

1. Problem Statement

The objective of this project is to model and analyze the kinematics of the UR5 robotic manipulator using the Standard Denavit–Hartenberg (DH) convention. The project includes:

- Deriving the Forward Kinematics (FK) to compute the end-effector pose from given joint angles.
- Computing the Geometric Jacobian matrix to relate joint velocities to end-effector velocities.
- Performing singularity analysis using determinant and condition number.
- Implementing Numerical Inverse Kinematics (IK) using the Jacobian pseudo-inverse method.
- Verifying the correctness of the IK solution using forward kinematics.

All implementations were developed in Python using NumPy.

2. Robot Description

The UR5 is a 6-DOF industrial robotic manipulator with six revolute joints. It follows a spherical wrist design:

- Joints 1–3 primarily control the position of the wrist center.
- Joints 4–6 primarily control the orientation of the end-effector.

This structural separation simplifies inverse kinematics and is common in industrial robotic arms.

3. Denavit–Hartenberg (DH) Modeling

The Standard DH convention was used to model the UR5 robot.

The Standard DH convention was selected as specified in the project guidelines and ensures systematic frame assignment and transformation consistency.

In Standard DH convention, each transformation consists of:

1. Rotation about Z-axis by θ_i
2. Translation along Z-axis by d_i
3. Translation along X-axis by a_i
4. Rotation about X-axis by α_i

3.1 UR5 Standard DH Parameters

Joint (i)	a_i (m)	α_i (rad)	d_i (m)	θ_i
1	0	$\pi/2$	0.0892	q_1
2	-0.425	0	0	q_2
3	-0.392	0	0	q_3
4	0	$\pi/2$	0.1093	q_4
5	0	$-\pi/2$	0.0947	q_5

6	0	0	0.0823	q6
---	---	---	--------	----

Each joint transformation matrix was constructed using the standard homogeneous transformation formula.

4. Forward Kinematics

Forward Kinematics computes the end-effector pose from joint angles.

The overall transformation is computed as:

$$T_0^6 = T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6$$

Each transformation matrix is 4×4:

$$\begin{aligned} T = \\ [R \ p] \\ [0 \ 1] \end{aligned}$$

Where:

- R is the 3×3 rotation matrix.
- p is the 3×1 position vector.

4.1 FK Verification

For the configuration:

$$q = [0, 0, 0, 0, 0, 0]$$

The computed end-effector position is:

$$[-0.817, -0.1916, -0.0055]$$

The rotation matrix was verified to satisfy:

- $R^T R = I$
- $\det(R) = 1$

This confirms the correctness of the forward kinematics implementation.

5. Jacobian Matrix

The **Geometric Jacobian** relates joint velocities to end-effector velocities:

$$V = J(q) \dot{q}$$

For a revolute joint i:

Linear velocity component:

$$J_{Vi} = z_i \times (p_e - p_i)$$

Angular velocity component:

$$J\omega_i = z_i$$

Where:

- z_i is the joint axis (Z-axis of frame i)
- p_i is the position of joint i
- p_e is the end-effector position

The final Jacobian is a 6×6 matrix.

6. Singularity Analysis

A singularity occurs when the Jacobian loses rank.

Two methods were used:

1. Determinant of Jacobian
 $\det(J) = 0$ indicates singularity.
2. Condition Number
 $\text{cond}(J) = \sigma_{\max} / \sigma_{\min}$

The condition number is more reliable because it indicates how close the system is to singularity.

Example:

A condition number of approximately 7.72 indicates a well-conditioned Jacobian matrix. Since the value is relatively small, the manipulator configuration is numerically stable and far from singularity.

7. Numerical Inverse Kinematics

Analytical IK for 6-DOF manipulators can be complex.

Therefore, Numerical IK using the Jacobian pseudo-inverse method was implemented.

The iterative update rule is:

$$q_{\text{new}} = q_{\text{old}} + J^+ e$$

The Moore–Penrose pseudo-inverse was used instead of the regular matrix inverse to handle singular or near-singular configurations and to ensure numerical stability.

Where:

- J^+ is the Moore–Penrose pseudo-inverse
- e is the error vector (position + orientation)

7.1 Error Computation

Position error:

$$e_p = p_d - p$$

Orientation error:

Computed using rotation matrix difference.

The full error vector:

$$e = [e_p ; e_o]$$

The algorithm iterates until:

$$\|e\| < \text{tolerance}$$

Additional safety measures:

- Maximum iteration limit
- Joint angle normalization within $[-\pi, \pi]$

8. Verification Results

Target Pose:

Position: [-0.6, -0.2, 0.2]

Orientation: Identity matrix

Computed Joint Angles:

[0.148, 0.766, -1.735, -0.602, 1.571, 1.423]

Verification Results:

```
IK Success: True
Joint Angles:
[ 0.14806031  0.76596303 -1.73491908 -0.60183937  1.5707976   1.42273642]

Final Position after IK:
[-0.59999941 -0.19999996  0.20000026]

Final Position Error:
6.40784963476179e-07

===== VERIFICATION RESULTS =====
Final Position: [-0.59999941 -0.19999996  0.20000026]
Position Error: 6.40784963476179e-07
Orientation Error: 1.62368460369052e-06
Jacobian Condition Number: 7.7209220495874815
Verification: SUCCESS
```

Final Position:

[-0.59999941, -0.19999996, 0.20000026]

Position Error:

6.4×10^{-7}

Orientation Error:

1.6×10^{-6}

Jacobian Condition Number:

7.72

The extremely small position and orientation errors confirm the correctness of the implemented inverse kinematics algorithm.

9. Limitations

- Convergence depends on the initial guess.
- Numerical IK may converge to different valid configurations.
- Strict physical joint limits were not enforced.
- Near singular configurations may cause numerical instability.

10. Conclusion

In this project, the UR5 manipulator was successfully modeled using the Standard DH convention. Forward kinematics, Jacobian matrix, singularity analysis, and numerical inverse kinematics were implemented and verified.

The numerical IK achieved high accuracy with negligible position and orientation error, demonstrating the correctness and stability of the implemented kinematic model.