5-bar planar parallel robot

874H1Z: Robot Design and Implementation

Group Number: **2**

Name	Candidate number	Role
Di Li	299089	Group Leader
Yuting Shen	299081	Member
Dongbo Qu	299078	Member
Lujie Pan	299100	Member

November 20, 2025

Contents

1. Mechanism Design

1.1 Structural Design

We designed a planar five-bar parallel mechanism consisting of two actuated joints at the base and two passive links, with the end-effector located at the intersection point of two passive links. A schematic representation of the mechanism is shown below.

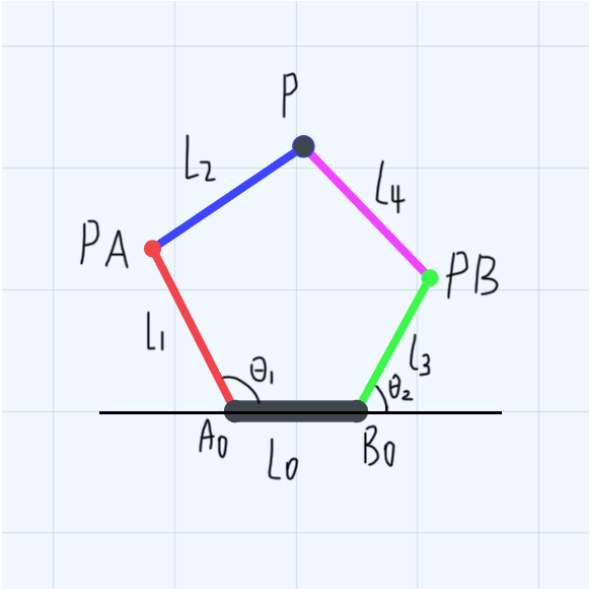


Figure 1.1: Schematic diagram of the five-bar mechanism.

The main geometric parameters include:

- Base spacing: L_0
- Actuated link lengths: L_1, L_3
- Passive link lengths: L_2, L_4
- Joint A: P_A
- Joint B: P_B
- End-effector : P
- L_1 angle: θ_1, L_3 angle: θ_2

1.2 Workspace and Link-Length Selection

Multiple link-length combinations were simulated using MATLAB to evaluate whether the robot could reach the center of every cell in an egg tray. The final selected link parameters satisfy:

- Full coverage of the egg-tray workspace.
- Compact structure suitable for narrow environments.
- Workspace center biased toward the right to avoid collision with the conveyor track.

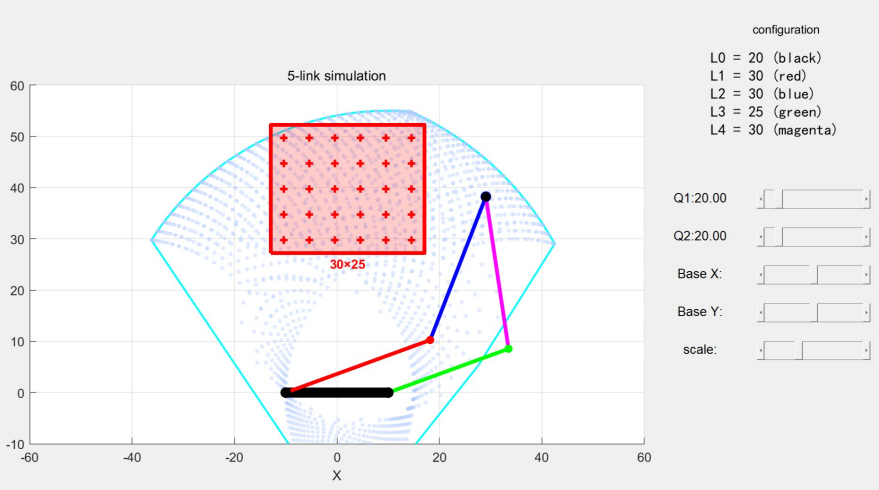


Figure 1.2: Simulated reachable workspace used for link-length selection.

We finally choose the robot's configuration:

$$L_0 = 200mm, L_1 = 300mm, L_2 = 300mm, L_3 = 250mm, L_4 = 300mm$$

2. Forward Kinematics

2.1 Geometric Formulation

Forward kinematics determines the position of the end-effector given joint angles θ_1 and θ_2.

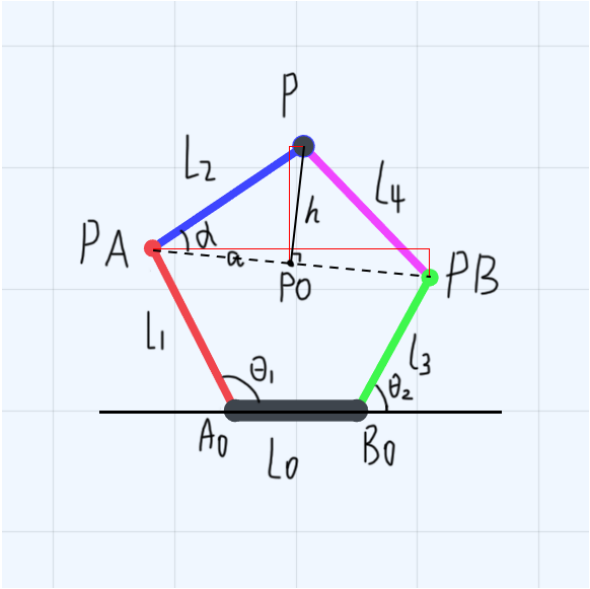


Figure 2.1: Schematic diagram of forward kinematics analysis.

The notation used in the forward kinematic derivation follows the link definitions and joint labels shown in Figure 2.1. Assume base A_0 is at $(0, 0)$ and base B_0 is at $(L_0, 0)$. Given joint angles θ_1 and θ_2:

$$P_A = \begin{bmatrix} L_1 \cos\theta_1 \\ L_1 \sin\theta_1 \end{bmatrix}, P_B = \begin{bmatrix} L_0 + L_3 \cos\theta_2 \\ L_3 \sin\theta_2 \end{bmatrix}$$

The end-effector point P must satisfy:

$$\|P - P_A\| = L_2, \quad \|P - P_B\| = L_4$$

Thus, P is the intersection of two circles centered at P_A and P_B with radii L_2 and L_4, respectively. Let

$$P_A = (x_A, y_A), \qquad P_B = (x_B, y_B).$$

Define the distance between the two circle centers ($P_A P_B$) :

$$d = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}.$$

A valid physical solution requires:

$$|L_2 - L_4| \leq d \leq L_2 + L_4.$$

Define the unit vector from P_A to P_B:

$$\hat{e} = \frac{1}{d} \begin{bmatrix} x_B - x_A \\ y_B - y_A \end{bmatrix}$$

Using P to create the perpendicular PP_0 for P_AP_B, define the distance from P_A to the P_0 is a.

$$a = L_2 \cos(\alpha), \quad \cos(\alpha) = \frac{L_2^2 + d^2 - L_4^2}{2dL_2} \quad \Rightarrow \quad a = \frac{L_2^2 - L_4^2 + d^2}{2d}$$

Thus:

$$P_0 = \begin{bmatrix} x_A \\ y_A \end{bmatrix} + a\hat{e}.$$

The length of PP_0 is:

$$h = \sqrt{L_2^2 - a^2}$$

Thus, the two possible intersection points are:

$$P = P_0 \pm h \begin{bmatrix} -\hat{e}_y \\ \hat{e}_x \end{bmatrix}$$

Written explicitly:

$$P_x = x_A + a\frac{x_B - x_A}{d} \mp h\frac{y_B - y_A}{d},$$

$$P_y = y_A + a\frac{y_B - y_A}{d} \pm h\frac{x_B - x_A}{d}$$

The $\pm$ corresponds to elbow-up configuration or elbow-down configuration.

2.2 MATLAB Simulation

The forward kinematics simulation shown in this section is implemented using the MATLAB scripts listed in Appendix A.1–A.6. The `robot.m` defines the configuration of robot sunch as length of link. The core solver `ForwKin_5link.m` computes the end-effector position from the actuated joint angles, while `draw_5link.m` and `draw_workspace.m` are used to visualize the robot configuration and to render the reachable workspace. Finally, the script `main_forwKin.m` integrates these functions to generate the configuration plots presented in the forward kinematics results.

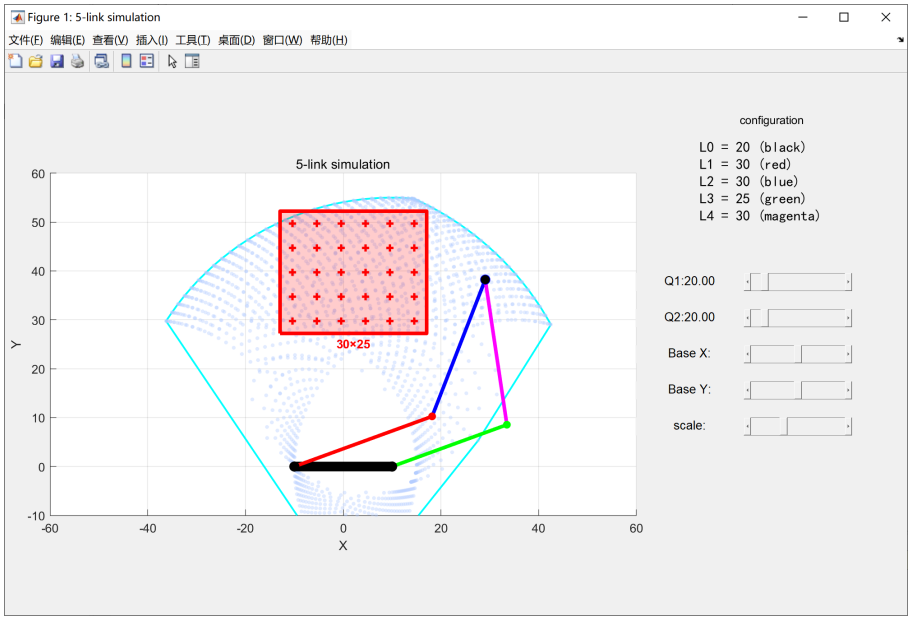


Figure 2.2: Configuration at $\theta_1 = 20°$, $\theta_2 = 20°$.

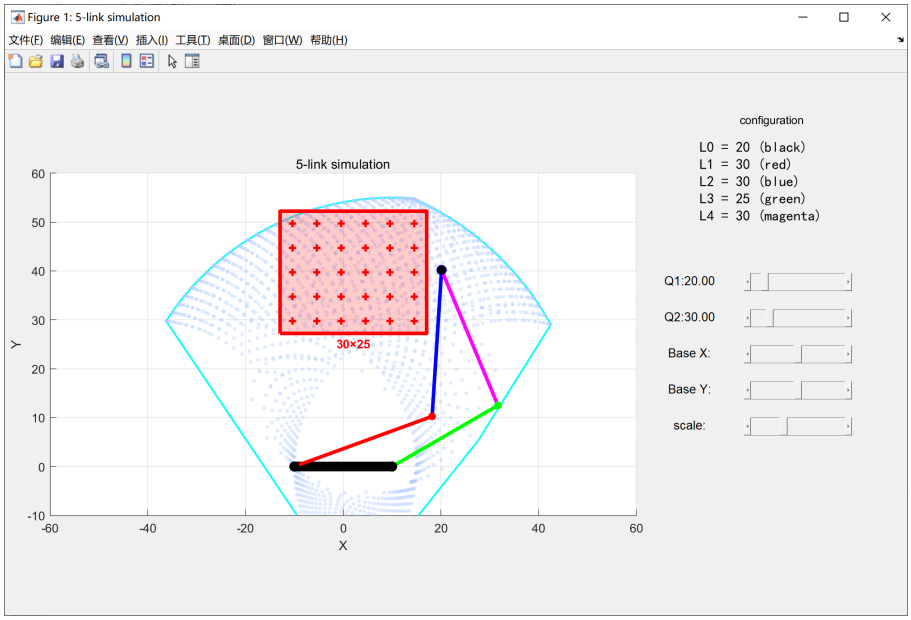


Figure 2.3: Configuration at $\theta_1 = 20°$, $\theta_2 = 30°$.

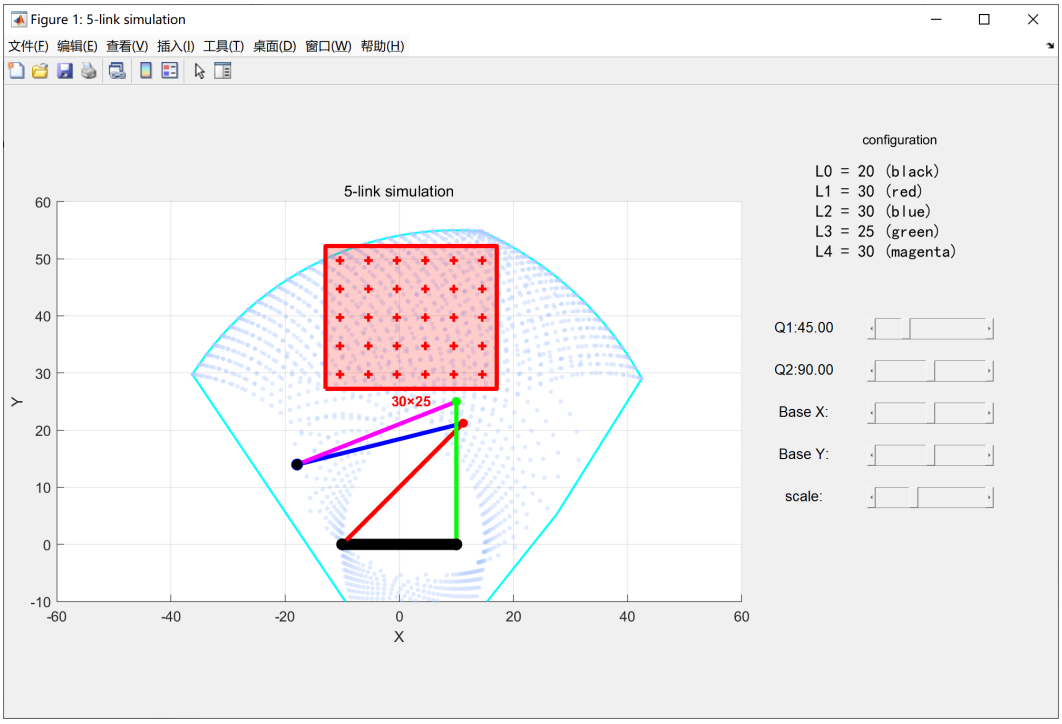


Figure 2.4: Configuration at $\theta_1 = 45°$, $\theta_2 = 90°$.

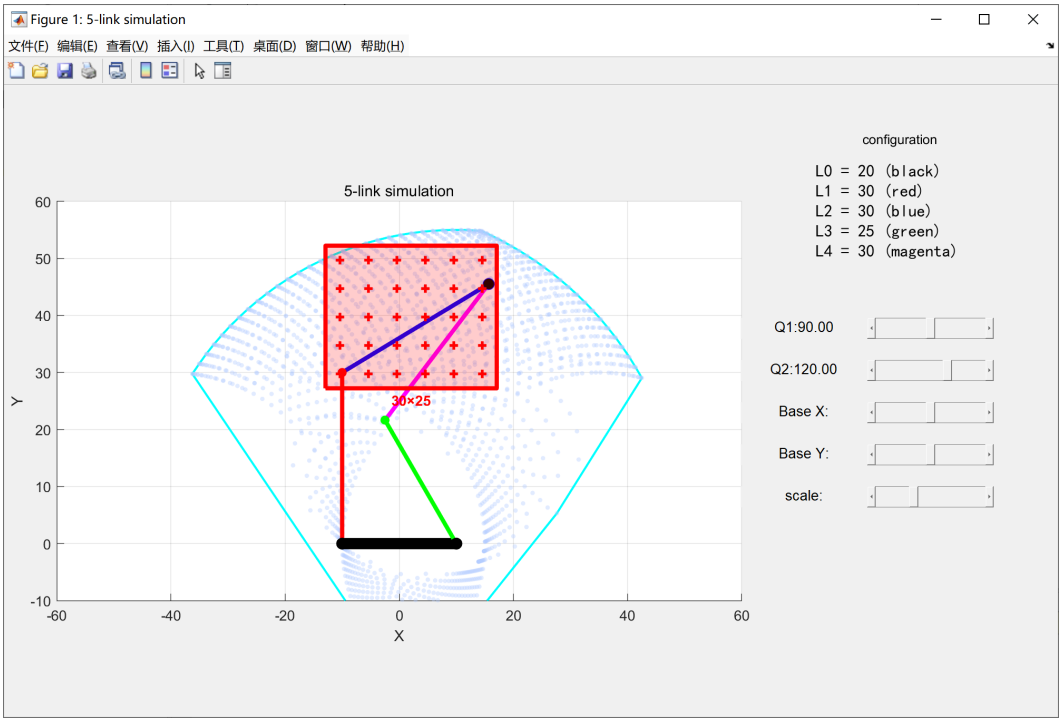


Figure 2.5: Configuration at $\theta_1 = 90°$, $\theta_2 = 120°$.

2.3 Issue of End-Effector Discontinuity

Because the mechanism is closed-loop and nonlinear, switching between elbow-up and elbow-down configurations causes an instantaneous jump in the end-effector position:

$$P_{\text{upper}} \neq P_{\text{lower}}$$

Such a configuration flip results in:

- Sudden reversal of passive-link angles.
- Potential mechanical collision or singularity crossing.
- Dangerous torques applied to physical servos.

Therefore, a fixed branch (typically elbow-up) must be maintained to avoid kinematic discontinuities and mechanism flipping, only the elbow-up branch is used in this project throughout.

3. Inverse Kinematics

3.1 Geometric Derivation

The inverse kinematics of the five-bar parallel manipulator determines the actuated joint angles θ_1 and θ_2 required to place the end-effector at a desired Cartesian location

$$P = (x, y).$$

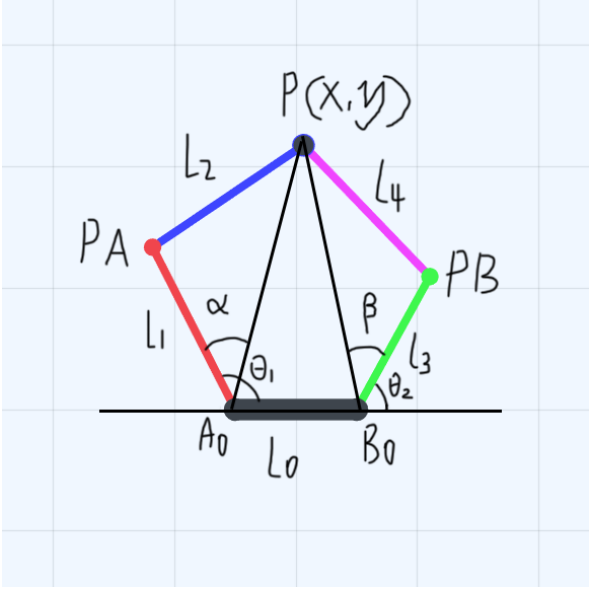


Figure 3.1: Schematic diagram of inverse kinematics analysis.

The notation used in the forward kinematic derivation follows the link definitions and joint labels shown in Figure 3.1.

Since the mechanism consists of two symmetric two-link serial chains meeting at P, the inverse kinematics may be solved independently for each chain.

For Chain A:

Base A_0 is fixed at $(0, 0)$. Given desired end–effector point P, the vector from A_0 to P is

$$\vec{v}_A = \begin{bmatrix} x \\ y \end{bmatrix}, \qquad l = \|\vec{v}_A\| = \sqrt{x^2 + y^2}.$$

The triangle formed by points A_0, A_2, and P has side lengths L_1, L_2, and l. Applying the law of cosines:

$$\cos\alpha = \frac{L_1^2 + l^2 - L_2^2}{2L_1 l}, \qquad \alpha = \arccos\left(\frac{L_1^2 + l^2 - L_2^2}{2L_1 l}\right).$$

Angle θ_1 is composed of the direction of $\vec{v}_A$ and the internal angle α:

$$\theta_1 = \operatorname{atan2}(y, x) \mp \alpha.$$

Thus Chain A has two possible solutions:

$$\theta_1^{(+)} = \operatorname{atan2}(y, x) - \alpha, \qquad \theta_1^{(-)} = \operatorname{atan2}(y, x) + \alpha,$$

corresponding to elbow-up and elbow-down configurations.

For Chain B:

Base B_0 is located at $(L_0, 0)$. The vector from B_0 to the target point P is

$$\vec{v}_B = \begin{bmatrix} x - L_0 \\ y \end{bmatrix}, \qquad m = \|\vec{v}_B\| = \sqrt{(x - L_0)^2 + y^2}.$$

The triangle B_0–B_2–P has sides L_3, L_4, and m. By the law of cosines:

$$\cos\beta = \frac{L_3^2 + m^2 - L_4^2}{2L_3 m}, \qquad \beta = \arccos\left(\frac{L_3^2 + m^2 - L_4^2}{2L_3 m}\right).$$

The direction of vector $\vec{v}_B$ is

$$\phi = \operatorname{atan2}(y,\ x - L_0).$$

Thus the joint angle at B_0 is:

$$\theta_2 = \pi - (\phi \mp \beta).$$

Explicitly:

$$\theta_2^{(+)} = \pi - (\phi - \beta), \qquad \theta_2^{(-)} = \pi - (\phi + \beta).$$

Again, "+" and "−" correspond to elbow-up and elbow-down.

3.2 Feasible IK Sets

Since Chain A has two possible solutions and Chain B has two possible solutions, the full manipulator admits up to four possible inverse kinematic configurations:

$$(\theta_1, \theta_2) \in \left\{ (\theta_1^{(+)}, \theta_2^{(+)}), (\theta_1^{(+)}, \theta_2^{(-)}), (\theta_1^{(-)}, \theta_2^{(+)}), (\theta_1^{(-)}, \theta_2^{(-)}) \right\}.$$

Only those satisfying the following constraints are physically valid:

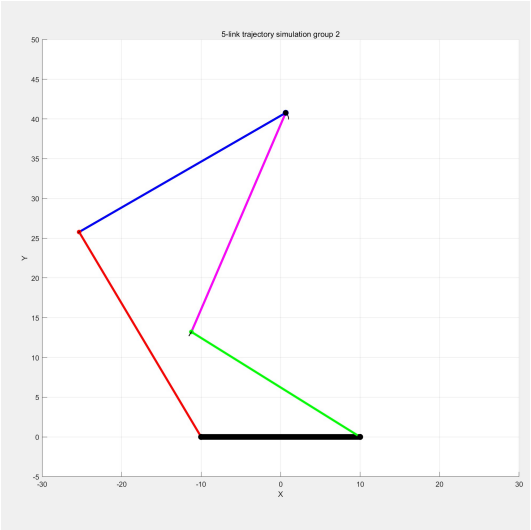
- Consistency with the selected branch (always elbow-up).
- Angle greater than or equal to 0 and less than or equal to 180.

3.3 Trajectory Tracking

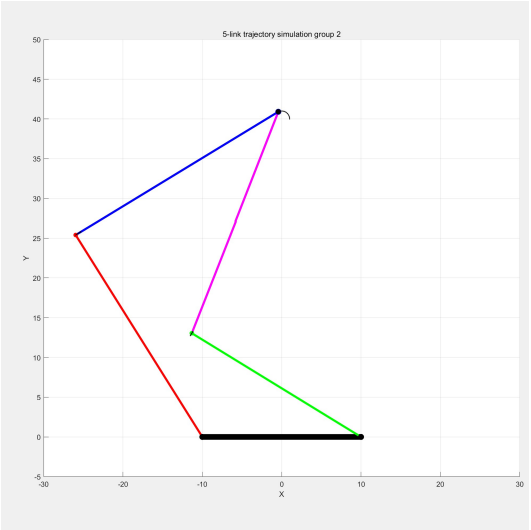
Trajectory Tracking Implementation

The trajectory tracking shown in this section corresponds to the MATLAB scripts listed in Appendix A.7–A.10. The file `main_invKin.m` generates the desired end-effector trajectories, including the circular and square paths used in the experiments and executes the full simulation pipeline to produce the plotted trajectory-following results. The script `InvKin_trajectory.m` converts these Cartesian trajectory points into joint-space trajectories by evaluating the inverse kinematics at each point. Finally, the function `draw_trajectory.m` visualizes both the end-effector motion and the corresponding joint configurations, producing the trajectory plots shown in this section.

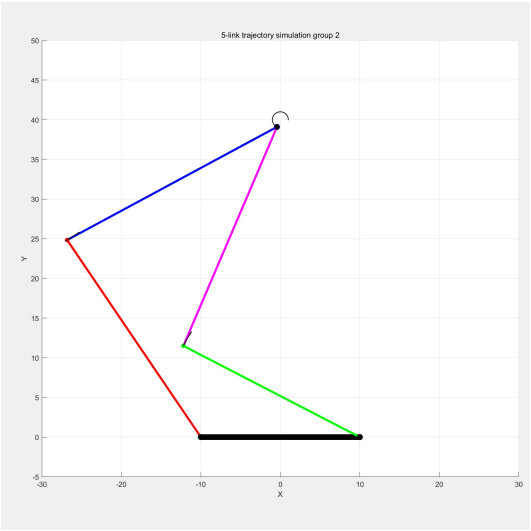
3.3.1 Circular trajectory (diameter: 2 cm)



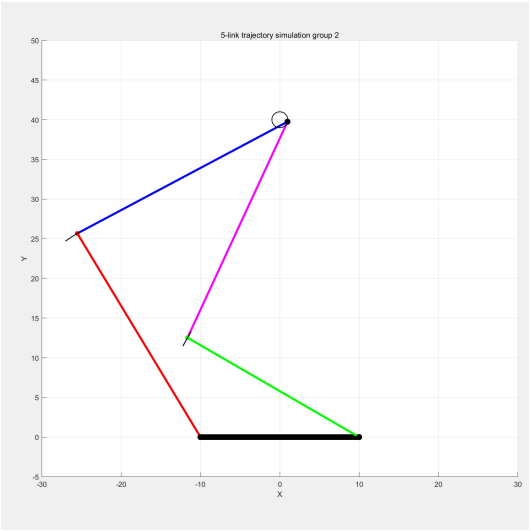
(a) Circular trajectory tracking example 1.



(b) Circular trajectory tracking example 2.



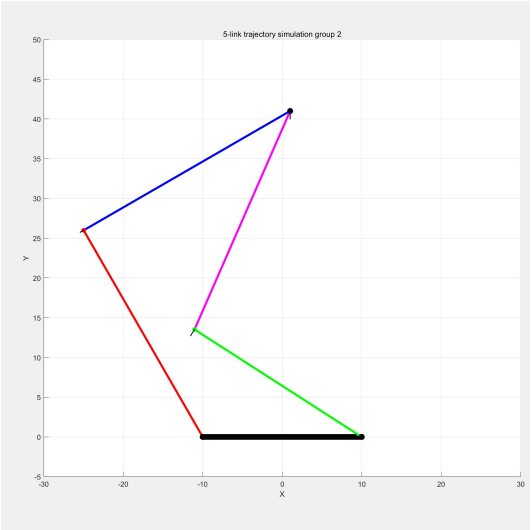
(c) Circular trajectory tracking example 3.



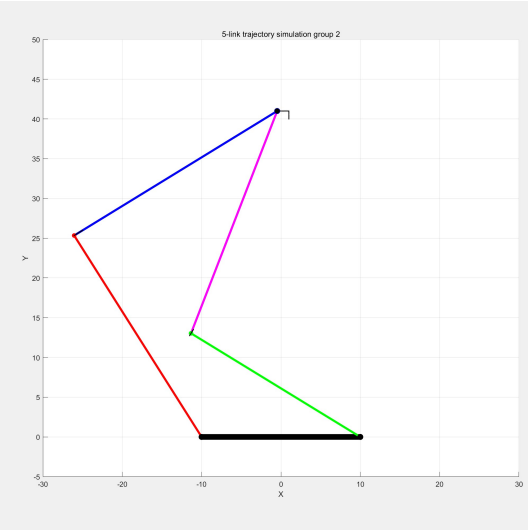
(d) Circular trajectory tracking example 4.

Figure 3.2: Circular trajectory tracking examples.

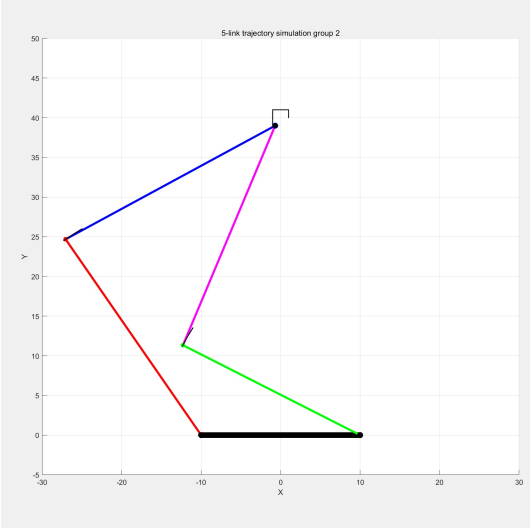
3.3.2 Square trajectory (side length: 2 cm)



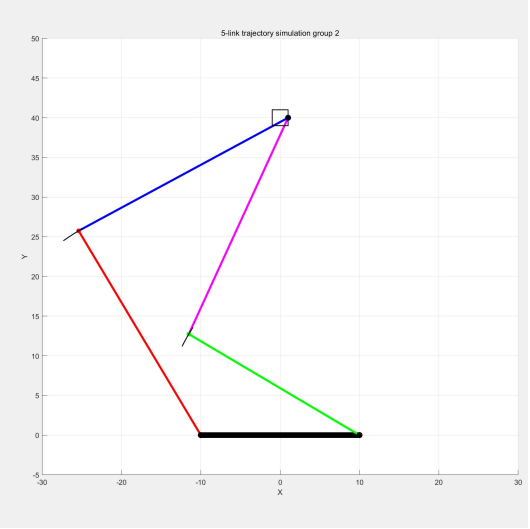
(a) Square trajectory tracking example 1.



(b) Square trajectory tracking example 2.



(c) Square trajectory tracking example 3.



(d) Square trajectory tracking example 4.

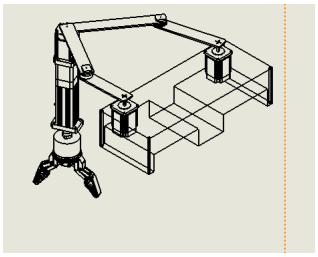
Figure 3.3: Square trajectory tracking examples.

4. 3D CAD Modeling and Structural Analysis

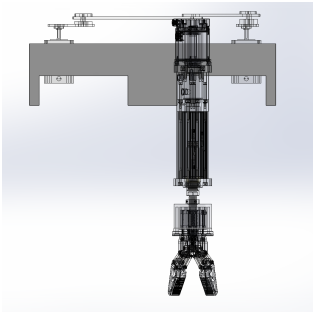
4.1 CAD Assembly

The full robot assembly was modeled using SolidWorks based on previously derived kinematic parameters, with the design comprehensively considering factors such as manufacturability, assembly sequence, motor selection, and functional requirements of the end-effector.

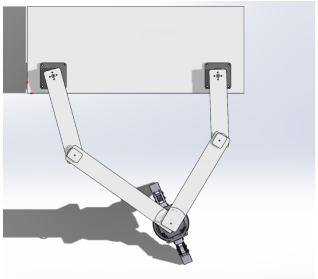
- **Base:** A robust base was designed to mount the two drive motors (servo or stepper motors) while maintaining the specified base distance L_0.
- **Links:** All links (L_1, L_2, L_3, L_4) are designed using alloy steel. To ensure rigidity and prevent fracture, the main body of the links features an I-shaped cross-section.
- **Joint Connections:** Screws are used for axial positioning at the revolute joints.
- **Motor Connection:** The motors are directly coupled to the driving links (L_1 and L_3), ensuring zero-backlash power transmission.
- **End-Effector:** As required by the project specifications, the end-effector is designed as a universal gripper interface.



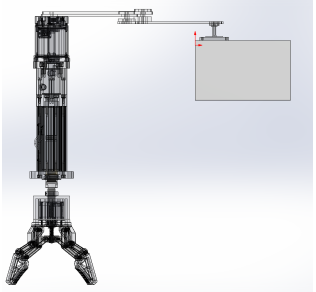
(a) Isometric view of the assembly.



(b) Front view engineering drawing.



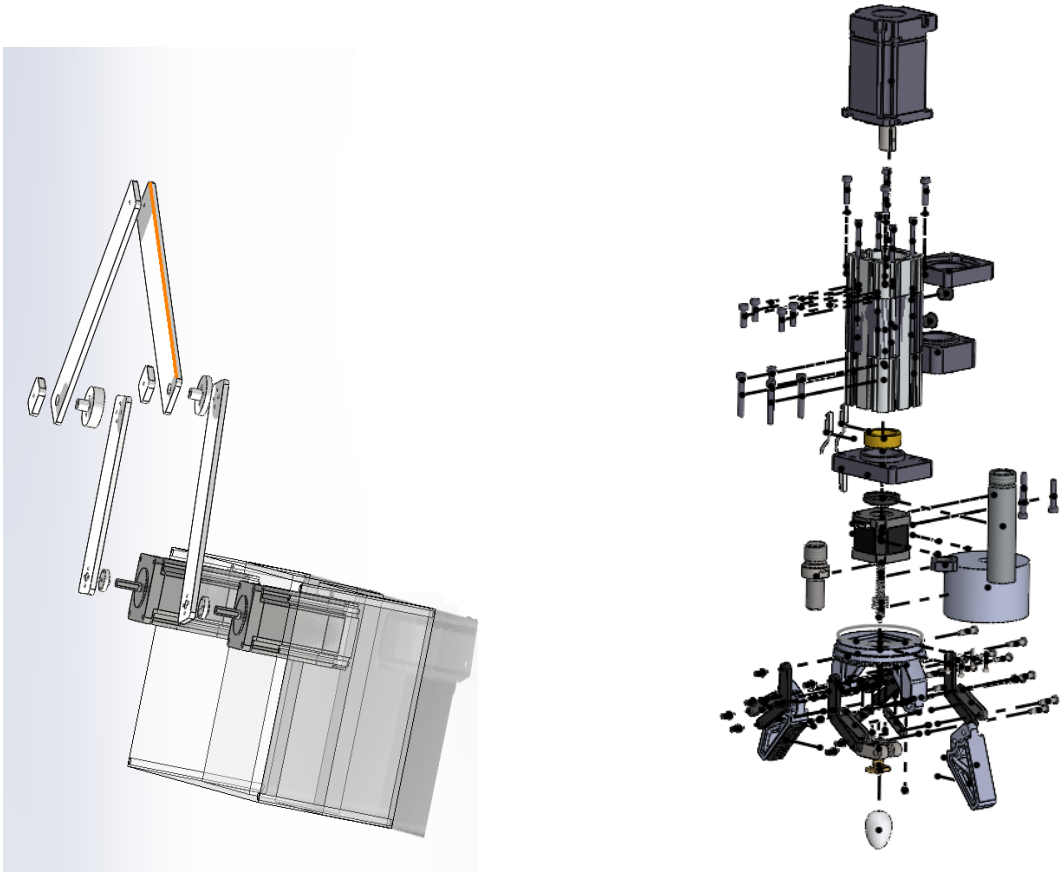
(c) Top view engineering drawing.



(d) Side view engineering drawing.

Figure 4.1: Multi-view engineering drawings of the assembly.

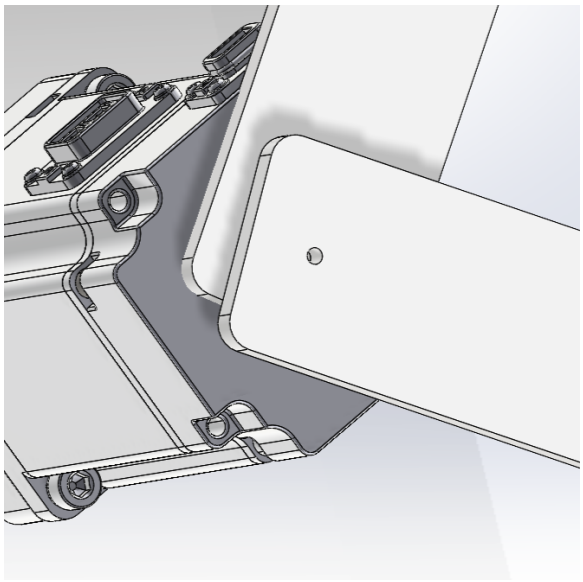
To clearly illustrate the core assembly relationships, we provide exploded views and close-up illustrations of the following key components:



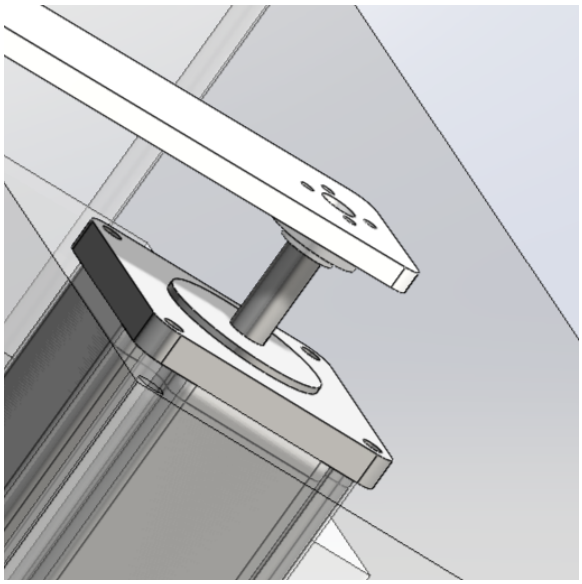
(a) 5-bar planar parallel robot (b) End-effector

Figure 4.2: Exploded view diagrams showing component arrangement.

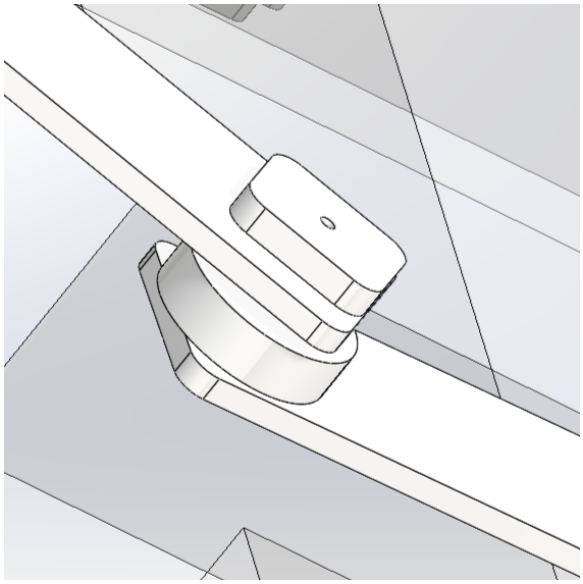
4.2 Key Connection Details



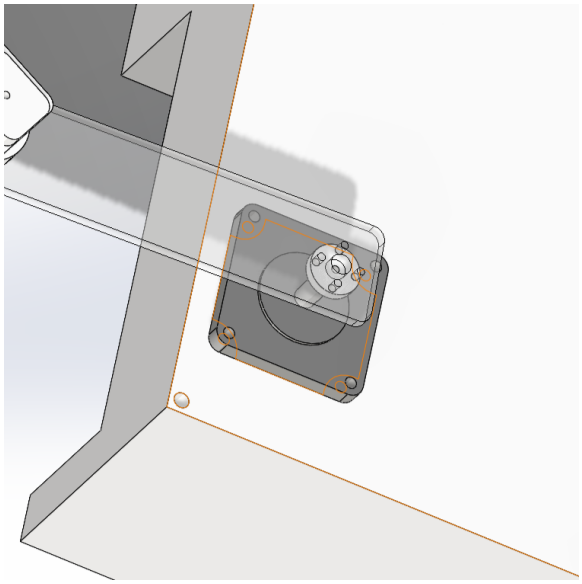
(a) Screw connection between the end-effector and the manipulator links.



(b) Link attachments to motor by 4 screws,another motor is in the same way.



(c) The connection of two links.



(d) Motor mounting.

4.3 Bill of Materials

The following table enumerates all components necessary for the complete robot assembly, classified into two categories: parts requiring machining and standard available components.

Item Number	ID	Description	number
1	base		1
2	PK569-A		2
3	Left Main		1
4	part3		2
5	part2		2
6	part1.2		1
7	part1		1
8	Left side driven		1
9	Right main		1
10	Right side driven		1
11	B1		1
12	LR1		2
13	G1		2
14	L1		4
15	L2		4
16	Motor		1
17	GE		2
18	T8 Nut 1.5		1
19	LCH0		1
20	Joint rod cap		8
21	B18.3.5M - 3 x 0.5 x 6 Socket FCHS — 6S		8
22	Joint rod		8
23	B18.3.4M - 3 x 0.5 x 8 SBHCS —S		2
24	Joint rod_3.8		4

Item Number	ID	Description	number
25	Joint rod cap 3.8		4
26	B18.3.6M - M2 x 0.4 x 6 Hex Socket Oval Pt. SS —S		4
27	washer_1		2
28	washer_2.8		2
29	lead screw		1
30	B18.3.1M - 3 x 0.5 x 16 Hex SHCS — 16SHX		4
31	Spring"Final_Assembly		1
32	base"Final_Assembly		1
33	01front flange EDA50-L_R-001-BZ		1
34	03cylinder block EDA50-L_R-003-BZ		1
35	04bearing housing EDA50-L_R-004-BZ		1
36	05threaded joint EDA50-L_R-005-BZ		1
37	06piston rod EDA50-L_R-006-BZ		1
38	10motor board EDA50-L-010-D60NH-BZ		1
39	11Coupling box EDA50-L-011-BZ		1
40	Square precision nut RN_M16×1.5P		1
41	DHS25×33×4.5"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]		1
42	Graphite copper sleeve JDB253312"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]		1

Item Number	ID	Description	number
43	spring washer-4"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]		4
44	spring washer-5"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]		16
45	hex socket-M4×40"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]	Purchased materials	4
46	hex socket-M5×16"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]	Purchased materials	4
47	hex socket-M5×20"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]	Purchased materials	4
48	hex socket-M5×25"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]	Purchased materials	4
49	hex socket-M5×35"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]	Purchased materials	4
50	SKT-12"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]		2
51	Magnetic induction switch DGN"DMC50-L01(400W)[DMC50-Motor direct connection-L05-S350-D010-K0400S3-P7]		3
52	0.4KW 60N-14 V7E-L06A-K4030-T11		1

Figure 4.4: Bill of materials for the 5-bar planar parallel robot.

4.4 Structural Static Analysis

To validate the mechanical strength of the robot design, we conducted static analysis under critical working conditions using the SolidWorks Simulation plugin.

4.4.1 Stress Analysis

The figure below shows the von Mises equivalent stress distribution under the most severe loading condition.

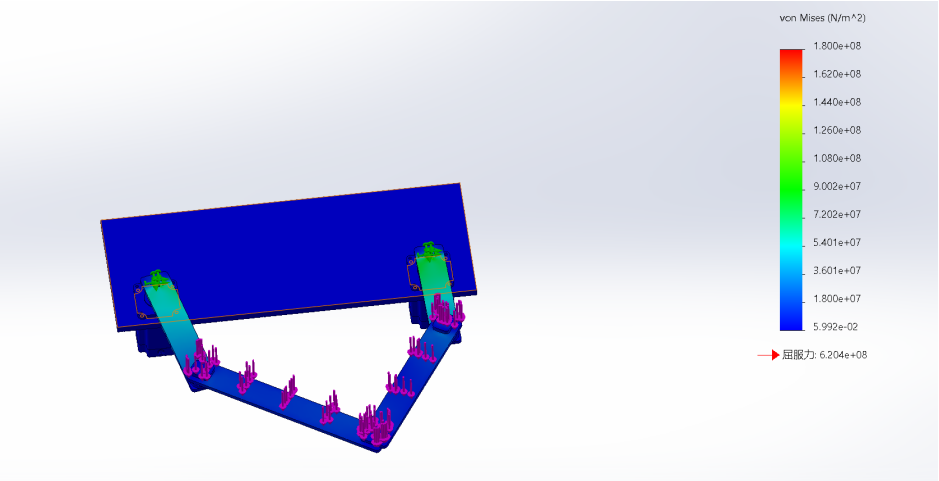


Figure 4.5: Overall von Mises stress distribution, showing a peak value of 9.005 MPa localized at the L1/L3 base joints. This confirms the structural integrity of the assembly under the specified load.

Result Interpretation: As shown in the figure, the maximum stress occurs at the base connections with links L1 and L3, with a measured peak stress value of 9.005 MPa. The structure demonstrates sufficient strength under static loading conditions, preventing plastic deformation or structural failure.

4.4.2 Displacement Analysis

The figure below displays the total displacement distribution of the model under applied loads, reflecting the overall stiffness of the robot.

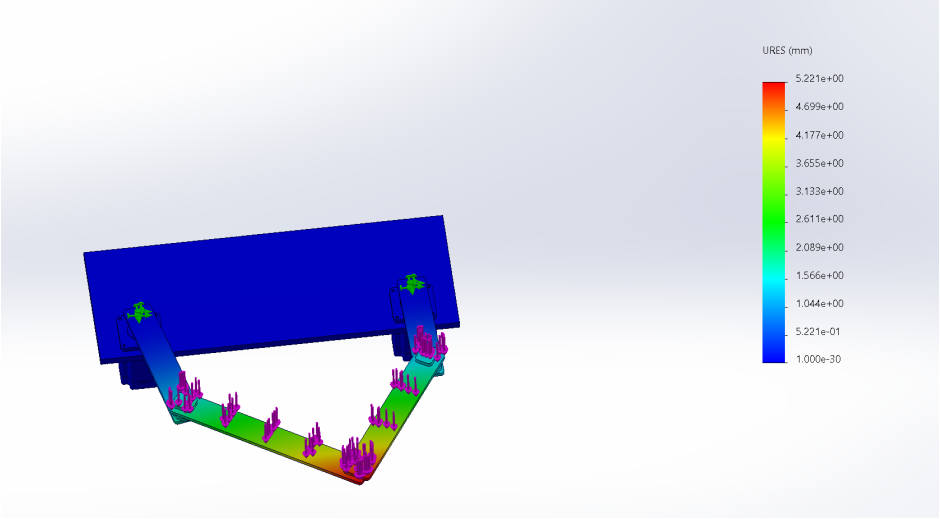


Figure 4.6: Total displacement field of the manipulator. The maximum deformation of 5.218 mm at the end-effector is small relative to the workspace, indicating sufficient overall stiffness for trajectory tracking tasks.

Result Interpretation: The maximum displacement occurs at the end-effector point P, with a value of 5.218 mm.

Stiffness Evaluation: This displacement magnitude is relatively small compared to the robot's workspace dimensions, indicating good structural stiffness that meets the requirements for general-precision trajectory tracking applications.

4.5 Conclusion

Through systematic 3D CAD design, we have developed a rationally structured five-bar linkage robot model with clear assembly relationships that satisfies kinematic requirements. The Bill of Materials provides clear guidance for subsequent procurement and manufacturing processes. Most importantly, finite element structural analysis has verified the design's feasibility and reliability from a mechanical perspective. The analysis results demonstrate that under preset loads, both stress and deformation remain within acceptable limits, ensuring the robot's safety and performance during actual operation.

A. Code

A.1 Configuration

Listing A.1: robot.m

```
function bot = robot()
bot.L0 = 20;
bot.L1 = 30;
bot.L2 = 30;
bot.L3 = 25;
bot.L4 = 30;
bot.scale = 1;
bot.base = [0,0];

bot.create_plot = @create_plot;
bot.delete_plot = @delete_plot;

end

function p = create_plot(main_ax)
p.L0_plot = plot(main_ax, NaN, NaN, 'k-', 'LineWidth', 8, 'Tag', 'robot');
p.L1_plot = plot(main_ax, NaN, NaN, 'r-', 'LineWidth', 3, 'Tag', 'robot');
p.L2_plot = plot(main_ax, NaN, NaN, 'b-', 'LineWidth', 3, 'Tag', 'robot');
p.L3_plot = plot(main_ax, NaN, NaN, 'g-', 'LineWidth', 3, 'Tag', 'robot');
p.L4_plot = plot(main_ax, NaN, NaN, 'm-', 'LineWidth', 3, 'Tag', 'robot');

p.base_point = plot(main_ax, NaN, NaN, 'ko', 'MarkerSize', 8, 'MarkerFaceColor',
    'k', 'Tag', 'robot');
p.joint1 = plot(main_ax, NaN, NaN, 'ro', 'MarkerSize', 6, 'MarkerFaceColor',
    'r', 'Tag', 'robot');
p.joint2 = plot(main_ax, NaN, NaN, 'go', 'MarkerSize', 6, 'MarkerFaceColor',
    'g', 'Tag', 'robot');
p.end_effector = plot(main_ax, NaN, NaN, 'bo', 'MarkerSize', 8,
    'MarkerFaceColor', 'k', 'Tag', 'robot');

end

function delete_plot(main_ax)
objs = findobj(main_ax, 'Tag', 'robot');
if ~isempty(objs)
    delete(objs);
end
end
```

A.2 Forward Kinematics

Listing A.2: ForwKin_5link.m

```
function joints = ForwKin_5link(q1, q2, robot)

scale = robot.scale;

% Link lengths
L0 = robot.L0 * scale;
L1 = robot.L1 * scale;
L2 = robot.L2 * scale;
L3 = robot.L3 * scale;
L4 = robot.L4 * scale;

% Base position
pos_x = robot.base(1);
pos_y = robot.base(2);

% Fixed base points
base_left  = [pos_x - L0/2, pos_y];
base_right = [pos_x + L0/2, pos_y];

% Active joint positions
joint1_pos = base_left  + L1 * [cos(q1), sin(q1)];
joint2_pos = base_right + L3 * [cos(q2), sin(q2)];

% Distance between active joints
d = norm(joint2_pos - joint1_pos);

% Reachability check
if d > (L2 + L4) || d < abs(L2 - L4)
    joints = [];
    return;
end

% Circle-circle intersection geometry
a = (L2^2 - L4^2 + d^2) / (2*d);
h = sqrt(L2^2 - a^2);

% Midpoint between the two intersection points
mid_point = joint1_pos + (a/d) * (joint2_pos - joint1_pos);

% Perpendicular vector (unit)
```

```
perp_vector = [ -(joint2_pos(2) - joint1_pos(2)), ...
                  joint2_pos(1) - joint1_pos(1) ] / d;

% Two possible end-effector positions
p1 = mid_point + h * perp_vector;
p2 = mid_point - h * perp_vector;

% Elbow-up selection using cross product
% Vector from left active joint -> right active joint
v = joint2_pos - joint1_pos;

% Candidate vectors from midpoint -> p1/p2
w1 = p1 - mid_point;

% 2D cross product z component
cross1 = v(1)*w1(2) - v(2)*w1(1);

% Elbow-up is traditionally defined as cross > 0
if cross1 > 0
    end_effector_pos = p1;
else
    end_effector_pos = p2;
end

% Output joints in consistent order
joints = [
    base_left;
    base_right;
    joint1_pos;
    joint2_pos;
    end_effector_pos
];
end
```

Listing A.3: draw_workspace.m

```
function draw_workspace(ax, robot)

q1_range = linspace(0, pi, 50);  % 0-180
q2_range = linspace(0, pi, 50);  % 0-180

workspace_points = [];

for q1 = q1_range
    for q2 = q2_range
        joints = ForwKin_5link(q1, q2, robot);
        if length(joints) == 5
```

```
            workspace_points = [workspace_points; joints(5, :)];
        end
    end
end

workspace_objs = findobj(ax, 'Tag', 'workspace');
if ~isempty(workspace_objs)
    delete(workspace_objs);
end


if ~isempty(workspace_points)
    sc = scatter(ax, workspace_points(:,1), workspace_points(:,2), 10, ...
        [0.7 0.8 1.0], 'filled', 'MarkerEdgeAlpha', 0.3, ...
        'MarkerFaceAlpha', 0.4, 'Tag', 'workspace');
    uistack(sc, 'bottom');

    [p, found] = find_rectangle_position(workspace_points, robot);
    if found
        plot_max_rectangle(ax, p);
    end
end

if size(workspace_points, 1) > 2
    try
        k = convhull(workspace_points(:,1), workspace_points(:,2));
        p = plot(ax, workspace_points(k,1), workspace_points(k,2), 'c-', ...
            'LineWidth', 1.5, 'Tag', 'workspace');
        uistack(p, 'bottom');
    catch
        p = plot(ax, workspace_points(:,1), workspace_points(:,2), 'c-', ...
            'LineWidth', 1.5, 'Tag', 'workspace');
        uistack(p, 'bottom');
    end
end
end

function [rect_position, found] = find_rectangle_position(workspace_points,
    robot)
if isempty(workspace_points)
    rect_position = [];
    return;
end

pos_x = robot.base(1);

found = false;
```

```
58
59  x_min = min(workspace_points(:,1));
60  x_max = max(workspace_points(:,1));
61  y_min = min(workspace_points(:,2));
62  y_max = max(workspace_points(:,2));
63
64  x_range1 = linspace(pos_x, x_max, floor(x_max-pos_x));
65  x_range2 = linspace(pos_x, x_min, floor(pos_x-x_min));
66  x_range = [x_range1, x_range2];
67  [~, sort_idx] = sort(abs(x_range- pos_x));
68  x_range = x_range(sort_idx);
69
70  y_range = linspace(y_min , y_max , floor(y_max-y_min));
71
72  rect_position = [];
73  for x = x_range
74      for y = y_range
75          pass = true;
76          points = tray_key_points([x, y]);
77          for i = 1:size(points, 1)
78              [~, ~, valid] = InvKin_5link(points(i, :), robot);
79              if ~valid
80                  pass = false;
81                  break;
82              end
83          end
84
85          if pass == true
86              found = true;
87              rect_position=[x, y];
88              return;
89          end
90      end
91  end
92  end
93
94
95  function plot_max_rectangle(ax, p)
96
97  x_center = p(1); y_center = p(2);
98  width = 30; height = 25;
99
100 x_corners = [x_center - width/2, x_center + width/2, x_center + width/2,
        x_center - width/2, x_center - width/2];
101 y_corners = [y_center - height/2, y_center - height/2, y_center + height/2,
        y_center + height/2, y_center - height/2];
102
```

```
plot(ax, x_corners, y_corners, 'r-', 'LineWidth', 3, 'Tag', 'workspace');
fill(ax, x_corners, y_corners, 'r', 'FaceAlpha', 0.2, 'EdgeColor', 'r', 'Tag',
    'workspace');

points = tray_key_points(p);
for i = 1:size(points, 1)
    point = points(i, :);
    plot(ax, point(1), point(2), 'r+', 'MarkerSize', 6, 'LineWidth', 2, 'Tag',
        'workspace');
end

text(ax, x_center, y_center - height / 2 - 2, sprintf('%.0fx%.0f', width,
    height), ...
    'HorizontalAlignment', 'center', 'FontWeight', 'bold', 'Color', 'red', ...
    'Tag', 'workspace');
end
```

Listing A.4: draw_5link.m

```
function draw_5link(robot, robot_plot, q1, q2)

joints = ForwKin_5link(q1, q2, robot);

L0_plot = robot_plot.L0_plot;
L1_plot = robot_plot.L1_plot;
L2_plot = robot_plot.L2_plot;
L3_plot = robot_plot.L3_plot;
L4_plot = robot_plot.L4_plot;

base_point = robot_plot.base_point;
joint1 = robot_plot.joint1;
joint2 = robot_plot.joint2;
end_effector = robot_plot.end_effector;

if length(joints) < 5
    set(L1_plot, 'XData', NaN, 'YData', NaN);
    set(L2_plot, 'XData', NaN, 'YData', NaN);
    set(L3_plot, 'XData', NaN, 'YData', NaN);
    set(L4_plot, 'XData', NaN, 'YData', NaN);
    set(L0_plot, 'XData', NaN, 'YData', NaN);
    return;
end

base_left = joints(1, :);
base_right = joints(2, :);
joint1_pos = joints(3, :);
joint2_pos = joints(4, :);
```

```
end_effector_pos = joints(5, :);

set(L0_plot, 'XData', [base_left(1), base_right(1)], ...
    'YData', [base_left(2), base_right(2)]);

set(L1_plot, 'XData', [base_left(1), joint1_pos(1)], ...
    'YData', [base_left(2), joint1_pos(2)]);

set(L3_plot, 'XData', [base_right(1), joint2_pos(1)], ...
    'YData', [base_right(2), joint2_pos(2)]);

set(L2_plot, 'XData', [joint1_pos(1), end_effector_pos(1)], ...
    'YData', [joint1_pos(2), end_effector_pos(2)]);

set(L4_plot, 'XData', [joint2_pos(1), end_effector_pos(1)], ...
    'YData', [joint2_pos(2), end_effector_pos(2)]);

set(base_point, 'XData', [base_left(1), base_right(1)], ...
    'YData', [base_left(2), base_right(2)]);
set(joint1, 'XData', joint1_pos(1), 'YData', joint1_pos(2));
set(joint2, 'XData', joint2_pos(1), 'YData', joint2_pos(2));
set(end_effector, 'XData', end_effector_pos(1), 'YData', end_effector_pos(2));

drawnow;
end
```

Listing A.5: tray_key_points.m

```
function points = tray_key_points(center)
width = 30;
height = 25;
side = 5;

i = center(1);
j = center(2);

points = zeros(30, 2);
index = 1;
for offset_x = 0:5
    x = i - width/2 + side/2 + offset_x * side;
    for offset_y = 0:4
        y = j - height/2 + side/2 + offset_y * side;
        points(index, :) = [x, y];
        index = index + 1;
    end
end
end
```

Listing A.6: main_forwKin.m

```
function create_ui(ax, height, robot, bot_plot)

x = 700;
y = height - 240;

q1 = 20;
q2 = 20;

q1_text = uicontrol('Style', 'text', 'Position', [x, y, 120, 20], ...
    'String', 'Q1:', 'FontSize', 10);
q1_slider = uicontrol('Style', 'slider', 'Position', [x + 120, y, 120, 20], ...
    'Min', 0, 'Max', 180, 'Value', q1);

y = y - 40;
q2_text = uicontrol('Style', 'text', 'Position', [x, y, 120, 20], ...
    'String', 'Q2:', 'FontSize', 10);
q2_slider = uicontrol('Style', 'slider', 'Position', [x+120, y, 120, 20], ...
    'Min', 0, 'Max', 180, 'Value', q2);

y = y - 40;
uicontrol('Style', 'text', 'Position', [x, y, 120, 20], ...
    'String', 'Base X:', 'FontSize', 10);
base_x_slider = uicontrol('Style', 'slider', 'Position', [x+120, y, 120, 20], ...
    'Min', -50, 'Max', 50, 'Value', 0);

y = y - 40;
uicontrol('Style', 'text', 'Position', [x, y, 120, 20], ...
    'String', 'Base Y:', 'FontSize', 10);
base_y_slider = uicontrol('Style', 'slider', 'Position', [x+120, y, 120, 20], ...
    'Min', -50, 'Max', 50, 'Value', 0);

y = y - 40;
uicontrol('Style', 'text', 'Position', [x, y, 120, 20], ...
    'String', 'scale:', 'FontSize', 10);
scale_slider = uicontrol('Style', 'slider', 'Position', [x+120, y, 120, 20], ...
    'Min', 0.5, 'Max', 2, 'Value', 1);

set(base_x_slider, 'Callback', @(src,evt) update_plot());
set(base_y_slider, 'Callback', @(src,evt) update_plot());
set(q1_slider, 'Callback', @(src,evt) update_plot());
set(q2_slider, 'Callback', @(src,evt) update_plot());
set(scale_slider, 'Callback', @(src,evt) update_plot());
    function update_plot()
        pos_x = get(base_x_slider, 'Value');
        pos_y = get(base_y_slider, 'Value');
```

```
        q1_deg = get(q1_slider, 'Value');
        q2_deg = get(q2_slider, 'Value');
        scale = get(scale_slider, 'Value');

        set(q1_text, 'String', sprintf('Q1:%.2f', q1_deg));
        set(q2_text, 'String', sprintf('Q2:%.2f', q2_deg));

        q1 = deg2rad(q1_deg);
        q2 = deg2rad(q2_deg);

        robot.base = [pos_x, pos_y];
        robot.scale = scale;

        draw_workspace(ax, robot);
        draw_5link(robot, bot_plot, q1, q2);
    end
update_plot();
end


close all;
clear;
clc;

robot = robot();

width = 1000;
height = 600;

figure('Position', [100, 100, width, height], 'Name', '5-link simulation');

main_ax = subplot('Position', [0.05, 0.1, 0.65, 0.8]);
axis equal;
grid on;
hold on;
xlabel('X');
ylabel('Y');
title('5-link simulation');

axis_limit = 60;
xlim([-axis_limit, axis_limit]);
ylim([-10, axis_limit]);

legend_ax = subplot('Position', [0.75, 0.7, 0.2, 0.2]);
axis off;
title('configuration');

```

```
bot_plot = robot.create_plot(main_ax);

link_params = {
    sprintf('L0 = %d (black)', robot.L0) ...
    sprintf('L1 = %d (red)', robot.L1), ...
    sprintf('L2 = %d (blue)', robot.L2), ...
    sprintf('L3 = %d (green)', robot.L3), ...
    sprintf('L4 = %d (magenta)', robot.L4), ...
    };

text(legend_ax, 0.1, 0.9, link_params, 'FontSize', 12, 'VerticalAlignment',
    'top', ...
    'FontName', 'SimHei');

create_ui(main_ax, height, robot, bot_plot);
```

A.3 Inverse Kinematics

Listing A.7: InvKin_5link.m

```
function [q1, q2, valid] = InvKin_5link(end_effector, robot)

pos_x = robot.base(1);
pos_y = robot.base(2);
scale = robot.scale;

valid = false;
px = end_effector(1);
py = end_effector(2);

% --- Link lengths ---
L0 = robot.L0 * scale;
L1 = robot.L1 * scale;
L2 = robot.L2 * scale;
L3 = robot.L3 * scale;
L4 = robot.L4 * scale;

% --- Base anchor points ---
base_left  = [pos_x - L0/2, pos_y];
base_right = [pos_x + L0/2, pos_y];

% --- Step 1: joint1 candidates ---
[P1, P2, ok1] = circle_intersection(base_left, L1, [px py], L2);
if ~ok1
    q1=[]; q2=[]; return;
```

```
end

% --- Step 2: joint2 candidates ---
[Q1, Q2, ok2] = circle_intersection(base_right, L3, [px py], L4);
if ~ok2
    q1=[]; q2=[]; return;
end

% --- Try the 4 possible combinations to find elbow-up ---
candidates = [
    P1; Q1;
    P1; Q2;
    P2; Q1;
    P2; Q2
];

found = false;
best_P = [];
best_Q = [];

for i = 1:4
    P = candidates(2*i-1,:);
    Q = candidates(2*i,:);

    % Ensure passive links L2 and L4 can connect
    d = norm(P - Q);
    if d > L2 + L4 || d < abs(L2 - L4)
        continue;
    end

    % Elbow-up check: cross(j1->j2, j1->end)
    v = Q - P;
    w = [px py] - P;
    cross_val = v(1)*w(2) - v(2)*w(1);

    if cross_val > 0   % elbow up
        best_P = P;
        best_Q = Q;
        found = true;
        break;
    end
end

if ~found
    q1=[]; q2=[]; return;
end

```

```
joint1 = best_P;
joint2 = best_Q;

% --- Compute joint angles ---
q1 = atan2(joint1(2)-base_left(2),  joint1(1)-base_left(1));
q2 = atan2(joint2(2)-base_right(2), joint2(1)-base_right(1));

% --- Angle limits (0~180 deg) ---
if q1 < 0 || q1 > pi || q2 < 0 || q2 > pi
    q1=[]; q2=[];
    valid = false;
    return;
end

valid = true;
end




%% ------------------------- Circle intersection -------------------------
function [p1, p2, ok] = circle_intersection(c1, r1, c2, r2)
d = norm(c2 - c1);
if d > r1 + r2 || d < abs(r1 - r2)
    p1=[]; p2=[]; ok=false;
    return;
end

a = (r1^2 - r2^2 + d^2) / (2*d);
h = sqrt(max(r1^2 - a^2, 0));

mid = c1 + a * (c2 - c1) / d;
offset = h * [-(c2(2)-c1(2))/d, (c2(1)-c1(1))/d];

p1 = mid + offset;
p2 = mid - offset;
ok = true;
end
```

Listing A.8: InvKin_trajectory.m

```
function angles = InvKin_trajectory(trajectory, robot)
angles = zeros(length(trajectory), 2);
index = 1;
for point = trajectory
    [q1, p1, result] = InvKin_5link(point, robot);
    if result
        angles(index, :) = [q1, p1];
```

```
        index = index + 1;
    end
end
end
```

Listing A.9: draw_trajectory.m

```
function draw_trajectory(ax, robot, traj)

angles = InvKin_trajectory(traj, robot);
num_points = size(angles, 1);

j_traj = zeros(num_points, 2, 3);
index = 1;

robot.delete_plot(ax);
robot_plot = robot.create_plot(ax);

for i = 1:num_points
    q1 = angles(i,1);
    q2 = angles(i,2);
    joints = ForwKin_5link(q1, q2, robot);
    j_traj(index, :, 1) = joints(3, :);
    j_traj(index, :, 2) = joints(4, :);
    j_traj(index, :, 3) = joints(5, :);
    index = index + 1;

    draw_5link(robot, robot_plot, q1, q2);

    left_joint = j_traj(:, :, 1);
    right_joint = j_traj(:, :, 2);
    end_joint = j_traj(:, :, 3);

    objs = findobj(ax, 'Tag', 'trajectory');
    if ~isempty(objs)
        delete(objs);
    end
    plot(ax, left_joint(:,1), left_joint(:,2), 'ko', 'MarkerSize', 1,
        'MarkerFaceColor', 'k', 'Tag', 'trajectory');
    plot(ax, right_joint(:,1), right_joint(:,2), 'ko', 'MarkerSize', 1,
        'MarkerFaceColor', 'k', 'Tag', 'trajectory');
    plot(ax, end_joint(:,1), end_joint(:,2), 'ko', 'MarkerSize', 1,
        'MarkerFaceColor', 'k', 'Tag', 'trajectory');
    pause(0.1);
end
end
```

Listing A.10: main_invKin.m

```
close all;
clear;
clc;

robot = robot();

width = 1600;
height = 1200;

figure('Position', [200, -10, width, height], 'Name', '5-link trajectory
    simulation group 2');

ax = subplot(1,1,1);
axis equal;
grid on;
hold on;
xlabel('X');
ylabel('Y');
title('5-link trajectory simulation group 2');

xlim([-30, 30]);
ylim([-5, 50]);

% draw circle
t = 0:1:50;
x_traj = 1*cos(2*pi*t/50);
y_traj = 1*sin(2*pi*t/50) + 40;
traj = [x_traj; y_traj];

draw_trajectory(ax, robot, traj);

% draw rectangle
x_traj = [linspace(1, 1, 10), linspace(1, -1, 20), linspace(-1, -1, 20),
    linspace(-1, 1, 20), linspace(1, 1, 10)];
y_traj = [linspace(40, 41, 10), linspace(41, 41, 20), linspace(41, 39, 20),
    linspace(39, 39, 20), linspace(39, 40, 10)];
traj = [x_traj; y_traj];
draw_trajectory(ax, robot, traj);
```

B. Group Member Contribution and Peer Review

Group Member Contribution and Peer Review Form

Group Number ___2___
As a group, briefly list each member's actual work in Table 1 for the group project. In Table 2, indicate the extent for each member's contributions.

Table1 Member contributions

	Group member candidate number	Group member candidate number	Group member candidate number	Group member candidate number
	299089	299081	299078	299100
Contributions (Actual work)	Derive and simulate the forward kinematics formula	End-effector design and CAD modeling	Derive and simulate the inverse kinematics formula	5-bar planar parallel robot design and CAD modeling

Table 2 Peer review form

	Group member candidate number	Group member candidate number	Group member candidate number	Group member candidate number
	299089	299081	299078	299100
Technical Contributions (100%). Contributes significantly to the success of the project. Contributes meaningfully to group discussions. For example, programming, derivations, CAD modeling.	100	100	100	100
Contributions to Deliverables (100%). Completes group assignments on time. Prepares work in a quality manner. For example, report writing, preparing figures.	100	100	100	100
Collaboration (100%). Attends group meetings regularly and arrives on time. Demonstrates a cooperative and supportive attitude.	100	100	100	100
Average Score	100	100	100	100