Chapter 10: Forward and Inverse Kinematics

Introduction

Kinematics is the study of motion without considering the forces that cause it. In the field of **robotics**, kinematics forms the foundation for understanding and controlling the position and orientation of a robot's end-effector. It is particularly important in **manipulator-type robots**, which are common in civil engineering applications such as automated construction, bricklaying robots, and bridge inspection arms.

Two essential problems arise in robot kinematics:

- **Forward Kinematics (FK):** Determines the position and orientation of the end-effector given the joint parameters.
- **Inverse Kinematics (IK):** Determines the required joint parameters to achieve a desired position and orientation of the end-effector.

Both are crucial for the precise movement of robots in tasks like 3D printing in construction, robotic surveying, and automated welding in prefabrication structures.

10.1 Basic Concepts of Kinematics

- **Degrees of Freedom (DOF):** Number of independent joint variables needed to specify the configuration of the robot.
- **Kinematic Chains:** Interconnection of rigid bodies (links) and joints to form a manipulator.
- **Types of Joints:**
    - Revolute (Rotational)
    - Prismatic (Translational)
- **Kinematic Parameters:**
    - Joint angles (θ) for revolute joints.
    - Joint displacements (d) for prismatic joints.

10.2 Forward Kinematics

Forward Kinematics (FK) involves calculating the **position and orientation** of the robot's end-effector using the known **joint parameters**.

10.2.1 Denavit-Hartenberg (D-H) Parameters

The D-H convention standardizes the assignment of coordinate frames to robotic links and simplifies the transformation matrices.

- **Four D-H Parameters:**
    a. **θ (theta):** Joint angle
    b. **d:** Link offset
    c. **a:** Link length
    d. **α (alpha):** Link twist

10.2.2 Transformation Matrix

Using D-H parameters, each joint transformation is represented by a 4x4 homogeneous transformation matrix:

$$T_i^{i-1} = \begin{bmatrix} \cosθ_i & -\sinθ_i \cosα_i & \sinθ_i \sinα_i & a_i \cosθ_i \\ \sinθ_i & \cosθ_i \cosα_i & -\cosθ_i \sinα_i & a_i \sinθ_i \\ 0 & \sinα_i & \cosα_i & d_i \\ 0 & 0 & 0 & 1 \\ \end{bmatrix}$$

10.2.3 Chain Multiplication

- The overall transformation from the base to end-effector is obtained by multiplying individual transformation matrices:

$$T_0^n = T_1^0 \cdot T_2^1 \cdot T_3^2 \cdots T_n^{n-1}$$

- This gives the position (x, y, z) and orientation (rotation matrix) of the end-effector.

10.3 Inverse Kinematics

Inverse Kinematics (IK) is the process of finding the joint parameters that achieve a specific **end-effector pose** (position and orientation).

10.3.1 The IK Problem

Given a desired transformation matrix T_0^n, find values of $\theta_1, \theta_2, \ldots, \theta_n$ (or other joint variables) such that:

$$T_0^n = f(\theta_1, \theta_2, \ldots, \theta_n)$$

10.3.2 Types of Solutions

- **Analytical Solution:**
  - Closed-form expressions.
  - Possible only for simple manipulators (e.g., 2 or 3 DOF arms).
- **Numerical Solution:**
  - Iterative techniques like Newton-Raphson or Gradient Descent.
  - Suitable for complex and redundant manipulators.

10.3.3 Multiple Solutions

- IK may have **multiple**, **one**, or **no** solution.
- For example, a 6-DOF manipulator can have up to 16 valid configurations.
- Redundancy (more than 6 DOF) leads to infinite solutions.

10.3.4 Constraints in IK

- **Physical constraints**: Joint limits, workspace boundaries.
- **Collision avoidance**: Especially critical in confined civil environments.
- **Singularities**: Configurations where control becomes unstable or impossible.

10.4 Jacobian Matrix in Kinematics

The **Jacobian matrix (J)** relates the joint velocities to the end-effector linear and angular velocities.

$$\dot{X} = J(q) \cdot \dot{q}$$

Where:

- $\dot{X}$ is the end-effector velocity vector.
- $\dot{q}$ is the joint velocity vector.
- $J(q)$ is the Jacobian matrix as a function of joint configuration.

10.4.1 Jacobian and Singularities

- Jacobian becomes **non-invertible** at singularities.
- Robot loses the ability to move in some directions.
- Important to identify and avoid in design and control.

10.5 Kinematics of Common Manipulators

10.5.1 2-DOF Planar Robot Arm

- Used for basic theoretical understanding.
- Simple FK and IK derivations with trigonometric equations.

10.5.2 3-DOF SCARA Robot

- Common in assembly operations.
- Combines revolute and prismatic joints.
- Demonstrates hybrid kinematic behavior.

10.5.3 6-DOF Industrial Manipulator

- Used in automated bricklaying, 3D printing, and tunneling.
- Complex kinematics with wrist partitioning (solving wrist and arm separately).

10.6 Applications in Civil Engineering Robotics

- **Robotic Arms for Masonry:** Positioning bricks using FK/IK.
- **Bridge Inspection Drones:** Calculating camera position using FK.
- **Tunnel Boring Automation:** Use of robotic arms with precise kinematic control.
- **3D Concrete Printing:** Layer-by-layer construction with precise nozzle control.

10.7 Numerical Methods for Solving Inverse Kinematics

Analytical solutions are not always feasible for complex robotic structures. Hence, **numerical methods** are employed.

10.7.1 Iterative Methods

1. **Newton-Raphson Method:**
    - Linearizes the non-linear kinematic equations.
    - Update rule:

$$q_{i+1} = q_i + J^{-1}(q_i)(X_d - f(q_i))$$

    - Converges quickly near the solution but requires good initial guess.
2. **Gradient Descent Method:**
    - Minimizes the cost function:

$$E(q) = \frac{1}{2} \| f(q) - X_d \|^2$$

    - Slower than Newton-Raphson but more stable in some cases.
3. **Damped Least Squares (Levenberg–Marquardt Algorithm):**
    - Handles singularities by adding damping:

$$\Delta q = ¿$$

    - Offers a balance between speed and stability.

10.7.2 Pseudo-Inverse Jacobian Approach

- When the Jacobian is not square or invertible, use:

$$J^{+¿} = ¿¿$$

- The joint velocities are estimated using:

$$\dot{q} = J^{+¿} \dot{X} ¿$$

- Common in **redundant manipulators**.

10.8 Kinematic Redundancy and Optimization

A manipulator is **kinematically redundant** if it has more DOFs than required to perform a task.

10.8.1 Advantages of Redundancy

- Greater flexibility.
- Collision avoidance.
- Joint limit avoidance.
- Singular configuration avoidance.

10.8.2 Optimization Criteria

- **Minimum joint displacement.**
- **Maximum manipulability index.**
- **Energy-efficient motion planning.**

Optimization is formulated as:

$$\min_{q} \Phi(q) \text{ subject to } f(q) = X_d$$

Where:

- $\Phi(q)$ is the cost function (e.g., energy, joint velocity).
- $f(q)$ is the forward kinematic equation.

10.9 Workspace Analysis

Understanding the **workspace** is critical to designing robots for confined or wide-open civil sites.

10.9.1 Types of Workspace

1. **Reachable Workspace:** All points the end-effector can reach in any orientation.
2. **Dexterous Workspace:** Points where all orientations are possible.
3. **Task-Specific Workspace:** Based on environmental and application constraints.

10.9.2 Workspace Determination

- Analytical for simple robots.
- Simulation-based for complex geometries using Monte Carlo or grid-based sampling.
- Affected by:

- Joint limits
- Link lengths
- Obstacle constraints

10.10 Trajectory Planning in Joint and Cartesian Space

Trajectory planning ensures smooth, safe, and precise motion from one point to another.

10.10.1 Joint Space Trajectory

- Plans the motion as changes in joint parameters.
- Easier to compute but less intuitive.
- Common profiles:
  - **Linear interpolation**
  - **Cubic polynomial**
  - **Trapezoidal velocity**

10.10.2 Cartesian Space Trajectory

- Plans the motion in end-effector coordinates (x, y, z, orientation).
- Requires inverse kinematics at each time step.
- Often used in **welding**, **painting**, and **3D printing** tasks.

10.11 Simulation and Kinematic Modeling Tools

Modern robotics relies heavily on simulation before deployment.

10.11.1 MATLAB Robotics Toolbox

- Supports FK, IK, Jacobian computation.
- Visualizes manipulator motion.

10.11.2 ROS (Robot Operating System)

- Framework for real-time control, motion planning, and sensor integration.
- Supports **MoveIt!** for kinematic planning.

10.11.3 Gazebo and V-REP (CoppeliaSim)

- 3D simulation environments.

- Integrates real-world physics with robotic models.
- Useful in **construction site simulation** and **automated inspection planning**.

10.12 Real-World Integration and Civil Engineering Use Cases

10.12.1 Rebar Tying Robots

- Use FK to reach and orient tie gun at correct position.
- IK ensures arm stays within constrained work zones.

10.12.2 Robotic Total Stations

- Automatically orient to desired location using kinematic control.
- Improve surveying precision and reduce human error.

10.12.3 Tunneling and Mining Robots

- Drilling heads with controlled orientation.
- Use workspace and trajectory planning to avoid collision with tunnel walls.

10.12.4 Climbing Inspection Robots

- Use multi-joint legs or arms.
- IK is crucial to maintain grip on curved or vertical surfaces like bridges and dams.
