Kinematic Analysis of Delta Parallel Robot

Report

April 27th 2025

Authors

Akaash S S
Mohnish Raja

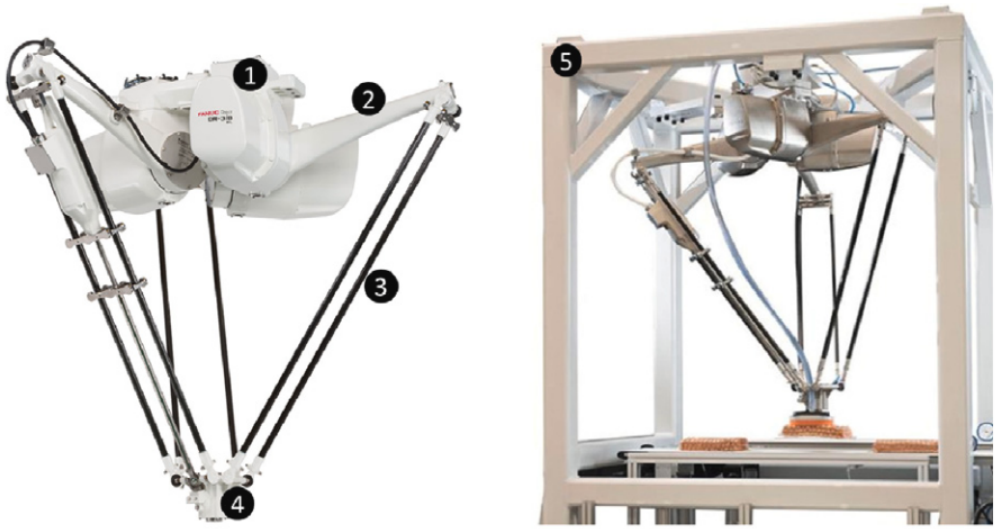
Contents

1 Abstract

This study presents the kinematic analysis and simulation of a Delta parallel robot. The robot model was developed in SolidWorks and imported into MATLAB for simulation. Forward kinematics was solved using a vector-based approach, while inverse kinematics was derived using the half-angle trigonometric method. The simulation captures both position and velocity profiles of the end-effector, along with the angular position and velocity of each joint. The workspace analysis is based on the forward kinematic solution. Both forward and inverse kinematics have been successfully implemented to evaluate the robot's motion characteristics under various input conditions.

2 Introduction

Robots are becoming more and more important in industries, especially for tasks that require high speed, precision, and repeatability. There are mainly two types of robots based on how their joints are connected — serial robots and parallel robots. Parallel robots are made up of multiple arms (or kinematic chains) that connect the base to the moving part, forming closed loops. This structure helps in sharing the load between the arms and offers advantages like higher accuracy, better stiffness, and faster movement compared to serial robots.



1- Fixed base
2- Link1 (Arm)
3- Link 2 (Forearm)
4- Moving base
5- Support structure

Figure 1: Delta Parallel Robot

Among different types of parallel robots, Delta robots are quite popular in applications like pick-and-place operations. One major challenge with parallel robots is solving the forward kinematics, which is usually more complex than the inverse kinematics. In this project, we studied the basic kinematics of a Delta robot using online resources and simulated its movement using MATLAB. The main focus

is on understanding how the robot moves, calculating its joint angles and end-effector position, and analyzing its workspace and motion behavior.

3 Model description of Delta Parallel Robot

The Delta-type parallel manipulator used in this project has a symmetrical structure made up of three identical arms. Each of these arms is directly connected to a motor that is mounted on a fixed base platform. These motors control the rotation of the upper arms, which are then linked to a set of lower arms using joints. The lower arms form parallelograms, which helps in maintaining the orientation of the end-effector and ensures that it only moves in translation — not rotation.

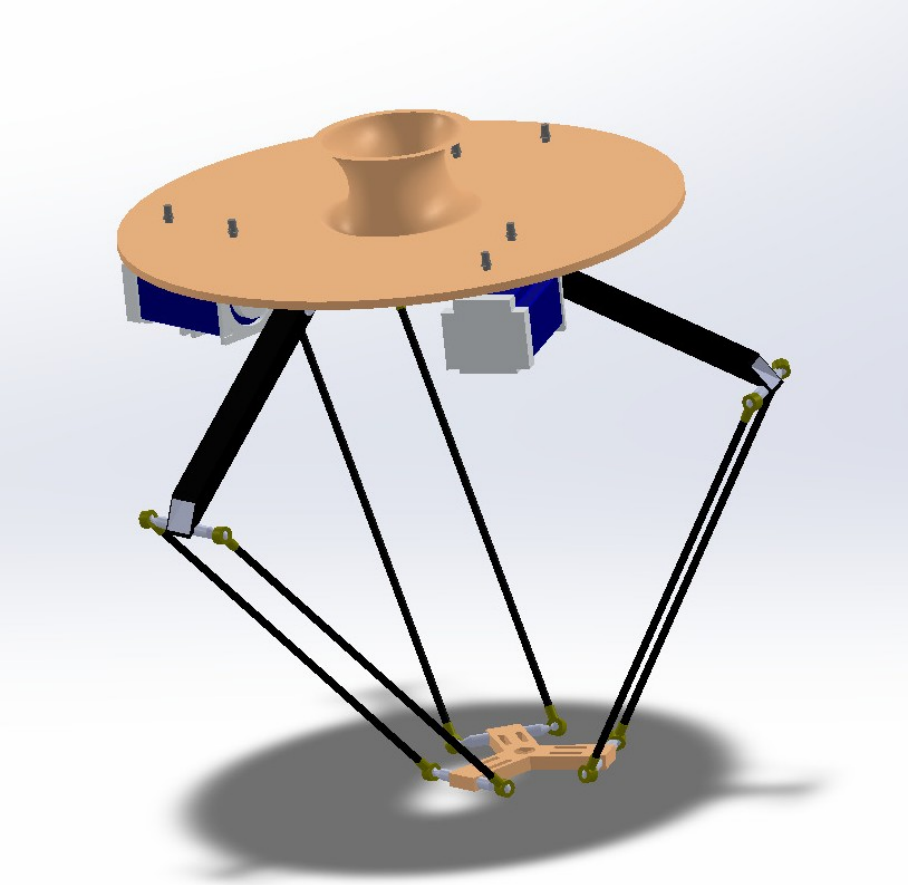


Figure 2: 3D CAD Model

The setup consists of two platforms: a fixed upper platform that holds the actuators, and a moving lower platform (also called the end-effector) that carries out the actual motion. Each of the three upper arms is connected at one end to a motor and at the other end to the lower parallelogram arms. The joints used between different links are universal joints, allowing motion in two directions. The parallelogram mechanism helps keep the orientation of the end-effector stable while it moves in 3D space.

The robot is designed to have three degrees of freedom, all of which are translational. This means the end-effector can move in the X, Y, and Z directions, but it cannot rotate. This setup allows the

robot to move very quickly and precisely within a limited working volume, which depends on the length of the links and how far the arms can move.

No.	Symbol	Data (mm)	Significance
1	R	70	The distance between the center point of the active arm and the center point of the coordinate system
2	L_1	95	Length of active arm
3	L_2	243	Length of driven arm
4	r	40	Distance between the center point of the terminal mobile platform and the middle point of the end of the driven arm

Table 1: Delta Robot Parameters

4 Forward Kinematic Analysis

The goal of forward kinematics is to find the position of the end-effector based on the known angles of the actuated joints. For the Delta robot, this is a bit tricky because it uses a parallel structure, which results in a closed-loop mechanism. Unlike inverse kinematics, which is usually simpler for parallel robots, forward kinematics involves solving nonlinear equations and often requires vector-based or geometric methods.
We define the following parameters:

- $\mathcal{R}$: Radius of the fixed (upper) platform
- ∇: Radius of the moving (lower) platform
- ℓ_1: Length of the upper (drive) arm
- ℓ_2: Length of the lower (follower) arm
- θ_i: Motor rotation angles for each of the three arms, where $i = 1, 2, 3$

The key points in each arm are:

- A_i: Point where the drive arm connects to the base
- B_i: Joint between the drive arm and follower arm
- C_i: Point on the moving platform where the follower arm is connected

4.1 Vector Positions

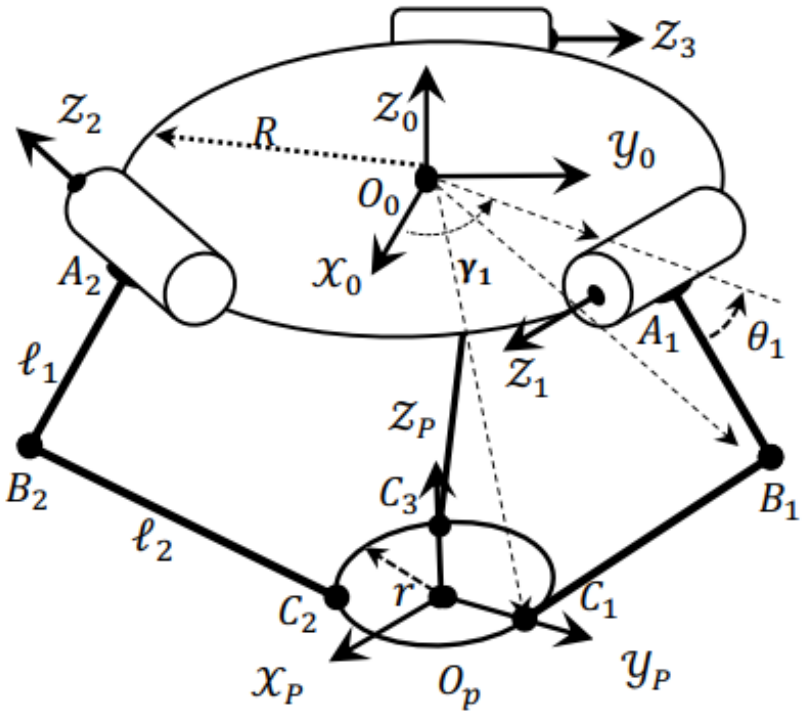


Figure 3: Kinematic diagram of Delta Parallel Robot

The position of point A_i in the base frame is:

$$\vec{O_o A_i} = \begin{bmatrix} \mathcal{R}\cos\gamma_i \\ \mathcal{R}\sin\gamma_i \\ 0 \end{bmatrix} \quad \text{where } \gamma_i = \frac{2\pi(i-1)}{3} \tag{1}$$

The position of point C_i in the moving platform frame is:

$$\vec{O_p C_i} = \begin{bmatrix} \nabla\cos\gamma_i \\ \nabla\sin\gamma_i \\ 0 \end{bmatrix} \tag{2}$$

The position of point B_i in the base frame (after applying the motor angle) is:

$$\vec{O_o B_i} = \begin{bmatrix} (\mathcal{R} + l_1\cos\theta_i)\cos\gamma_i \\ (\mathcal{R} + l_1\cos\theta_i)\sin\gamma_i \\ -l_1\sin\theta_i \end{bmatrix} \tag{3}$$

The center of the moving platform (i.e., end-effector) is denoted as:

$$\vec{E} = \begin{bmatrix} x_P \\ y_P \\ z_P \end{bmatrix} \tag{4}$$

Therefore, the point C_i in the base frame becomes:

$$\vec{O_oC_i} = \begin{bmatrix} x_P + \nabla \cos\gamma_i \\ y_P + \nabla \sin\gamma_i \\ z_P \end{bmatrix} \tag{5}$$

4.2 Constraint Equation

Since the lower link has a fixed length ℓ_2, the constraint for each arm is:

$$\|\vec{O_oB_i} - \vec{O_oC_i}\| = \ell_2 \tag{6}$$

Squaring and simplifying this gives the general form:

$$x_P^2 + y_P^2 + z_P^2 + a_i x_P + b_i y_P + c_i z_P + d_i = 0 \tag{7}$$

where:

$$\begin{aligned} a_i &= 2(\mathcal{R} - \nabla + \ell_1 \cos\theta_i)\cos\gamma_i \\ b_i &= 2(\mathcal{R} - \nabla + \ell_1 \cos\theta_i)\sin\gamma_i \\ c_i &= 2\ell_1 \sin\theta_i \\ d_i &= \ell_1^2 - \ell_2^2 + (\mathcal{R} - \nabla)^2 + 2(\mathcal{R} - \nabla)\ell_1 \cos\theta_i \end{aligned} \tag{8}$$

Writing this for $i = 1, 2, 3$, we get three nonlinear equations. Subtracting two of them from the third allows us to eliminate the squared terms and reduce the system to linear equations in x_P and y_P.

4.3 Substituting Linear Forms

We can express x_P and y_P as linear equations in terms of z_P:

$$x_P = m_1 z_P + n_1, \quad y_P = m_2 z_P + n_2 \tag{9}$$

Where:

$$m_1 = \frac{c_2b_1 - c_3b_1 + c_3b_2 - c_1b_2 + c_1b_3 - c_2b_3}{b_2a_1 - b_2a_2 - b_3a_1 + b_3a_2 - b_1a_2 + b_1a_3 + b_2a_2 - b_2a_3} \tag{10}$$

$$n_1 = \frac{d_2b_1 - d_2b_2 + d_3b_2 - d_1b_2 + d_2b_2 - d_2b_3}{b_2a_1 - b_2a_2 - b_3a_1 + b_3a_2 - b_1a_2 + b_1a_3 + b_2a_2 - b_2a_3} \tag{11}$$

$$m_2 = \frac{c_1a_2 - c_1a_3 + c_2a_3 - c_2a_1 + c_3a_1 - c_3a_2}{b_2a_1 - b_2a_2 - b_3a_1 + b_3a_2 - b_1a_2 + b_1a_3 + b_2a_2 - b_2a_3} \tag{12}$$

$$n_2 = \frac{d_1a_2 - d_1a_3 + d_2a_3 - d_2a_1 + d_3a_1 - d_3a_2}{b_2a_1 - b_2a_2 - b_3a_1 + b_3a_2 - b_1a_2 + b_1a_3 + b_2a_2 - b_2a_3} \tag{13}$$

Substituting these into the first equation gives a quadratic in z_P:

$$Az_P^2 + Bz_P + C = 0 \tag{14}$$

$$z_P = \frac{-a_1m_1 - b_1m_2 - c_1 - 2m_1n_1 - 2m_2n_2 \pm \sqrt{\Delta}}{2(m_1^2 + m_2^2 + 1)} \tag{15}$$

where the discriminant Δ is:

$$\begin{aligned} \Delta = {} & a_1^2m_1^2 + 2a_1b_1m_1m_2 + 2a_1c_1m_1 + 4a_1m_1m_2n_2 - 4a_1m_2^2n_1 - 4a_1n_1 \\ & + b_1^2m_2^2 + 2b_1c_1m_2 + 4b_1m_1m_2n_1 - 4b_1m_1^2n_2 - 4b_1n_2 \\ & + c_1^2 + 4c_1m_1n_1 + 4c_1m_2n_2 \\ & - 4d_1(m_1^2 + m_2^2 + 1) \\ & - 4m_1^2n_2^2 + 8m_1m_2n_1n_2 - 4m_2^2n_1^2 - 4n_1^2 - 4n_2^2 \end{aligned} \tag{16}$$

4.4 Final Solutions

Once z_P is found using the equation above, we can substitute it back into the linear equations to get:

$$x_P = m_1 z_P + n_1, \quad y_P = m_2 z_P + n_2 \tag{9}$$

$$x_P = \frac{-\frac{m_1}{2}\left(-\sqrt{\Delta} + P + Q\right) + n_1 R}{R} \tag{17}$$

$$y_P = \frac{-\frac{m_2}{2}\left(-\sqrt{\Delta} + P + Q\right) + n_2 R}{R}$$

$$\text{where:} \quad P = a_1 m_1 + b_1 m_2 + c_1, \quad Q = 2(m_1 n_1 + m_2 n_2), \quad R = m_1^2 + m_2^2 + 1 \tag{18}$$

This gives us the complete position (x_P, y_P, z_P) of the end-effector for a given set of motor angles $\theta_1, \theta_2, \theta_3$.

4.5 Feasibility of Forward Kinematics Solution

The Delta robot's direct kinematics solution can be interpreted geometrically as the **intersection point of three spheres**, where each sphere represents the possible location of the moving platform joint C_i, given the position of the elbow joint B_i and the fixed length of the follower link ℓ_2. However, not all mathematical solutions to this intersection are physically feasible. Once the forward kinematics equations are solved, it is necessary to validate whether the resulting position (x_P, y_P, z_P) satisfies all physical constraints of the mechanism.

1. Discriminant Check

The final expression for z_P comes from a quadratic equation of the form:

$$A z_P^2 + B z_P + C = 0 \tag{15}$$

To determine whether a real solution exists, we compute the discriminant:

$$\Delta = B^2 - 4AC \tag{19}$$

- If $\Delta < 0$, the solutions are complex, meaning the three spheres do not intersect and the joint angles are invalid.
- If $\Delta \geq 0$, real solutions for z_P exist, and further validation is required.

2. Solving for x_P and y_P

Once z_P is determined, the corresponding x_P and y_P values are obtained using linear relationships:

$$x_P = m_1 z_P + n_1, \quad y_P = m_2 z_P + n_2 \tag{9}$$

This gives us one or two possible positions (x_P, y_P, z_P), depending on the number of real roots.

3. Link Length Constraint Validation

Even if the values are real, they are only valid if they satisfy the constant-length constraint of the follower link. For each leg $i = 1, 2, 3$, the distance between points B_i and C_i must equal ℓ_2:

$$\left\| \vec{O_0 B_i} - \vec{O_0 C_i} \right\| \approx \ell_2 \tag{6}$$

Only the solutions that satisfy this for all three legs are considered feasible.

4. Selecting the Physical Solution

In most physical Delta robot setups, the moving platform lies below the fixed base. Therefore, among the real solutions of z_P, we typically select:

$$z_P < 0 \tag{20}$$

The other solution is generally rejected as it falls outside the usable workspace or violates the mechanical constraints.

4.6 Matlab Simulations

```
function [result_plus, result_minus] = ConstraintEquationSolver(R, r, l1, l2, theta, gamma)...

function [xP, yP, zP] = FeasiblityChecker(xP_minus, yP_minus, zp_minus, xP_plus, yP_plus, zp_plus)
    if zp_minus < 0
        xP = xP_minus;
        yP = yP_minus;
        zP = zp_minus;
    else
        xP = xP_plus;
        yP = yP_plus;
        zP = zp_plus;
    end
end

%Inputs
R = 70; r = 40; l1 = 95; l2 = 243;
theta = [pi/4, pi/3, pi/6]; % [theta1, theta2, theta3]

gamma = [0, 2*pi/3, 4*pi/3]; % [gamma1, gamma2, gamma3]

%Solves the constraint equation of FK for [Xp, Yp, Zp]
[result_plus, result_minus] = ConstraintEquationSolver(R, r, l1, l2, theta, gamma);

% Apply feasibility check for the obtained solution
[xP, yP, zP] = FeasiblityChecker(result_minus.xP, result_minus.yP, result_minus.zP, ...
                                 result_plus.xP, result_plus.yP, result_plus.zP);

% Display final output
fprintf('Final Position:\n');
fprintf('xP = %.6f\n', xP);
fprintf('yP = %.6f\n', yP);
fprintf('zP = %.6f\n', zP);
```

Figure 4: Forward Kinematics Matlab Script

5 Inverse Kinematics

Inverse kinematics (IK) refers to the process of finding the joint angles that position the robot's end-effector at a specified location. For the Delta parallel robot, which has three translational degrees of freedom, the goal is to calculate the three actuator angles $\theta_1, \theta_2, \theta_3$ given a target end-effector position (x_p, y_p, z_p).

This robot's IK is relatively straightforward compared to its forward kinematics.The inverse kinematics equation is obtained by transforming **equation (7)**. The equations are solved individually for each actuator using the **half-angle method**. This approach turns a trigonometric equation into a quadratic, making it much easier to handle.

5.1 Inverse Kinematics Equation

Each actuator satisfies an equation of the form:

$$A_i \cdot \cos(\theta_i) + B_i \cdot \sin(\theta_i) + C_i = 0 \quad \text{for } i = 1, 2, 3 \tag{21}$$

Where:

$$A_i = 2\ell_1 x_p \cos\gamma_i + 2\ell_1 y_p \sin\gamma_i + 2(\mathcal{R} - \nabla)\ell_1 \tag{22}$$
$$B_i = 2\ell_1 z_p \tag{23}$$
$$C_i = \ell_1^2 - \ell_2^2 + (\mathcal{R} - \nabla)^2 + 2(\mathcal{R} - \nabla)x_p \cos\gamma_i + 2(\mathcal{R} - \nabla)y_p \sin\gamma_i + x_p^2 + y_p^2 + z_p^2 \tag{24}$$

5.2 Final Solutions: Half-Angle Method

To solve the above equation, we define:

$$t_i = \tan\left(\frac{\theta_i}{2}\right) \tag{25}$$

Using half-angle identities:

$$\cos(\theta_i) = \frac{1 - t_i^2}{1 + t_i^2}, \quad \sin(\theta_i) = \frac{2t_i}{1 + t_i^2} \tag{26}$$

Substituting into the main equation:

$$A_i\left(\frac{1 - t_i^2}{1 + t_i^2}\right) + B_i\left(\frac{2t_i}{1 + t_i^2}\right) + C_i = 0 \tag{27}$$

Multiply through by $(1 + t_i^2)$:

$$A_i(1 - t_i^2) + 2B_i t_i + C_i(1 + t_i^2) = 0 \tag{28}$$

Simplifying:

$$(A_i - C_i)t_i^2 + 2B_i t_i + (A_i + C_i) = 0 \tag{29}$$

Solving this quadratic:

$$t_i = \frac{-B_i \pm \sqrt{B_i^2 - (A_i - C_i)(A_i + C_i)}}{A_i - C_i} \tag{30}$$

Finally, compute:

$$\theta_i = 2 \cdot \tan^{-1}(t_i) \tag{31}$$

This is repeated for $i = 1, 2, 3$.

5.3 Feasibility of Inverse Kinematics Solutions

Although the math gives two potential solutions for each joint, not all of them are physically valid. Several conditions must be checked:

1. **Joint Limits**: The calculated angle must be within the motor's allowable range.
2. **Configuration Selection**: Some solutions might correspond to physically undesirable arm configurations (like arm flipping or crossing).

3. **Link Constraints**: The solution must not cause the arms to stretch beyond their maximum length or result in self-collision.

4. **Discriminant Check**: The discriminant of the quadratic equation must be non-negative for a real solution to exist. A negative value indicates the point lies outside the reachable workspace.

In our implementation, we only consider real, feasible values of θ_i. If no valid solution exists, that end-effector position is considered unreachable.

5.4 Matlab Simulations

```
function [result_plus, result_minus] = ConstraintEquationSolver(R, r, l1, l2, theta, gamma)...

function [xP, yP, zP] = FeasiblityChecker(xP_minus, yP_minus, zp_minus, xP_plus, yP_plus, zp_plus)
    if zp_minus < 0
        xP = xP_minus;
        yP = yP_minus;
        zP = zp_minus;
    else
        xP = xP_plus;
        yP = yP_plus;
        zP = zp_plus;
    end
end

%Inputs
R = 70; r = 40; l1 = 95; l2 = 243;
theta = [pi/4, pi/3, pi/6]; % [theta1, theta2, theta3]

gamma = [0, 2*pi/3, 4*pi/3]; % [gamma1, gamma2, gamma3]

%Solves the constraint equation of FK for [Xp, Yp, Zp]
[result_plus, result_minus] = ConstraintEquationSolver(R, r, l1, l2, theta, gamma);

% Apply feasibility check for the obtained solution
[xP, yP, zP] = FeasiblityChecker(result_minus.xP, result_minus.yP, result_minus.zP, ...
                                 result_plus.xP, result_plus.yP, result_plus.zP);

% Display final output
fprintf('Final Position:\n');
fprintf('xP = %.6f\n', xP);
fprintf('yP = %.6f\n', yP);
fprintf('zP = %.6f\n', zP);
```

Figure 5: Inverse Kinematics Matlab Script

6 Workspace Analysis

The workspace of a robot refers to the total three-dimensional region that the end-effector can reach, based on the robot's mechanical design and joint constraints. The workspace of a Delta robot is determined by the lengths of its links, the geometry of the base and moving platforms, and the allowed range of motion of the actuated joints.

The workspace is the graphical representation of all valid solutions to the **forward kinematics** problem. All positions corresponding to a real solution of the forward kinematics are included in the plot.

This is done by iterating through a range of values for the three joint angles $\theta_1, \theta_2, \theta_3$, and solving the forward kinematics at each step. The positions of the end-effector (x, y, z) are calculated and stored if they result in feasible solutions. All the valid points are collected into a matrix, which is then used to plot the 3D workspace.

```m
%Inputs
R = 70; r = 40; l1 = 95; l2 = 243;
theta = [pi/4, pi/3, pi/6]; % [theta1, theta2, theta3]

angle = linspace(-pi/2, pi/2, 25);
[th1, th2, th3] = ndgrid(angle, angle, angle);

% Initialize workspace variable
wrkspace = [];

% Loop over all angle combinations
for i = 1:numel(th1)
    theta = [th1(i), th2(i), th3(i)];

    try
        % Forward Kinematics using your custom functions
        [result_plus, result_minus] = ConstraintEquationSolver(R, r, l1, l2, theta, gamma);
        [xP, yP, zP] = FeasiblityChecker(result_minus.xP, result_minus.yP, result_minus.zP, ...
                                         result_plus.xP, result_plus.yP, result_plus.zP);

        % If valid solution
        if isreal(xP) && isreal(yP) && isreal(zP) && zP < 0
            wrkspace = [wrkspace; xP, yP, zP];
        end

    catch
        % Skip if no real solution (e.g., due to complex sqrt)
        continue;
    end
end

%% PLOT WORKSPACE - SCATTER
figure;
scatter3(wrkspace(:,1), wrkspace(:,2), wrkspace(:,3), 5, 'filled');
title('Delta Robot Workspace - Scatter Plot');
xlabel('x (mm)');
ylabel('y (mm)');
zlabel('z (mm)');
grid on;
axis equal;


function [result_plus, result_minus] = ConstraintEquationSolver(R, r, l1, l2, theta, gamma)...

function [xP, yP, zP] = FeasiblityChecker(xP_minus, yP_minus, zp_minus, xP_plus, yP_plus, zp_plus)...
```

Figure 6: **Workspace Matlab Script**

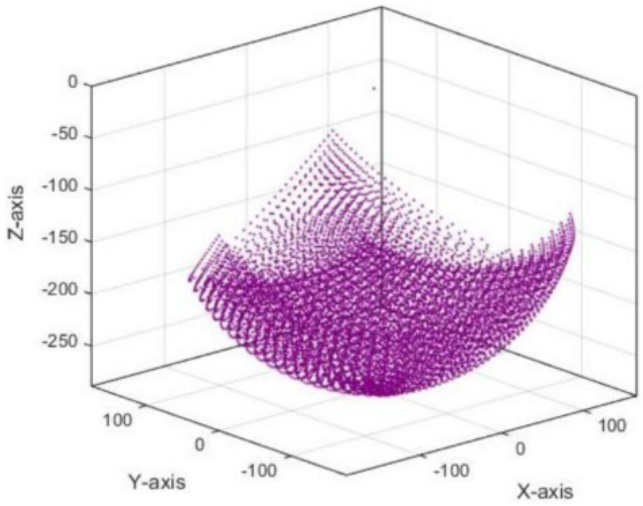


Figure 7: **Workspace Volume**

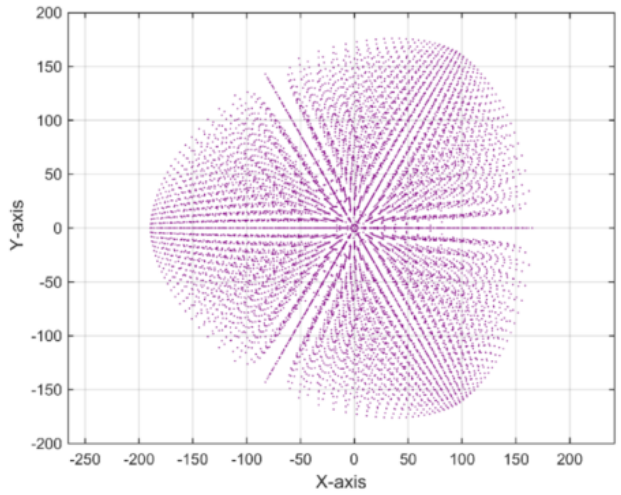


Figure 8: **Workspace - XY Plane**

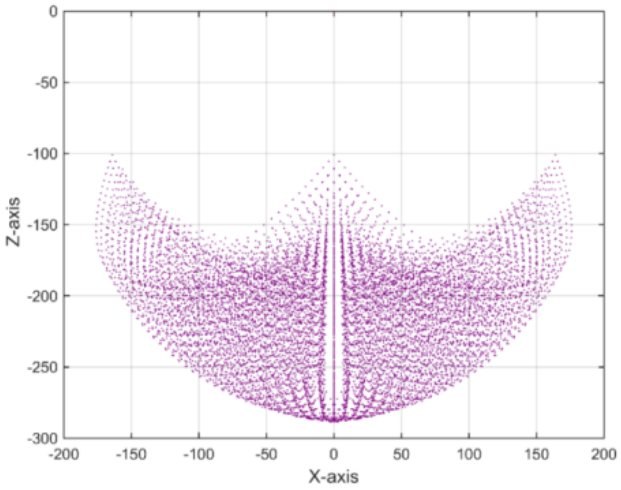


Figure 9: **Workspace - ZX Plane**

7 Conclusion

This project explored the basic kinematic modeling of a Delta-type parallel robot, focusing primarily on forward and inverse kinematics. Using geometric and vector-based methods, the position of the end-effector was derived from given joint angles, and the required joint angles were calculated for a specified end-effector position.

The forward kinematics was solved by vector based approach, representing the reachable regions of each arm. This method highlighted that while a mathematical solution may exist, not all are physically feasible due to constraints like link lengths and joint limits. The inverse kinematics was handled using a half-angle trigonometric method, which provided a relatively straightforward way to compute actuator angles from a given position.

Workspace analysis was also included to understand the range of motion and limitations of the mechanism. By iterating over various joint configurations, the feasible area the robot can reach was visualized.

In summary, this project helped in developing a basic understanding of how kinematic principles apply to parallel robots, particularly the Delta configuration. Though based entirely on online resources and existing literature, working through the mathematics and logic gave useful insights into the structure and constraints of parallel manipulators.