Robotic Kinematic Analysis Report

Naghim Ibragimov
K22031784
k22031784@kcl.ac.uk

Department of Engineering
King's College London

20 November 2025

Contents

1 Abstract **1**

Abstract **1**

2 Introduction **1**

3 Background and Literature Review **2**

4 Methodologies of the Tasks **3**
4.1 Terminology and Definitions 3
4.2 Task 1: Method 3
4.3 Task 2: Method 4
4.4 Task 3: Method 4
4.5 Methodology Summary 4

5 Work and Results of the Tasks **5**
5.1 Task 1: Working out and Solution 5
5.1.1 Robot Graph 5
5.1.2 Forward Kinematics 5
5.1.3 Inverse Kinematics 6
5.1.4 Jacobian and Singularity Analysis 7
5.2 Task 2: Working out and Solution 7
5.2.1 Part (1): DH Table 7
5.2.2 Part (2): Forward Kinematics 8
5.2.3 Angle Sum Identities 8
5.2.4 Final Forward Kinematics 8
5.2.5 Part (3): Inverse Kinematics 9
5.2.6 Part (4): Jacobian of the Manipulator 9
5.3 Task 3: Working Out and Solution 10
5.3.1 (1) XML Inspection and Sketch of the Robot 10
5.3.2 (2) End–Effector Target Experiments and Analysis 11

6 Conclusion **12**

References **13**

A Appendix **14**

1 Abstract

The objective of this report is to understand fundamentals in robot kinematics by analysing and testing the theory through analytical derivations and simulation. Denavit–Hartenberg conventions were applied to formulate the forward and inverse kinematics of the manipulators. Jacobian matrices were derived to analyse the end-effector motion and also to identify singular configurations. A MuJoCo simulation of a robot defined by a URDF model provided a realistic validation of the theoretical results and demonstrated how to build and manipulate URDF descriptions for real robots. The key findings include identification of singularities and improved understanding of how to construct URDF files and extract transformation matrices for an actual robot.

2 Introduction

Robot kinematics is the study of the motion of robot manipulators without considering the forces that cause the motion. Forward kinematics determines the end-effector pose from given joint variables, whereas inverse kinematics solves for the joint variables that achieve a desired pose. The Denavit–Hartenberg (DH) provides a systematic way to assign coordinate and derive homogeneous transformation matrices between links. The geometric Jacobian, is the partial derivative of the end-effector pose with respect to joint variables, captures the differential relationship between joints and the linear and angular parts of the end effector. The Jacobian is also used to identify singularities, where its either multiple DOF or undefined.

The main aim of this coursework is to demonstrate understanding of robotic kinematics by deriving and validating forward and inverse kinematic models, constructing Jacobians, and analysing them accordingly. In addition, this work evaluates how theoretical kinematic models can be verified using modern simulation tools such as MuJoCo and URDF-based robot descriptions. The report therefore combines theoretical derivation, varification (MATLAB/Python), and practical simulation as a complete workflow for robot analysis.

The coursework comprises three tasks that progressively build upon each other. Task 1 involves a planar 3R arm; using the classical DH table, the homogeneous transformation matrices from the base to the end effector are derived, inverse kinematics is solved for arbitrary end-effector positions, and the Jacobian is computed to analyse singularities. Task 2 is slightly more difficult, because of many Z-angles and lengths, making slightly more difficult to analyse, however the idea of solving is the same. Finally, Task 3 is all about getting used to wroking with URDF and seeing how robot is working with computer commands.

Computational tools were employed to support the analysis such as MATLAB for checking the pots and doing some maths. Python (Visual Studio) scripts were used for working with the Task 3, which is the URDF files and moving the robot arm to Target Poisition.

Overall, the report aims to connect theoretical robot kinematics with practical experience.

The following section provides the theoretical and literature background necessary to understand these methodologies and their relevance to modern robotic systems.

3 Background and Literature Review

Robot kinematics shows the mathematical idea for describing and analysing robotic motion. Unlike dynamics, which studies forces and torques, kinematics focuses solely on the relationships between robot links and joints. Central concepts include forward kinematics(FK), which essential first step to determine the pose; Inverse Kinematics (IK), which computes the joint values required to reach a desired pose; and the Jacobian, which relates joint velocities to end effector velocities and is crucial for studying singularities. A standard way to show the relationship is the Denavit Hartenberg (DH) convention, which assigns coordinate frames to links using four parameters: joint angle, link offset, link length, and twist angle. With these parameters, homogeneous transformation matrices can be constructed to derive FK, IK and the Jacobian. These principles are essential to solve first 2 tasks.

The maths methods used in this report are mainly derived from material learned from famous textbooks. Craig's *Introduction to Robotics: Mechanics and Control* and Corke's *Robotics, Vision and Control*. These provide the methods for constructing DH tables (both classical and improved) and explain other essential robot kinematic topics such as FK, IK, and Jacobians, all of which guided the analytical work in Tasks1 and2. The King's College London robotics lectures further supported the practical methodology, particularly in the understanding of DH tables, and the math behind the topics needed for the coursework. Online resources such as Brian Douglas's YouTube series *Robotics: Kinematics and Dynamics* and Peter Corke's *MATLAB Robotics Toolbox* tutorials were used to check parts of the mathematical work and to help visualise the sketch in Task 1.

Beyond analytical solutions, robotic simulation is essential for checking kinematic results. This motivates the use of the Unified Robot Description Format (URDF), an XML-based modelling standard for representing the geometry and structure of a robot. Tola and Corke's 2024 paper, *"Understanding URDF: A Dataset and Analysis"*, demonstrated how URDF encodes link and joint relationships. These were directly applied in Task 3 when inspecting the UR5 XML model imported into MuJoCo. To understand the MuJoCo and some python libraries that were used, following tutorials were used as practice. *MuJoCo URDF Modelling Guide* (Deep Robotics Lab) and *Building Robots in MuJoCo* (Stanford AI Robotics Group).

Overall, the reviewed literature and theoretical foundations established a consistent working path for the coursework: from mathematical derivations in Tasks 1 and 2, to simulation in Task 3. This coursework gave me a complete understanding of how analytic kinematics extends into modern robotic simulation environments such as MuJoCo.

4 Methodologies of the Tasks

4.1 Terminology and Definitions

Before outlining the methodologies, several key terms are defined for clarity:

- **Denavit–Hartenberg (DH) Parameters** — a standard scheme for describing a robot geometry using four parameters per link: link length a_i, link twist α_i, link offset d_i, and joint angle θ_i.
- **Forward Kinematics (FK)** — mapping known joint variables to the end-effector pose by multiplying homogeneous transformations.
- **Inverse Kinematics (IK)** — computing joint variables that give an end-effector pose.
- **Jacobian Matrix** — the matrix of partial derivatives that relates joint-space velocities $\dot{\mathbf{q}}$ to end-effector linear velocities $\dot{\mathbf{x}}$; i.e., $\dot{\mathbf{x}} = J(\mathbf{q})\,\dot{\mathbf{q}}$.
- **Singularity** — (e.g., $\det(J) = 0$), .
- **URDF (Unified Robot Description Format)** — an XML-based format for describing robot links, joints, and frames for simulation/visualisation.
- **XML (Extensible Markup Language)** — a structured text format; URDF encodes robot structure using XML tags.

4.2 Task 1: Method

The methodology begins by **sketching the robot arm** from the provided DH parameters. Each joint axis is drawn according to the DH table:

The hand-drawn sketch is **verified in MATLAB** (Robotics System Toolbox) by constructing a rigid-body model from the DH table.

Forward kinematics (FK) are then formulated by multiplying each homogeneous transform T_{i-1}^{i} to obtain the overall map T_0^3. The **end-effector position vector** is the last column of T_0^3, expressed in terms of $\mathbf{q} = [q_1, q_2, q_3]^\top$.

Using this 3×1 position vector, the **Jacobian** is derived by partial differentiation:

$$J(\mathbf{q}) \;=\; \begin{bmatrix} \frac{\partial x}{\partial q_1} & \frac{\partial x}{\partial q_2} & \frac{\partial x}{\partial q_3} \\ \frac{\partial y}{\partial q_1} & \frac{\partial y}{\partial q_2} & \frac{\partial y}{\partial q_3} \\ \frac{\partial z}{\partial q_1} & \frac{\partial z}{\partial q_2} & \frac{\partial z}{\partial q_3} \end{bmatrix}.$$

Singularities are identified by testing when $\det(J) = 0$ (equivalently, when the Jacobian columns become linearly dependent).

4.3 Task 2: Method

For Task 2, the DH table is constructed from the picture given in the coursework, then the workflow mirrors Task 1:

1. Build the **DH table** from the image.
2. Derive **FK** via $T_0^3 = T_0^1 T_1^2 T_2^3$.
3. Obtain **IK** by algebraic manipulation of the FK equations (using standard trigonometric identities).
4. Compute the **Jacobian** by partial differentiation of the end-effector position.

In this task, explicit singularity calculation (e.g., $\det(J) = 0$) is **not required**.

4.4 Task 3: Method

1. **Model selection and import.** I selected the *UR5 6-DoF* arm from the MuJoCo repository and loaded the provided XML (MJCF) with meshes. Units are metres and angles are radians (`<compiler angle="radian">`).
2. **XML inspection and sketch.** I inspected the `<body>` hierarchy and `<joint>` definitions to identify link order, joint types and axes. Based on this, I produced a labelled sketch/screenshot showing base frame, joints J1–J6 and the end-effector (EE).
3. **End–effector positioning.** I moved the arm to a Cartesian target $[a, b, c]^\top$ in the viewer (position only, orientation not enforced). I recorded the reached pose and the joint configuration.
4. **Conceptual analysis.** I explained FK/IK/possible singularities qualitatively—no symbolic derivations—linking observations to the model structure.

4.5 Methodology Summary

Across all tasks, the procedure is consistent:

1. Define or refine **DH parameters** and frame assignments.
2. Derive **forward kinematics** by multiplying homogeneous transforms.
3. Solve **inverse kinematics** by algebraic rearrangement of FK relations.
4. Construct the **Jacobian** via partial differentiation of the end-effector position.
5. (Where applicable) Determine **singularities** from $\det(J) = 0$.
6. Connect analytical models to **URDF/XML** for simulation.

5 Work and Results of the Tasks

5.1 Task 1: Working out and Solution

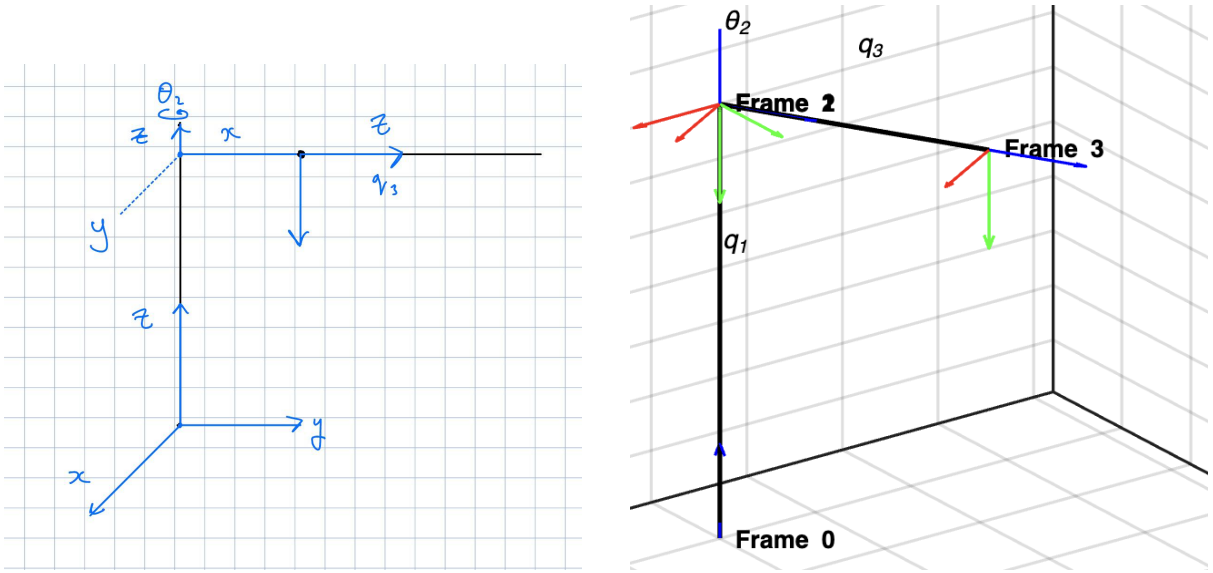
The manipulator for Task 1 is described by the following Denavit–Hartenberg parameters:

Table 1: D–H parameters for Task 1

i	θ_i	d_i	a_i	α_i
1	0	q_1	0	0
2	q_2	0	0	$-90°$
3	0	q_3	0	0

$$^0T_3 = A_1 A_2 A_3.$$

5.1.1 Robot Graph



(a) Hand sketch from D–H parameters (paper drawing).

(b) MATLAB verification with frames shown.

Figure 1: Task 1 robot graph: manual sketch and MATLAB visual verification.

5.1.2 Forward Kinematics

Using the standard DH formulation, the three link transforms are obtained from the table.

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_2 = \begin{bmatrix} \cos q_2 & 0 & -\sin q_2 & 0 \\ \sin q_2 & 0 & \cos q_2 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here, A_1 is a pure translation along z, A_2 is a rotation by q_2 about z with a -90° twist about x, and A_3 is a translation along z by q_3. The overall forward kinematics is

$$^0T_3 = A_1 A_2 A_3.$$

Multiplying Ai matrices gives the final answer, last column is the end-effector that will be used for inverse kinematics

$$^0T_3 = \begin{bmatrix} \cos q_2 & 0 & -\sin q_2 & -q_3 \sin q_2 \\ \sin q_2 & 0 & \cos q_2 & q_3 \cos q_2 \\ 0 & -1 & 0 & q_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{p}_{ee} = \begin{bmatrix} -q_3 \sin q_2 \\ q_3 \cos q_2 \\ q_1 \end{bmatrix}.$$

5.1.3 Inverse Kinematics

Let the target end-effector position be $\mathbf{p}_{ee} = [a,\, b,\, c]^T$. From the FK expression,

$$a = -q_3 \sin q_2, \qquad b = q_3 \cos q_2, \qquad c = q_1.$$

Step 1: Use $\sin^2 q_2 + \cos^2 q_2 = 1$. Dividing by q_3 (assuming $q_3 \neq 0$) and square-adding,

$$\left(\frac{a}{q_3}\right)^2 + \left(\frac{b}{q_3}\right)^2 = 1 \quad \Rightarrow \quad \boxed{q_3 = \sqrt{a^2 + b^2} \equiv r \geq 0.}$$

Step 2: Recover q_2. With $q_3 = r$,

$$\sin q_2 = -\frac{a}{r}, \qquad \cos q_2 = \frac{b}{r},$$

hence

$$\boxed{q_2 = \operatorname{atan2}(-a,\, b).}$$

Step 3: Recover q_1.

$$\boxed{q_1 = c.}$$

Final IK (principal branch).

$$\boxed{q_1 = c, \qquad q_3 = \sqrt{a^2 + b^2}, \qquad q_2 = \operatorname{atan2}(-a,\, b).}$$

5.1.4 Jacobian and Singularity Analysis

From

$$x = -q_3 \sin q_2, \quad y = q_3 \cos q_2, \quad z = q_1,$$

the analytical Jacobian is

$$J(\mathbf{q}) = \begin{bmatrix} \partial x/\partial q_1 & \partial x/\partial q_2 & \partial x/\partial q_3 \\ \partial y/\partial q_1 & \partial y/\partial q_2 & \partial y/\partial q_3 \\ \partial z/\partial q_1 & \partial z/\partial q_2 & \partial z/\partial q_3 \end{bmatrix}.$$

Entry-by-entry:

$$\begin{aligned} x_{q_1} &= 0, \quad x_{q_2} = -q_3 \cos q_2, \quad x_{q_3} = -\sin q_2, \\ y_{q_1} &= 0, \quad y_{q_2} = -q_3 \sin q_2, \quad y_{q_3} = \cos q_2, \\ z_{q_1} &= 1, \quad z_{q_2} = 0, \quad z_{q_3} = 0. \end{aligned}$$

Thus

$$\boxed{J(\mathbf{q}) = \begin{bmatrix} 0 & -q_3 \cos q_2 & -\sin q_2 \\ 0 & -q_3 \sin q_2 & \cos q_2 \\ 1 & 0 & 0 \end{bmatrix}.}$$

Singularity condition.

$$\det(J) = \begin{vmatrix} 0 & -q_3 \cos q_2 & -\sin q_2 \\ 0 & -q_3 \sin q_2 & \cos q_2 \\ 1 & 0 & 0 \end{vmatrix} = -q_3(\cos^2 q_2 + \sin^2 q_2) = \boxed{-q_3}.$$

Hence the manipulator is singular when

$$\boxed{q_3 = 0.}$$

5.2 Task 2: Working out and Solution

5.2.1 Part (1): DH Table

Using the Classic Denavit–Hartenberg convention, the parameters are:

i	θ_i	d_i	a_i	α_i
1	θ_1	0	l_1	$\frac{\pi}{2}$
2	θ_2	0	l_2	0
3	θ_3	0	0	0

These parameters follow directly from the geometry: link 1 contains the 90° twist, link 2 has link length l_2, and link 3 is a pure revolute joint without any physical link.

5.2.2 Part (2): Forward Kinematics

The homogeneous transformation for each link using the DH parameters is:

Transformation A_1:

$$A_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & l_1\cos\theta_1 \\ \sin\theta_1 & 0 & -\cos\theta_1 & l_1\sin\theta_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & 0 \\ \sin\theta_3 & \cos\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.2.3 Angle Sum Identities

Since terms such as $\cos(\theta_2+\theta_3)$ and $\sin(\theta_2+\theta_3)$ appear in the final result, we recall the standard trigonometric identities:

$$\cos(A+B) = \cos A\cos B - \sin A\sin B, \qquad \sin(A+B) = \sin A\cos B + \cos A\sin B.$$

5.2.4 Final Forward Kinematics

The end-effector transform is obtained by:

$$T_0^3 = A_1A_2A_3.$$

After symbolic multiplication (details omitted for clarity), the closed-form forward kinematics expression is:

$$T_0^3 = \begin{bmatrix} \cos\theta_1\ \cos(\theta_2+\theta_3) & -\cos\theta_1\ \sin(\theta_2+\theta_3) & \sin\theta_1 & (l_1+l_2\cos\theta_2)\cos\theta_1 \\ \sin\theta_1\ \cos(\theta_2+\theta_3) & -\sin\theta_1\ \sin(\theta_2+\theta_3) & -\cos\theta_1 & (l_1+l_2\cos\theta_2)\sin\theta_1 \\ \sin(\theta_2+\theta_3) & \cos(\theta_2+\theta_3) & 0 & l_2\sin\theta_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

This expression gives the full forward kinematics of the Task 2 three-link manipulator.

5.2.5 Part (3): Inverse Kinematics

Let the desired end-effector position be $(a, b, c) \in R^3$. From the forward kinematics,

$$a = (l_1 + l_2 \cos\theta_2)\cos\theta_1, \quad b = (l_1 + l_2 \cos\theta_2)\sin\theta_1, \quad c = l_2 \sin\theta_2.$$

Solving for θ_1. From the first two equations,

$$\boxed{\theta_1 = \operatorname{atan2}(b, a)}.$$

Solving for θ_2. Define the horizontal distance $r = \sqrt{a^2 + b^2}$. Then

$$r = l_1 + l_2 \cos\theta_2, \qquad c = l_2 \sin\theta_2,$$

hence

$$\cos\theta_2 = \frac{r - l_1}{l_2}, \qquad \sin\theta_2 = \frac{c}{l_2}.$$

Using $\sin^2\theta_2 + \cos^2\theta_2 = 1$ gives the reachability condition

$$(r - l_1)^2 + c^2 = l_2^2.$$

Thus,

$$\boxed{\theta_2 = \operatorname{atan2}\left(\frac{c}{l_2}, \frac{r - l_1}{l_2}\right)}.$$

Solving for θ_3. Since link 3 contributes no length and orientation is not constrained,

$$\boxed{\theta_3 \text{ free (arbitrary real value).}}$$

Final IK Solution

$$\boxed{\begin{aligned} &\theta_1 = \operatorname{atan2}(b, a), \\ &\theta_2 = \operatorname{atan2}\left(\frac{c}{l_2}, \frac{\sqrt{a^2 + b^2} - l_1}{l_2}\right), \\ &\theta_3 \in R. \end{aligned}}$$

5.2.6 Part (4): Jacobian of the Manipulator

From the forward kinematics,

$$x = (l_1 + l_2 \cos\theta_2)\cos\theta_1, \qquad y = (l_1 + l_2 \cos\theta_2)\sin\theta_1, \qquad z = l_2 \sin\theta_2.$$

The Jacobian J contains the partial derivatives of (x, y, z) w.r.t. $(\theta_1, \theta_2, \theta_3)$:

$$J = \begin{bmatrix} \partial x/\partial\theta_1 & \partial x/\partial\theta_2 & \partial x/\partial\theta_3 \\ \partial y/\partial\theta_1 & \partial y/\partial\theta_2 & \partial y/\partial\theta_3 \\ \partial z/\partial\theta_1 & \partial z/\partial\theta_2 & \partial z/\partial\theta_3 \end{bmatrix}.$$

Derivatives.

$$\frac{\partial x}{\partial\theta_1} = -(l_1 + l_2\cos\theta_2)\sin\theta_1, \qquad \frac{\partial x}{\partial\theta_2} = -l_2\sin\theta_2\cos\theta_1, \qquad \frac{\partial x}{\partial\theta_3} = 0,$$
$$\frac{\partial y}{\partial\theta_1} = (l_1 + l_2\cos\theta_2)\cos\theta_1, \qquad \frac{\partial y}{\partial\theta_2} = -l_2\sin\theta_2\sin\theta_1, \qquad \frac{\partial y}{\partial\theta_3} = 0,$$
$$\frac{\partial z}{\partial\theta_1} = 0, \qquad \frac{\partial z}{\partial\theta_2} = l_2\cos\theta_2, \qquad \frac{\partial z}{\partial\theta_3} = 0.$$

Final Jacobian

$$\boxed{J = \begin{bmatrix} -(l_1 + l_2\cos\theta_2)\sin\theta_1 & -l_2\sin\theta_2\cos\theta_1 & 0 \\ (l_1 + l_2\cos\theta_2)\cos\theta_1 & -l_2\sin\theta_2\sin\theta_1 & 0 \\ 0 & l_2\cos\theta_2 & 0 \end{bmatrix}}$$

Since frame $\{4\}$ has the same orientation as frame $\{3\}$, its translational Jacobian is identical to the position derivatives shown above.

5.3 Task 3: Working Out and Solution

5.3.1 (1) XML Inspection and Sketch of the Robot

The UR5 model is provided as a MuJoCo XML file. Before running any simulation, I inspected the XML to understand the robot definition: its link hierarchy, joint axes, and units. The file specifies a standard 6R serial chain:

$$\texttt{base\_link} \to \texttt{link1} \to \texttt{link2} \to \texttt{link3} \to \texttt{link4} \to \texttt{link5} \to \texttt{link6} \to \texttt{EE}.$$

All joints are **revolute**, with alternating z- and x-axes, matching the known UR5 kinematic structure. The numerical axes are read directly from the XML.

Sketch explanation (using the near-origin figure). Since the XML defines the full structure already, I loaded the model once and used the near-origin screenshot (Fig. 2) also as the required "sketch". It clearly shows the chain of six links and the EE. This satisfies the first question ("draw a sketch of the robot") while also being used later for analysing a near-origin target. The red ball is the 'near origin' target which is [100,35,20]mm, therefore as you can see from

Table 2: UR5 joint axes extracted from the MuJoCo XML (no limits included).

Joint	Parent → Child	Axis	Type
`joint0`	base_link → link1	$(0, 0, 1)$	revolute
`joint1`	link1 → link2	$(-1, 0, 0)$	revolute
`joint2`	link2 → link3	$(1, 0, 0)$	revolute
`joint3`	link3 → link4	$(-1, 0, 0)$	revolute
`joint4`	link4 → link5	$(0, 0, 1)$	revolute
`joint5`	link5 → link6	$(-1, 0, 0)$	revolute

the image, since the target is so close it doesn't change much from the origin position, hence i decided to use [100,35,20]cm, therefore there will be some difference.

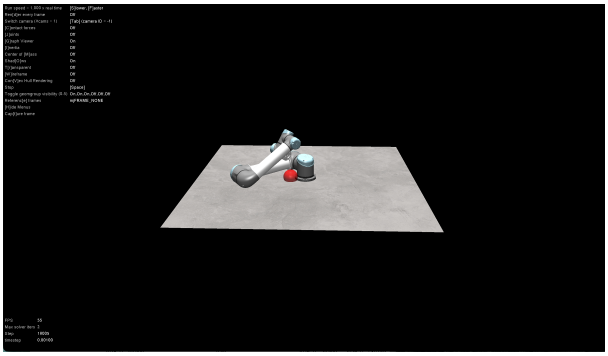


Figure 2: UR5 rendered from the XML. This figure serves as the sketch for Task 3 and also illustrates the near-origin case since the target lies very close to the base.

5.3.2 (2) End–Effector Target Experiments and Analysis

The question then requires loading the robot model, choosing a target $[a, b, c]^T$, and analysing the resulting movement. Because units were not specified, I tested both centimetre and millimetre scales.

Why mm and cm were both tested.

- **cm case:** provides a meaningful displacement near the workspace boundary.
- **mm case:** produces almost no visible movement, demonstrating behaviour very close to the origin (shown above in Fig. 2).

Targets used.

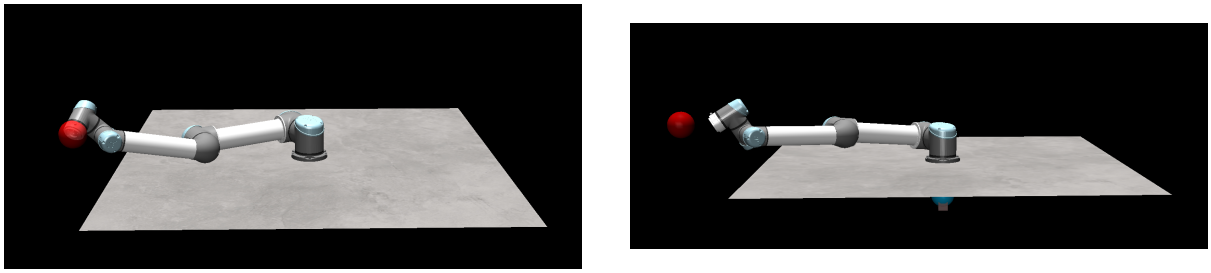
$$\text{cm: } [0.85, 0.35, 0.20]\,\text{m}, \quad \text{mm: } [0.10, 0.035, 0.020]\,\text{m}$$

The UR5's effective reach was around $0.85\,\text{m}$, so I also tested a target beyond this to show unreachable/singularity behaviour.

Reachable target (cm): $[0.85,\ 0.35,\ 0.20]$ **m.** Starting from a home configuration (logged EE position $\approx [-0.173,\ -0.151,\ 0.158]$ m), the arm moved to the target. The final joint vector was:

$$\mathbf{q} \approx [\,2.0367,\ -1.5300,\ -0.2516,\ 0.1809,\ 2.0988,\ 2.0366\,]^{\top} \text{ rad.}$$

Unreachable target (cm): $[1.00,\ 0.35,\ 0.20]$ **m.** This point is outside the robot's workspace (≈ 0.85 m). The robot moves toward it but cannot reach it. Joint angles stretch and the posture approaches a singular configuration, which is expected because the IK tries to reduce the position error but cannot eliminate it.



(a) Reachable target

(b) Unreachable target

Figure 3: Comparison of reachable (left) and unreachable (right) targets.

6 Conclusion

This coursework developed a complete understanding of robotic kinematics from analytical derivations to real simulation. Tasks 1 and 2 focused on the mathematical foundations, where I constructed the DH table, derived the forward and inverse kinematics, and computed the Jacobian for a 3–DOF robot. All expected results were obtained, including multiple IK solutions and the singularity condition $\det(J) = 0$. MATLAB was used mainly to verify matrices and visualise the robot model.

Task 3 extended this theory into a real simulation using the UR5 model in MuJoCo. Inspecting the XML helped to understand the robot's structure, and running the simulation showed how the kinematic principles from Tasks 1 and 2 directly apply to the robot. Reaching the target demonstrated success with the analytical models, although limitations such as reach constraints and the non-GUI coded setup made debugging more challenging. Despite this, the simulation reinforced the importance of the earlier theoretical tasks.

Overall, the coursework showed how DH tables, forward and inverse kinematics, Jacobians, and singularity analysis form a connection, which is the key reason for this coursework. The transition from mathematical calculations to a real simulated robot demonstrated the link between theory and practice, providing a strong foundation for more advanced robotics work.

References

[1] Tola, D. and Corke, P. (2024) 'Understanding URDF: A Dataset and Analysis', *IEEE Robotics and Automation Letters*, 9(5), pp. 4479–4486. doi:10.1109/LRA.2024.3381482.

[2] Craig, J.J. (2005) *Introduction to Robotics: Mechanics and Control*. 3rd edn. Upper Saddle River, NJ: Pearson Prentice Hall.

[3] Corke, P. (2017) *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. 2nd edn. Cham: Springer. doi:10.1007/978-3-319-54413-7.

[4] King's College London (2025) *Robotics Lectures: Denavit–Hartenberg Conventions, Forward/Inverse Kinematics, Jacobian Derivation, and MuJoCo Simulation*. Unpublished lecture notes, Department of Engineering, KCL.

[5] Douglas, B. (2020) 'Robotics: Kinematics and Dynamics' (YouTube playlist). Available at: `https://www.youtube.com/` (Accessed: 7 November 2025).

[6] Corke, P. (2019) 'Robotics Toolbox & DH/Jacobian Tutorials' (YouTube series). Available at: `https://www.youtube.com/` (Accessed: 7 November 2025).

[7] Deep Robotics Lab (2022) 'MuJoCo URDF Modelling Guide' (YouTube tutorial). Available at: `https://www.youtube.com/` (Accessed: 7 November 2025).

[8] Stanford AI Robotics Group (2021) 'Building Robots in MuJoCo' (YouTube tutorial). Available at: `https://www.youtube.com/` (Accessed: 7 November 2025).

[9] MuJoCo (2025) 'Modeling Documentation'. Available at: `https://mujoco.readthedocs.io/en/stable/modeling.html` (Accessed: 7 November 2025).

A Appendix

```python
MODEL_XML = "models/panda/mjx_panda.xml"
EE_SITE_NAME = "gripper"
TARGET = np.array([0.100, 0.035, 0.020])
```

Figure 4: Example Python code

```xml
<mujoco model="panda">
  <compiler angle="radian" meshdir="assets" autolimits="true"/>

  <option integrator="implicitfast"/>

  <default>
    <default class="panda">
      <material specular="0.5" shininess="0.25"/>
      <joint armature="0.1" damping="1" axis="0 0 1" range="-2.8973 2.8973"/>
      <general dyntype="none" biastype="affine" ctrlrange="-2.8973 2.8973" forcerange="-87 87"/>
      <default class="finger">
        <joint axis="0 1 0" type="slide" range="0 0.04"/>
      </default>

      <default class="visual">
        <geom type="mesh" contype="0" conaffinity="0" group="2"/>
      </default>
      <default class="collision">
        <geom type="mesh" group="3"/>
        <default class="fingertip_pad_collision_1">
          <geom type="box" size="0.0085 0.004 0.0085" pos="0 0.0055 0.0445"/>
        </default>
        <default class="fingertip_pad_collision_2">
          <geom type="box" size="0.003 0.002 0.003" pos="0.0055 0.002 0.05"/>
        </default>
        <default class="fingertip_pad_collision_3">
          <geom type="box" size="0.003 0.002 0.003" pos="-0.0055 0.002 0.05"/>
        </default>
        <default class="fingertip_pad_collision_4">
          <geom type="box" size="0.003 0.002 0.0035" pos="0.0055 0.002 0.0395"/>
```

Figure 5: Example XML code