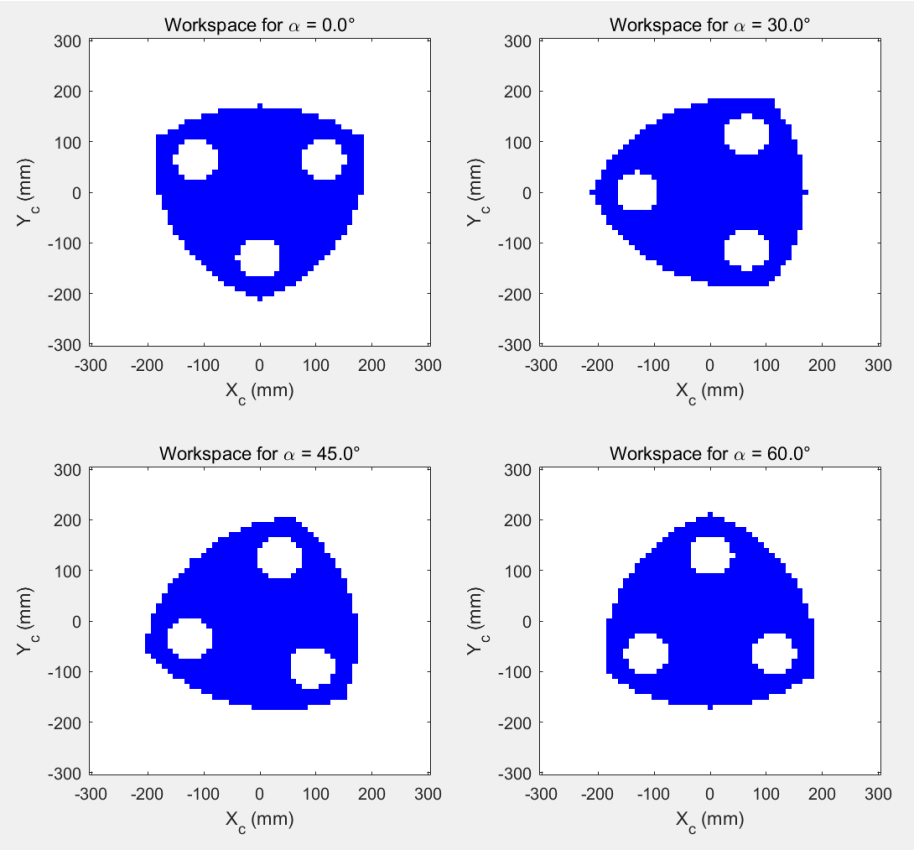


Figure 2.4: 2D workspace of Parallel Robot with different α

The 2D workspace for fixed platform orientations $\alpha = 0°, 30°, 45°, 60°$ is shown in Figure 2.4. Each plot represents the set of reachable positions (X_c, Y_c) for the given orientation. The results indicate that the robot's workspace is bounded by the geometry of its links and platform dimensions, resulting in a triangular-like shape with voids. In addition, internal voids correspond to singular configurations or points where the inverse kinematics equations have no real solutions, which marks potential limitations in the robot's design. Increasing the orientation α can shift the workspace and slightly alter its shape illustrating the importance of orientation in mechanism synthesis.

3 PART III
Singularity Avoidance Using Screw Theory

3.1 Introduction

The analysis and avoidance of singularities are critical challenges in robotic manipulator design and control. Singular configurations often result in the loss of control, undefined forces, or velocities, leading to undesirable system behaviors. Singularities occur when the Jacobian matrix, which relates joint velocities to end-effector velocities, becomes rank-deficient, causing the manipulator to lose one or more degrees of freedom in task space. Screw theory has been widely applied in the analysis of singular configurations in parallel robots. Early work by Hao and Ding (2006) introduced a general method for analyzing the relationships between different criteria of singular configurations using screw theory, and showed the importance of second-order criteria in determining bifurcated singular configurations. By leveraging screw theory, kinematic models of various planar parallel mechanisms can be derived, enabling singularity analysis. Furthermore, optimal design methods for parallel mechanisms that consider singularity locations can achieve larger workspaces (Kang and Kim, 2012).

Jacobian matrix and singularity analysis play a critical role in robotic motion analysis. When the determinant of the matrix equals zero, the robot is in a singular configuration, where certain directions of task-space motion cannot be achieved through joint motion. Using screw theory, the Jacobian matrix of planar parallel robots can be derived (Choi, Lee, and Lee, 2012). Guo et al. (2013) introduced a novel 4-RRCR parallel mechanism based on screw theory and employed a recursive elimination method to solve the Jacobian matrix, conducting kinematic and singularity analysis. Additionally, screw theory can be applied to the kinematic analysis of Delta robots, resulting in closed-form solutions for displacement analysis and demonstrating its effectiveness in detecting bifurcated singularities (Gallardo-Alvarado, Balmaceda-Santamaría, and Castillo-Castaneda, 2014). In subsequent studies, these techniques have been extended to more complex robotic systems, including the 6-UCU parallel manipulator and the PRoM-120 parallel robot based on 2-PRU/PRS kinematic chains, where inverse kinematics and velocity analysis were conducted using loop vector equations, which verified these proposed optimizations (Liu et al., 2014; Adriyan and Sufiyanto, 2018; Kang and Kim, 2012).

Despite these advances, practical challenges persist in integrating singularity avoidance strategies into real-world robotic applications. This study builds upon existing literature by implementing a Jacobian-

based singularity detection method combined with a simple path planning algorithm to demonstrate singularity avoidance for a robotic manipulator. The aim is to visualize singular configurations and design trajectories that circumvent these critical regions.

3.2 Methodology and Implementation

The robotic arm used in this study is modeled with four degrees of freedom. The physical parameters of the robotic arm are defined as follows:

- Link lengths (a): [0, 10, 10, 0]
- Twist angles (α): [$\pi/2$, 0, 0, 0]
- Offsets (d): [5, 0, 0, 0]
- Joint variables (θ): [$\theta_1, \theta_2, \theta_3, \theta_4$]

The corresponding Denavit-Hartenberg (DH) table is provided below:

Table 3.1: DH Table for the Robotic Arm

Joint	a_i (**Link Length**)	α_i (**Twist Angle**)	d_i (**Offset**)	θ_i (**Joint Variable**)
1	0	$\pi/2$	5	θ_1
2	10	0	0	θ_2
3	10	0	0	θ_3
4	0	0	0	θ_4

The Jacobian matrix is crucial for analyzing the motion and singularities of the robotic arm. The DH transformation matrix for each link is defined as:

$$T_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

The Jacobian matrix is composed of linear velocity (J_v) and angular velocity (J_w) components for each joint. These are calculated as:

$$J_v = \mathbf{z}_i \times (\mathbf{p}_{\text{end}} - \mathbf{p}_i), \quad J_w = \mathbf{z}_i \tag{3.2}$$

The complete Jacobian matrix is then given as:

$$J = \begin{bmatrix} J_{v_1} & J_{v_2} & J_{v_3} & J_{v_4} \\ J_{w_1} & J_{w_2} & J_{w_3} & J_{w_4} \end{bmatrix} \tag{3.3}$$

By using Denavit-Hartenberg (DH) convention to model the kinematics of a robotic manipulator with four degrees of freedom. The forward kinematics are derived to compute the manipulator's position and orientation in task space. Then Jacobian matrix is calculated for singularity analysis by derivation of forward kinematics of joint variables. Since the singularity is identified where the determinant approaches zero, in order to detect the singular configuration, the determinant of the position component of the Jacobian matrix is found on a series of joint configurations. For path planning, path points are defined in the task space and linear trajectories are generated between these points. The trajectory is then verified against the singular graph to ensure safe operation.

To validate the model and analyse the path of the end-effector, A simple linear trajectory is generated to connect these waypoints, the following waypoints are defined in task space:

$$\text{Waypoints: } \begin{bmatrix} 0 & 0 & 5 \\ 10 & 0 & 10 \\ 15 & 5 & 10 \end{bmatrix} \tag{3.4}$$

Specifically, the DH parameters of the manipulator are first defined according to the physical geometry, including link lengths, twists, and offsets, and then the Jacobian matrix is calculated using MATLAB and the singular analysis is visualised. The Jacobian determinant is calculated for varying values of the first joint angle (θ_1), with other joints fixed at zero for simplicity. This analysis provides a singularity detection plot, highlighting configurations where the determinant is near zero. Additionally, a path planning algorithm is implemented to connect three task-space waypoints: $[0,0,5]$, $[10,0,10]$, and $[15,5,10]$. The generated trajectory is visualised in three-dimensional space.

3.3 Results

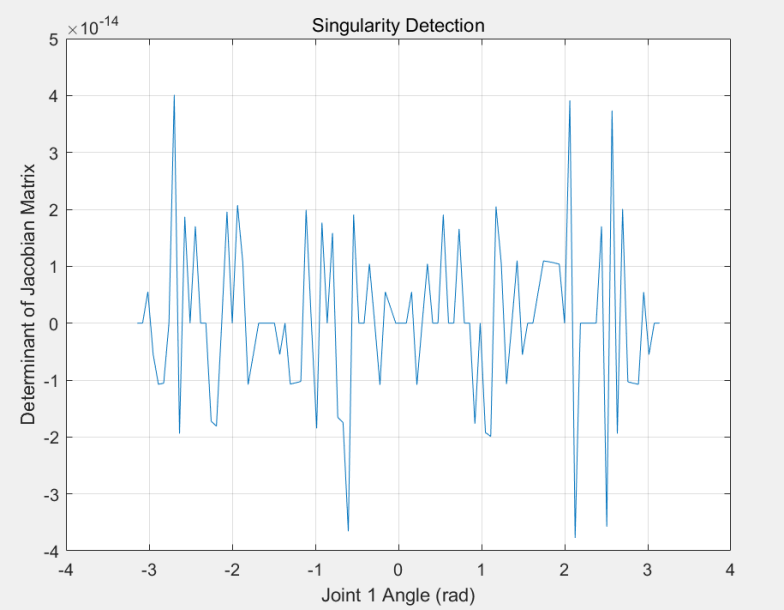


Figure 3.1: Singularity detection plot

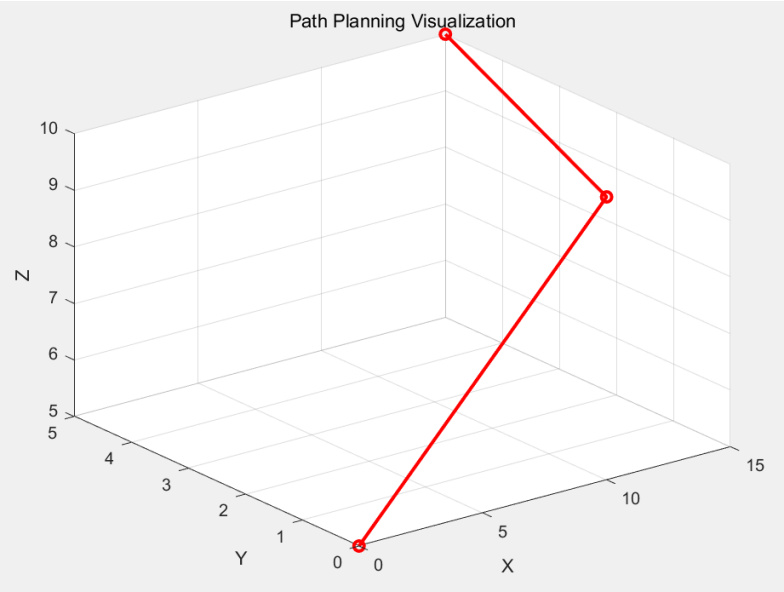


Figure 3.2: Path planning visualization

The singularity detection results are shown in Figure 3.1, where the determinant of the Jacobian matrix is plotted against the first joint angle. Peaks and troughs in the graph correspond to configurations of maximum and minimum controllability, respectively. Singular configurations are observed at points

where the determinant approaches zero. This behavior indicates a loss of degrees of freedom, validating the necessity of avoiding such configurations during operation.

The path planning results are depicted in Figure 3.2, showing a smooth linear trajectory connecting the three waypoints. This trajectory successfully avoids singular configurations, as validated by the singularity detection plot. The results demonstrate the efficacy of integrating singularity analysis into path planning, ensuring both safety and task completion.

3.4 Discussion

The theoretical foundation of this study is rooted in screw theory and Jacobian analysis, which provide a systematic approach for understanding robotic kinematics and singularities. Screw theory enables the representation of both linear and angular velocities in a unified framework, making it an essential tool for analyzing the motion capabilities of manipulators. By leveraging the Denavit-Hartenberg (DH) convention, the forward kinematics and the Jacobian matrix of the manipulator were derived, facilitating the detection of singular configurations. The results validate the theoretical prediction that singularities occur when the Jacobian matrix becomes rank-deficient, , represent configurations where the manipulator loses one or more degrees of freedom in task space. This can lead to control instability and unpredictable motion, hence its important to avoid such configurations during manipulator operation. From the results, the integration of singularity detection and path planning demonstrated the practical application of theoretical concepts. The singularity detection method successfully identified configurations with potential motion limitations, and the path planning algorithm avoided these configurations by ensuring smooth trajectories through predefined waypoints.

However, the approaches used in this study also have limitations. The singularity detection focused solely on the determinant of the top 3×3 positional Jacobian matrix, which, while effective, may overlook singularities related to orientation or specific joint constraints. A more comprehensive approach could involve analyzing the full Jacobian matrix and incorporating higher-order criteria, such as second-order screw theory, to detect bifurcated singularities. In addition, while the path planning method straightforward, relies on predefined linear trajectories and does not dynamically adapt to environmental changes or obstacles. Future improvements could involve the incorporation of advanced optimization-based algorithms, such as RRT* or genetic algorithms, to enhance the adaptability and robustness of the planned paths. Additionally, considering joint constraints and velocity limits during trajectory generation would improve the feasibility of the implementation in real-world scenarios.

In the context of human-robot interaction, ensuring the safe operation of manipulators is critical. Singular configurations, if not properly addressed, could result in unpredictable behavior, posing safety risks in human environments. The proposed methodology directly contributes to safety by identifying and avoiding singularities during manipulator operation. Furthermore, smooth and predictable trajec-

tories reduce the likelihood of abrupt motions, ensuring reliable operation. For manipulators operating in shared human environments, additional factors, such as real-time obstacle detection and adaptive motion planning, must be considered. These enhancements would enable the manipulator to respond to dynamic changes in the workspace, further improving safety and efficiency.

3.5 Conclusion

The integration of screw theory and Jacobian analysis provided a robust framework for singularity detection, while the path planning algorithm demonstrated the practical applicability of these methods in ensuring safe and efficient manipulator operation. Although effective, the approaches used in this study highlight the need for further advancements, particularly in addressing orientation singularities, real-time adaptability, and dynamic obstacle avoidance. Future research should focus on these areas to develop more versatile and robust solutions for robotic manipulators in complex environments.

A Appendix

All the Matlab code and resulting images can be found on GitHub:

https://github.com/JiajunGuo1027/Serial-and-Parallel-Robot-Kinematics.git

References

Adriyan, A. and Sufiyanto, S. (2018). Kinematic and Singularity Analysis of PRoM-120 – A Parallel Robotic Manipulator with 2-PRU/PRS Kinematic Chains. *JURNAL REKAYASA MESIN* [online]. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.21776_ub.jrm.2018.009.03.7.

Choi, J., Lee, J., and Lee, H. (2012). Analysis of Jacobian and Singularity of Planar Parallel Robots Using Screw Theory. *TRANSACTIONS OF THE KOREAN SOCIETY OF MECHANICAL ENGINEERS A* [online]. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.3795_ksme-a.2012.36.11.1353.

Gallardo-Alvarado, J., Balmaceda-Santamaría, A. L., and Castillo-Castaneda, E. (2014). An Application of Screw Theory to The Kinematic Analysis of A Delta-type Robot. *JOURNAL OF MECHANICAL SCIENCE AND TECHNOLOGY* [online]. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.1007_s12206-014-0841-8.

Guo, S., Wang, C., Qu, H., and Fang, Y. (2013). A Novel 4-RRCR Parallel Mechanism Based on Screw Theory and Its Kinematics Analysis. *PROCEEDINGS OF THE INSTITUTION OF MECHANICAL ENGINEERS, ...* [online]. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.1177_0954406212469774.

Hao, K. and Ding, Y. (2006). Screw Theory and Singularity Analysis of Parallel Robots. *2006 INTERNATIONAL CONFERENCE ON MECHATRONICS AND AUTOMATION* [online]. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.1109_icma.2006.257468.

Kang, J.-K. and Kim, W. (2012). Workspace Optimal Design of Parallel Mechanisms Reflecting The Singularity Locations. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.7746_jkros.2012.7.2.101.

Liu, G. J., Qu, Z., Liu, X., and Han, J. (2014). Singularity Analysis and Detection of 6-UCU Parallel Manipulator. *ROBOTICS AND COMPUTER-INTEGRATED MANUFACTURING* [online]. Available from: https://www.paperdigest.org/paper/?paper_id=doi.org_10.1016_j.rcim.2013.09.010.