# ROBT403 Laboratory Work 3

Assem Akbayeva, Aidana Zhumagaliyeva,
Alma Toleubekova, Zhuldyz Altayeva

***Index Terms*— ROS, Inverse Kinematics, Forward Kinematics, 3 DOF planar manipulator**

## I. Introduction

In this lab, we explore the implementation of trajectory generation for robotic joint movements using cubic polynomials. The focus of the lab is on trajectory planning within the robot's workspace, both without and with via points, and performing forward and inverse kinematics calculations for the 3-DOF system, which are essential for determining both the joint angles needed to achieve specific end-effector positions and the position of the end-effector based on given joint angles.

For the real robot, we used 5-degree-of-freedom (5-DOF) robotic arm, from which we deactivated joints 2 and 4 to obtain a simplified 3-DOF configuration. For the simulation part, we recreated the robot in Gazebo environment using the dimensions of the real robot.

Through these exercises, we gain a deeper understanding of trajectory planning and control, ensuring the robot can execute smooth movements across its workspace.

## II. Methodology

### A. Task 1

*1) Calculating Forward Kinematics:* In robotics, forward kinematics is used to determine the position and orientation of the end-effector (tool) given the joint parameters. We'll derive the forward kinematics using the Denavit-Hartenberg (DH) convention for a 3-DOF (degree of freedom) robot.

First of all, we assumed that **zero frame and frame 1** are placed on the same axis at the first joint, hence, the distance $a_0$ (from $z_0$ to $z_1$) would be 0.

*2) Calculating the DH Parameters:* 1). Because of our assumption, where the **zero frame is along with frame 1 on the first joint**:

- The distance between the two $z$-axes, $a_0$, is 0 because the frames are coincident at the first joint.
- However, for subsequent links (from frame 2 onward), the link lengths $a_1, a_2$ will be the physical lengths of the links.

Submitted November 2021 and revised month year.
*Corresponding authors: Students.*
Z. Kappassov is with Nazarbayev University, Nur-Sultan, Kazakhstan (e-mail: kappassov@isir.upmc.fr).
This work was supported by MES of Kazakhstan, Grant number AP09058050.

The following DH table will be constructed:

| Link $i$ | $\alpha_{i-1}$ (deg) | $a_{i-1}$ (mm) | $d_i$ (mm) | $\theta_i$ (deg) |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | $\theta_1$ |
| 2 | 0 | 145 | 0 | $\theta_2$ |
| 3 | 0 | 145 | 0 | $\theta_3$ |
| End-effector | 0 | 45 | N/A | N/A |

where, the parameters mean:

- $\alpha_{i-1}$ (Twist angle): The angle between $z_{i-1}$ and $z_i$, measured about $x_{i-1}$.
- $a_{i-1}$ (Link length): The distance between $z_{i-1}$ and $z_i$ along $x_{i-1}$.
- $d_i$ (Link offset): The distance along $z_{i-1}$ between frames $i-1$ and $i$.
- $\theta_i$ (Joint angle): The angle between $x_{i-1}$ and $x_i$, measured about $z_{i-1}$.

2). Explanation

- **For link 1**: Since the zero frame coincides with frame 1 at the first joint, the distance $a_0 = 0$. The twist angle $\alpha_0 = 0$ because there is no rotation between $z_0$ and $z_1$.
- **For link 2 and 3**: These are the physical link lengths of your manipulator, so $a_1 = 145\,\text{mm}$ and $a_2 = 145\,\text{mm}$.
- **For the end-effector**: We still need to include the offset $L_3 = 45\,\text{mm}$, since the end-effector extends beyond the third joint.

### B. Forward Kinematics Derivation Using DH Parameters

To derive the forward kinematics, we will use the standard Denavit-Hartenberg transformation matrix formula, defined as follows:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_{i-1} & \sin\theta_i\sin\alpha_{i-1} & a_{i-1}\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_{i-1} & -\cos\theta_i\sin\alpha_{i-1} & a_{i-1}\sin\theta_i \\ 0 & \sin\alpha_{i-1} & \cos\alpha_{i-1} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We will proceed link by link using the following DH parameters we derived above.

**1. Transformation Matrices**

**From Base (Frame 0) to Link 1 (Frame 1)**:

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

**Explanation:**

- Since $a_0 = 0$, there is no translation along the $x_0$-axis.
- $\alpha_0 = 0$ means no rotation around the $x_0$-axis.
- $d_1 = 0$, so no translation along the $z_1$-axis.

**From Link 1 (Frame 1) to Link 2 (Frame 2)**:

$$T_2^1 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & 145\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & 145\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Explanation:**

- $a_1 = 145\,\text{mm}$, so there is a translation along the $x_1$-axis by 145 mm.
- $\alpha_1 = 0$, which indicates no rotation between the frames.

**From Link 2 (Frame 2) to Link 3 (Frame 3)**:

$$T_3^2 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 145\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & 145\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Explanation:**

- $a_2 = 145\,\text{mm}$, so there is a translation along the $x_2$-axis by 145 mm.
- $\alpha_2 = 0$, indicating no rotation between the frames.

**From Link 3 (Frame 3) to End Effector**: This transformation matrix represents a pure translation by $L_3 = 45\,\text{mm}$ along the $x_3$-axis:

$$T_{\text{EE}}^3 = \begin{bmatrix} 1 & 0 & 0 & 45 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**2. Total Transformation Matrix**

To find the overall forward kinematics, we will multiply the individual transformation matrices:

$$T_{\text{total}} = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdot T_{\text{EE}}^3$$

This gives us the complete transformation matrix from the base frame to the end-effector. We will perform the multiplication step by step:

**Step 1: Multiply $T_1^0$ and $T_2^1$:**

$$T_2^0 = T_1^0 \cdot T_2^1 = \begin{bmatrix} c_{\theta_1}c_{\theta_2} & -c_{\theta_1}s_{\theta_2} & 0 & 145c_{\theta_1}c_{\theta_2} \\ s_{\theta_1}c_{\theta_2} & -s_{\theta_1}s_{\theta_2} & 0 & 145s_{\theta_1}c_{\theta_2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Step 2: Multiply $T_2^0$ and $T_3^2$:**

$$T_3^0 = T_2^0 \cdot T_3^2 = \begin{bmatrix} \cos(\theta_{123}) & -\sin(\theta_{123}) & 0 & X_2 \\ \sin(\theta_{123}) & \cos(\theta_{123}) & 0 & Y_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Step 3: Multiply $T_3^0$ and $T_{\text{EE}}^3$:**

$$T_{\text{EE}}^0 = T_3^0 \cdot T_{\text{EE}}^3 = \begin{bmatrix} c_{\theta_{123}} & -s_{\theta_{123}} & 0 & 145\left(c_{\theta_{12}} + c_{\theta_{123}}\right) + 45c_{\theta_{123}} \\ s_{\theta_{123}} & c_{\theta_{123}} & 0 & 145\left(s_{\theta_{12}} + s_{\theta_{123}}\right) + 45s_{\theta_{123}} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

**Final Transformation:** The final transformation matrix $T_{\text{EE}}^0$ represents the position and orientation of the end-effector with respect to the base frame.

### C. Kinematic Modeling and Workspace Visualization

We coded a method for computing forward kinematics for a 3-DoF planar robot using Denavit-Hartenberg (D-H) parameters. The function `dh_transform` generates transformation matrices based on joint angles, link lengths, and twists. We then included a `forward_kinematics` function that combines these transformations to calculate the end-effector's position given the joint angles `theta1`, `theta2`, and `theta3`.

To visualize the workspace, we created the `plot_workspace` function. This function iterates through a range of joint angles, calculates the corresponding end-effector positions using the forward kinematics, and plots them in a 2D space. The workspace is displayed as a scatter plot, representing all reachable positions of the robot.

For demonstration, we used link lengths of `L1 = 135 mm`, `L2 = 135 mm`, and `L3 = 46.7 mm`. This shows the range of positions the robot can reach given these parameters.

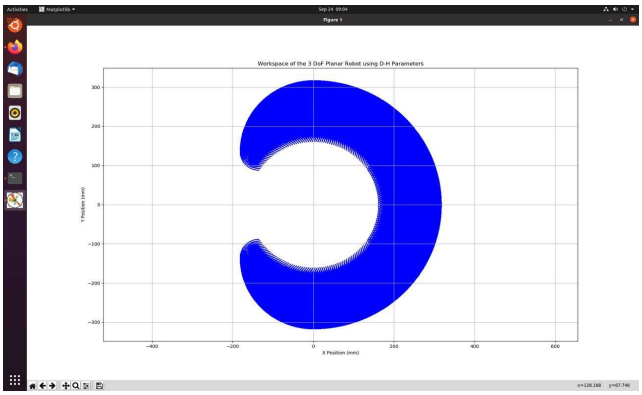



Fig. 1: Robot's workspace-90,+90 range of joints

This first figure is plotted using -90,+90 range of joints' motion, while the second plot represents the workspace of -45,+45 rotating robot joints.

### D. Task 2

In Task 2, we implemented the following functions:

- cubic-coefficients: This function calculates the coefficients of the cubic polynomial that describes the smooth transition from an initial joint position to a final joint position over a specified time interval. The cubic polynomial ensures that the initial and final velocities of the joint are zero, providing smooth and natural motion.
- cubic-trajectory: This function uses the cubic coefficients to evaluate the joint trajectory at any given point in time,

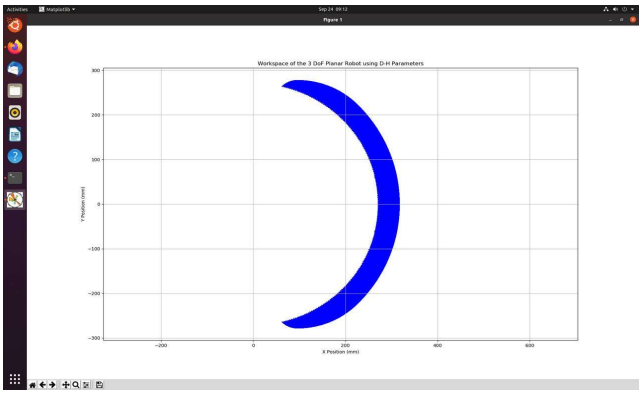


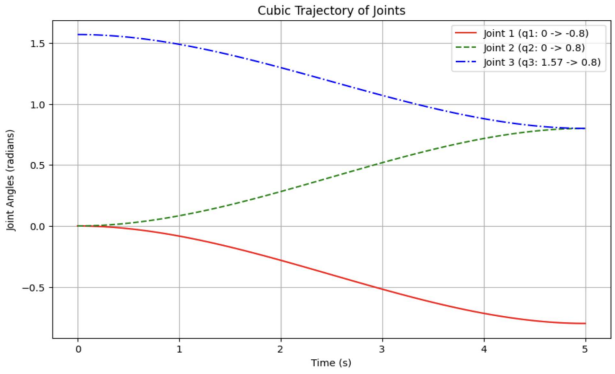

Fig. 2: Robot's workspace-45,+45 range of joints

generating the intermediate positions of the joint during the transition.

After implementing the functions, we calculated the cubic coefficients for each of the three joints. The joint movements are specified as follows:

- Joint 1 moves from 0 to -0.8 radians.
- Joint 2 moves from 0 to 0.8 radians.
- Joint 3 moves from 1.57 to 0.8 radians.

We then generated the trajectories for each joint over a time span of 5 seconds and plotted the resulting joint angles as a function of time. This visualization helps us observe how each joint moves smoothly from its initial to its final position following the cubic trajectory. The plots show the smooth transitions and validate the correctness of our cubic trajectory generation.




## E. Task 3

In the third task, we needed to implement of the function 'inverse_kinematics_with_orientation' for a 3-DOF planar manipulator. The task involves computing the joint angles of the manipulator given the desired end-effector position and orientation. Specifically, we are solving for the angles $q_1$, $q_2$, and $q_3$ of a 3-link planar manipulator using inverse kinematics. This problem is made more challenging by the fact that the end-effector is offset from the wrist by a fixed distance $L_3 = 45\,\text{mm}$, requiring us to first determine the wrist position before applying inverse kinematics.

*1) Position of the End-Effector:* We start by considering the desired position of the end-effector $(x, y)$ and its orientation $\phi$ relative to the x-axis. Due to the offset $L_3$, we first compute the position of the wrist, which is the point where the second link ends and the third joint is located. This wrist position is given by:

$$x_{\text{wrist}} = x - L_3 \cos(\phi), \tag{1}$$
$$y_{\text{wrist}} = y - L_3 \sin(\phi). \tag{2}$$

This compensates for the offset between the wrist and the actual end-effector position along the direction defined by $\phi$.

*2) Inverse Kinematics Equations:* Once the wrist position $(x_{\text{wrist}}, y_{\text{wrist}})$ is determined, we apply inverse kinematics to calculate the joint angles $\theta_1$ and $\theta_2$.

The equations of motion for the first and second links are:

$$x_{\text{wrist}} = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2), \tag{3}$$
$$y_{\text{wrist}} = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2), \tag{4}$$

where $l_1$ and $l_2$ are the lengths of the first and second links, respectively.

By squaring and adding these two equations, we can eliminate $\theta_1$ and obtain an expression for $\theta_2$:

$$x_{\text{wrist}}^2 + y_{\text{wrist}}^2 = l_1^2 + l_2^2 + 2l_1 l_2 \cos(\theta_2). \tag{5}$$

Solving for $\cos(\theta_2)$, we get:

$$\cos(\theta_2) = \frac{x_{\text{wrist}}^2 + y_{\text{wrist}}^2 - l_1^2 - l_2^2}{2l_1 l_2}. \tag{6}$$

*3) Solving for $\theta_2$:* Once we have $\cos(\theta_2)$, we can solve for $\theta_2$ using the `atan2` function to handle both the positive and negative square roots:

$$\theta_2 = \text{atan2}\left(\pm\sqrt{1 - \cos^2(\theta_2)}, \cos(\theta_2)\right). \tag{7}$$

Here, the choice of the positive or negative square root corresponds to selecting the *elbow-up* or *elbow-down* configuration.

*4) Solving for $\theta_1$:* With $\theta_2$ determined, we can solve for $\theta_1$. The original position equations can be rewritten as:

$$x_{\text{wrist}} = l_1 \cos(\theta_1) + k_1, \tag{8}$$
$$y_{\text{wrist}} = l_1 \sin(\theta_1) + k_2, \tag{9}$$

where:

$$k_1 = l_2 \cos(\theta_1 + \theta_2), \tag{10}$$
$$k_2 = l_2 \sin(\theta_1 + \theta_2). \tag{11}$$

Using the two-argument `atan2` function, we can now compute $\theta_1$:

$$\theta_1 = \text{atan2}(y_{\text{wrist}}, x_{\text{wrist}}) - \text{atan2}(k_2, k_1). \tag{12}$$

*5) Solving for $\theta_3$:* Finally, we compute $\theta_3$ using the desired orientation $\phi$ of the end-effector. Given that the sum of angles $\theta_1 + \theta_2$ determines the orientation of the wrist, we find $\theta_3$ as follows:

$$\theta_3 = \phi - (\theta_1 + \theta_2). \tag{13}$$

*6) Summary of Equations:* To summarize, the final joint angles $q_1$, $q_2$, and $q_3$ are calculated using the following steps:

- Wrist Position:

$$x_{\text{wrist}} = x - L_3 \cos(\phi), \tag{14}$$
$$y_{\text{wrist}} = y - L_3 \sin(\phi). \tag{15}$$

- Cosine of $\theta_2$:

$$\cos(\theta_2) = \frac{x_{\text{wrist}}^2 + y_{\text{wrist}}^2 - l_1^2 - l_2^2}{2 l_1 l_2}. \tag{16}$$

- Solve for $\theta_2$:

$$\theta_2 = \text{atan2}\left(\pm\sqrt{1 - \cos^2(\theta_2)}, \cos(\theta_2)\right). \tag{17}$$

- Solve for $\theta_1$:

$$\theta_1 = \text{atan2}(y_{\text{wrist}}, x_{\text{wrist}}) - \text{atan2}(k_2, k_1). \tag{18}$$

- Solve for $\theta_3$:

$$\theta_3 = \phi - (\theta_1 + \theta_2). \tag{19}$$

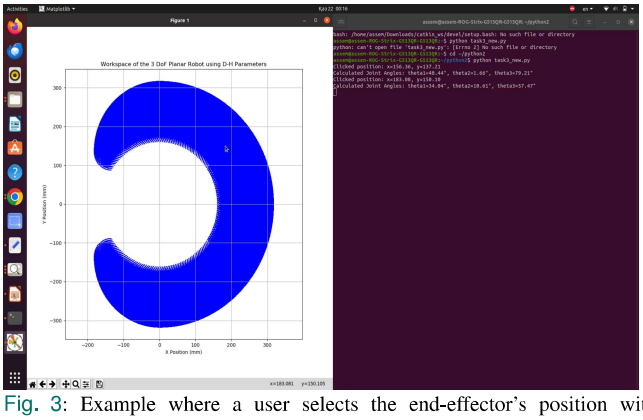



Fig. 3: Example where a user selects the end-effector's position within workspace and program calculates the joint angles

We coded a forward and inverse kinematics model for a 3-degrees-of-freedom (DoF) planar robot using Denavit-Hartenberg (D-H) parameters, and then integrated it with visualization capabilities to explore the robot's workspace.

*7) Step 1: Denavit-Hartenberg Transformation:* First, we included a function `dh_transform` to compute the transformation matrices based on the D-H convention. This function takes four parameters (`theta`, `d`, `a`, and `alpha`) which represent the joint angle, link offset, link length, and link twist, respectively. The function returns the corresponding transformation matrix, which is essential for calculating the relative position of one joint with respect to another. But if user clicks outside the workspace, then it does not provide any joint angles.

*8) Step 2: Forward Kinematics Calculation:* Next, we coded the forward kinematics through the `forward_kinematics` function. This function computes the end-effector's position (i.e., the robot's hand or tool) based on the angles of the three joints (`theta1`, `theta2`, and `theta3`) and the lengths of the robot's links (`L1`, `L2`, and `L3`). It uses the D-H transformation matrices and combines them to get the final position of the robot's end-effector.

*9) Step 3: Inverse Kinematics Calculation:* We then included the `inverse_kinematics` function to compute the joint angles required to position the end-effector at a given point in space (specified by coordinates `x` and `y`). This function uses trigonometric equations to solve for the joint angles based on the desired position and the robot's link lengths. If the desired point is outside the robot's reach, the function will notify that the target is unreachable.

*10) Step 4: Plotting the Workspace:* To visualize the robot's capabilities, we created the `plot_workspace` function, which plots the reachable area of the robot. It does this by simulating a range of possible joint angles and calculating the corresponding end-effector positions using forward kinematics. The resulting positions are displayed as points in a 2D plot, representing the workspace of the robot.

*11) Step 5: User Interaction:* Finally, we included a mouse click event handler, `on_click`, to allow users to interact with the plot. When the user clicks on a point within the plot, the inverse kinematics function calculates the joint angles needed to reach the clicked point. The calculated joint angles are then displayed, showing the user how the robot would need to move to reach the selected position.

*12) Example Application:* For demonstration, we used typical link lengths of `L1 = 135 mm`, `L2 = 135 mm`, and `L3 = 46.7 mm`. When run, the program generates a workspace plot and allows users to click on different points to see how the robot's joints would adjust to reach the selected positions. Then we simulated it in gazebo and on the real robot

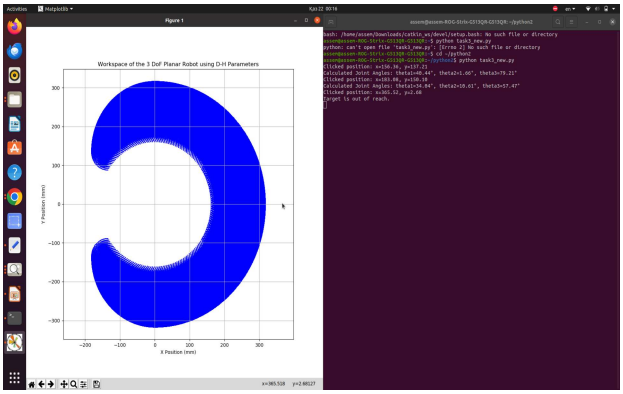



Fig. 4: Example where a user touches the area out of the robot's workspace

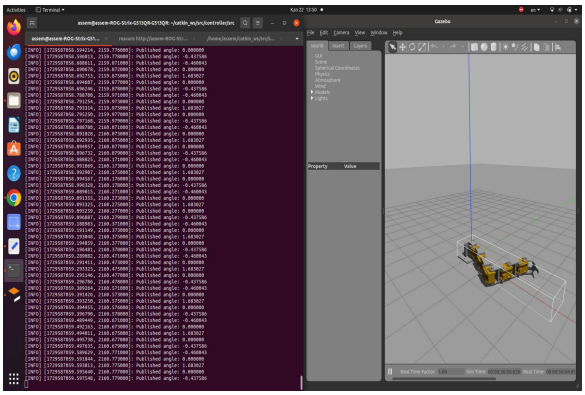


Fig. 5: Gazebo simulation where robot moves to defined end-point

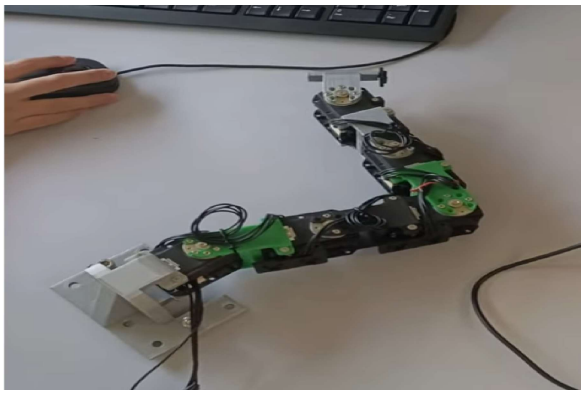


Fig. 6: The real robot moves to defined end-point

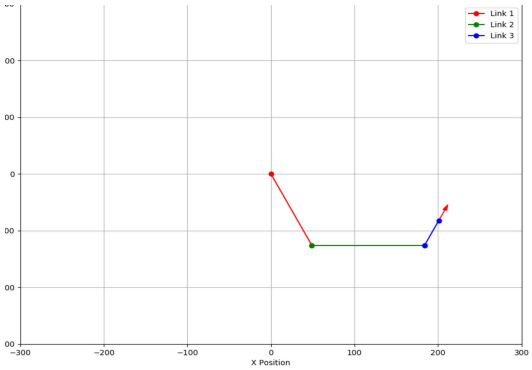



Fig. 8: Final configuration

## F. Task 4

In this task, the goal was to implement a function to visualize the configuration of a 3-DOF planar manipulator at any point during its motion trajectory. The function, plot_manipulator, is designed to plot the manipulator's joints and links, while also indicating the orientation of the end-effector with an arrow. Additionally, the function must clearly visualize both the initial and final configurations of the manipulator.

Using the D-H convention, we calculated the transformation matrices for each joint based on the input joint angles. A cubic polynomial trajectory was generated for each joint, providing smooth transitions from a randomly chosen starting position to a final position.

To visualize the workspace of the manipulator, the forward kinematics were calculated over a range of joint angles, and the corresponding end-effector positions were plotted.

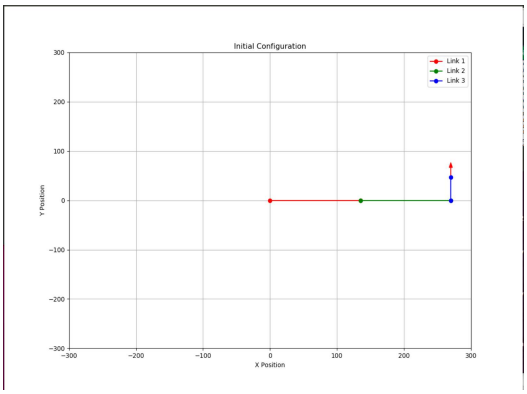



Fig. 7: Initial configuration

- Trajectory Plotting: The function plot_trajectory_within_workspace was implemented to first plot the workspace and then superimpose the trajectory of the end-effector as it moves from the start position to the end position. The trajectory was visualized using a cubic polynomial for smooth interpolation between points.
- Start and End Points: To clearly highlight the initial and final configurations, the start point was marked with a red dot, while the end point was marked with a green dot. This helps distinguish the two configurations clearly in the final plot.

The plot_manipulator function provides a clear and intuitive visualization of the manipulator's configuration and movement.

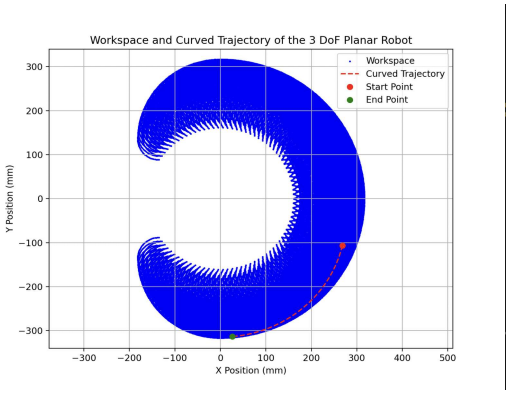



Fig. 9: Example 1 with randomly generated start and end points

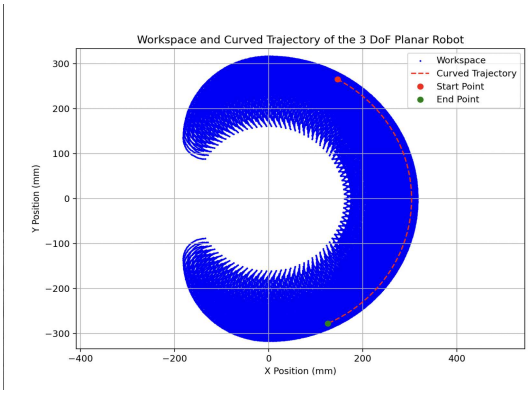



Fig. 10: Example 2 with randomly generated start and end points

## G. Task 5

The aim of this task was to generate joint-space cubic trajectories between multiple via points in the Cartesian workspace and compute the corresponding end-effector path in Cartesian space. To achieve the given objective we implemented similar algorithms as in Task 3: Inverse Kinematics (14) and (15) and Cubic Polynomial: Smooth interpolation of joint

angles between via points using the cubic polynomial method. Similar to the case without via-points, the program could reach the assigned trajectory on Cartesian Space only if the goal position was within the workspace, otherwise it would give according message. The generated joint-space trajectory smoothly interpolated between five via points in the Cartesian workspace. The function `plot_via_points_trajectory` was implemented to handle these tasks. It takes as inputs:

- A set of via points in Cartesian space.
- Initial and final time for the trajectory.
- Boundary conditions for position, velocity, and acceleration.

The trajectory was generated for three via points, and the joint angles were smoothly interpolated using cubic splines. The end-effector path was plotted in Cartesian space, showcasing smooth transitions between the via points. The corresponding joint trajectories were also plotted, showing continuous changes in joint angles with no abrupt discontinuities in velocity or acceleration. The code that was used for the simulation and real robot had 5 via points that are close to each other. The reason is that the higher amount ensures smoother trajectory for the robot's motion despite the increased computation time. Screenshots of the Gazebo simulation are presented below, there are start point, via points and goal position reached during the simulation:

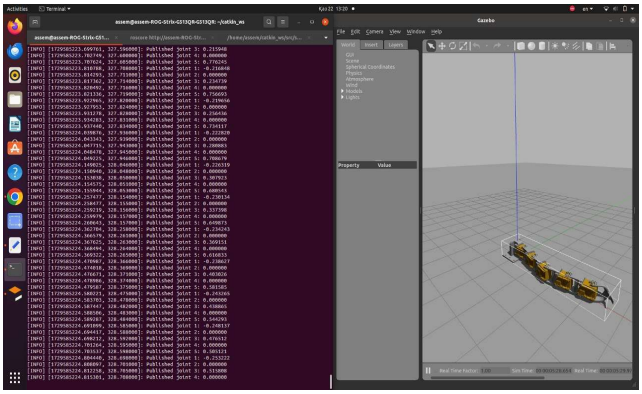


Fig. 11: Gazebo simulation at the start

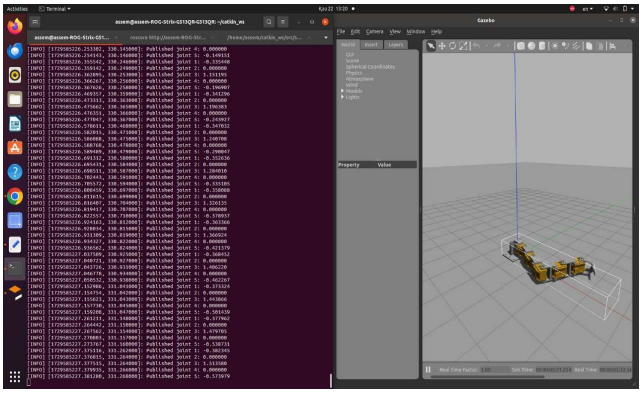


Fig. 12: Gazebo simulation at the first via-point

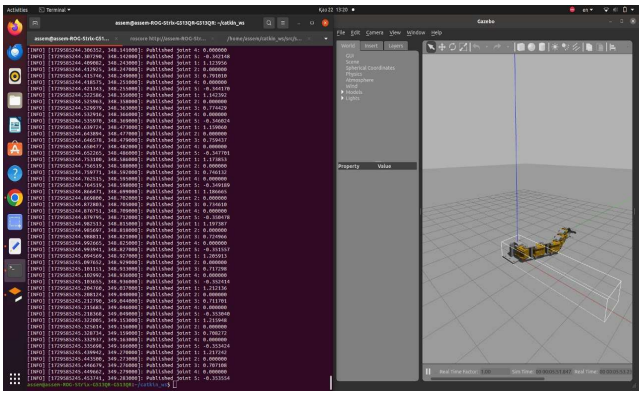


Fig. 13: Gazebo simulation at goal position

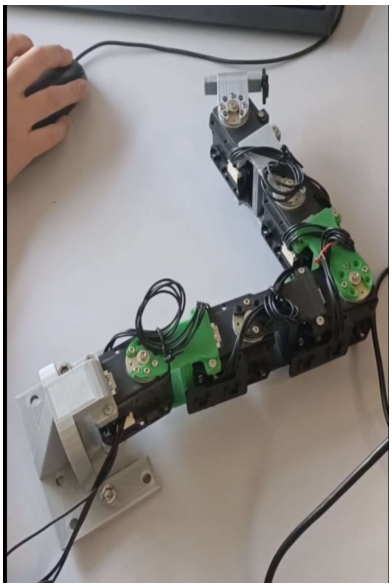


Fig. 14: Real Robot