Robotics: Kinematics

Minor in AI - IIT ROPAR

31 May, 2025

1 What is Forward Kinematics?

Definition

Forward Kinematics (FK) is the process of determining the *position* and *orientation* (pose) of a robot's **end-effector** (the tool or hand) based on known **joint parameters**. These joint parameters include angles for revolute joints and displacements for prismatic joints.

In other words, FK answers the question:

> *"If we know the configuration of all the robot's joints, where is the end-effector located and how is it oriented in space?"*

Why Forward Kinematics Matters

- **Robot Simulation and Animation:** FK allows visualization of robot motion given joint movements.
- **Motion Planning:** Understanding how changes in joints affect the position of the tool.
- **Foundation for Inverse Kinematics and Dynamics:** FK results are prerequisites for calculating joint commands from desired end-effector poses or forces.

Inputs to Forward Kinematics

- **Joint Variables:** The values specifying each joint's position:
  - *Revolute joints:* Angles (e.g., $\theta_1, \theta_2, \ldots$)
  - *Prismatic joints:* Linear displacements (e.g., $d_1, d_2, \ldots$)
- **Link Parameters:** Geometric constants defining robot structure such as:
  - Lengths of links
  - Twist angles
  - Offsets

Outputs of Forward Kinematics

- The **pose** of the end-effector:
  - *Position* in 3D space (e.g., x, y, z)
  - *Orientation* (e.g., rotation matrix, Euler angles, quaternions)
- Usually expressed with respect to the **world** or **base** frame.

Properties of Forward Kinematics

- **Deterministic:** For every valid set of joint parameters, FK produces a unique end-effector pose.
- **Direct mapping:** From joint space to Cartesian space.

Methods for Forward Kinematics

Several approaches exist to calculate FK:

- **Geometric Method:** Uses direct geometric and trigonometric relations based on robot shape.
  - Intuitive for simple robots.
  - Can get complicated for robots with many degrees of freedom.
- **Denavit–Hartenberg (DH) Convention:**
  - A systematic, matrix-based method.
  - Assigns coordinate frames to each link and expresses transformations as 4×4 matrices.
  - Widely used in industrial robotics for serial manipulators.
  - Easy to implement and generalize to any robot with serial links.
- **Product of Exponentials (PoE) Method:**
  - Advanced approach using screw theory and Lie algebra.
  - More general and mathematically elegant.
  - Useful for complex robotic systems and theoretical analysis.

In this course, the focus is on the Denavit–Hartenberg Method, due to its practicality and widespread adoption in industrial robotics.

2 Overview of the Denavit–Hartenberg (DH) Method

Purpose

The Denavit–Hartenberg (DH) method is a **systematic procedure** designed to compute Forward Kinematics for **serial-link robots**. It transforms the complex problem of determining the pose of a robot's end-effector into a sequence of manageable matrix operations.

What the DH Method Involves

- **Frame Assignment Rules:** Attach coordinate frames to each joint and link according to a standardized convention.
- **DH Parameters:** Extract four geometric parameters per link that describe the relative position and orientation between consecutive frames.
- **DH Transformation Matrix:** Construct a 4×4 homogeneous transformation matrix from these parameters for each joint.
- **Chaining Transformations:** Multiply all these matrices sequentially to compute the pose of the end-effector relative to the base frame.

The DH method simplifies 3D motion analysis into repeated matrix multiplications, making the kinematic equations easier to handle both analytically and computationally.

DH Frame Assignment Rules

To apply the DH method, follow these four essential steps to assign coordinate frames to each robot link:

1. **Assign the z_i axis:**
   - Align z_i with the **axis of motion** of joint i.
   - For a revolute joint, z_i is the axis of rotation.
   - For a prismatic joint, z_i is the direction of translation.

2. **Assign the x_i axis:**
   - Make x_i **perpendicular** to both z_{i-1} and z_i.
   - x_i points along the **common normal** between z_{i-1} and z_i.
3. **Place the origin of frame i:**
   - At the **intersection** of the axes z_i and x_i.
   - If they do not intersect, place it at the joint location.
4. **Define the y_i axis:**
   - Complete a **right-handed coordinate system** by setting

$$\vec{y_i} = \vec{z_i} \times \vec{x_i}$$

DH Parameters and Their Geometric Meaning

For each link and joint pair (i.e., the transformation from frame $i-1$ to frame i), we define four Denavit–Hartenberg parameters:

- θ_i: **Joint angle** — rotation about the z_{i-1} axis.
  - Variable for **revolute joints**.
- d_i: **Link offset** — translation along the z_{i-1} axis.
  - Variable for **prismatic joints**.
- a_i: **Link length** — the distance between z_{i-1} and z_i measured along the x_i axis (common normal length).
- α_i: **Link twist** — the angle between the z_{i-1} and z_i axes measured about the x_i axis.

Summary of DH Parameters and Joint Types

- For a **revolute joint**: θ_i is *variable*, while d_i is *constant*.
- For a **prismatic joint**: d_i is *variable*, while θ_i is *constant*.

These four parameters $(\theta_i, d_i, a_i, \alpha_i)$ fully specify the **relative transformation** from frame $i-1$ to frame i.

Visualizing DH Parameters

- d_i: Offset along the previous z-axis to reach the common normal.
- θ_i: Angle about the previous z-axis between the old x and new x axes.
- a_i: Length of the common normal (distance between z_{i-1} and z_i axes).
- α_i: Angle about the common normal (the x-axis), from the old z axis to the new z axis.

3 How DH Parameters Become Transformation Matrices

To compute the transformation from frame $i-1$ to frame i, the Denavit–Hartenberg (DH) method breaks down the process into **four elementary transformations**. Each corresponds to either a rotation or a translation along one of the coordinate axes defined by the frames. By applying these transformations in sequence, we capture both the rotation and translation required to move from one joint frame to the next.

Step 1: Rotation about z_{i-1} by θ_i

This rotation accounts for the joint angle θ_i, which is variable for revolute joints. It represents the rotation of frame i around the previous frame's z-axis.

$$R_z(\theta_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This matrix rotates any vector by θ_i around the z-axis. - The rotation affects the x and y coordinates while leaving z unchanged. - It corresponds to the "turning" motion of a revolute joint.

Step 2: Translation along z_{i-1} by d_i

This translation moves the frame along the z-axis by a distance d_i. For prismatic joints, d_i is variable; for revolute joints, it is constant.

$$T_z(d_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This matrix shifts the position along the z-axis without changing orientation. - It models the linear displacement for prismatic joints or the fixed offset along the joint axis.

Step 3: Translation along x_i by a_i

This translation moves the frame along the x_i axis by the link length a_i. This length is the distance between the z_{i-1} and z_i axes measured along the common normal.

$$T_x(a_i) = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This moves the origin along the x-axis to the next joint's axis location. - It captures the linear offset (link length) between consecutive joint axes.

Step 4: Rotation about x_i by α_i

This rotation accounts for the link twist α_i, which is the angle between the z_{i-1} and z_i axes measured about x_i.

$$R_x(\alpha_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This rotates the coordinate frame around the x-axis, aligning z_i with the next joint axis. - It accounts for the twist or angular displacement between links.

Combining the Four Transformations

The total transformation matrix from frame $i-1$ to frame i is obtained by matrix multiplication of these four elementary transformations in the specific order:

$$T_i^{i-1} = R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i)$$

- The order is critical since matrix multiplication is not commutative. - This combined matrix encapsulates the rotation and translation to move from frame $i-1$ to i. - It is a homogeneous transformation matrix, representing both rotation (upper-left 3×3 block) and translation (upper-right 3×1 vector).

Interpretation of the Sequence:

- First, rotate about the previous joint axis z_{i-1} by the joint angle θ_i.
- Next, translate along that same axis by the joint offset d_i.
- Then translate along the new x_i axis by the link length a_i.
- Finally, rotate about the x_i axis by the link twist α_i.

Each transformation builds on the last, progressively repositioning and reorienting the coordinate frame from one joint to the next.

This systematic approach simplifies the complex 3D kinematics problem into repeatable matrix multiplications, enabling efficient and accurate computation of the robot's configuration.

4 Final DH Transformation Matrix

By multiplying the four elementary transformations

$$T_i^{i-1} = R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i),$$

we obtain the final Denavit–Hartenberg transformation matrix between frames $i-1$ and i:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Interpretation of Each Row:

- **Rows 1–3:** These rows contain the rotation and translation components.
  - The first three columns form the 3×3 rotation matrix R, describing orientation change from frame $i-1$ to frame i.
  - The fourth column is the translation vector $\mathbf{d} = [a_i \cos\theta_i,\ a_i \sin\theta_i,\ d_i]^T$, describing the position offset between the frames.
- **Row 4:** Always [0 0 0 1], this row preserves homogeneous coordinates for matrix multiplication.

Using the Transformation Matrices:

To find the pose of the end-effector relative to the base frame (frame 0), multiply all individual transformations from the base up to the last joint n:

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

- Each T_i^{i-1} is the DH transformation matrix for joint i. - The product T_n^0 gives the full forward kinematics: the position and orientation of the end-effector in the base frame. - This matrix multiplication combines all joint rotations and link translations in sequence.

Summary:

- The DH transformation matrix succinctly captures both rotation and translation between two consecutive robot links. - It allows a modular, repeatable approach for serial manipulators of any length. - By chaining these matrices, one can compute the overall robot configuration efficiently.

5 Final DH Transformation Matrix

By multiplying the four elementary transformations

$$T_i^{i-1} = R_z(\theta_i) \cdot T_z(d_i) \cdot T_x(a_i) \cdot R_x(\alpha_i),$$

we obtain the final Denavit–Hartenberg transformation matrix between frames $i-1$ and i:

$$T_i^{i-1} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Interpretation of Each Row:

- **Rows 1–3:** These rows contain the rotation and translation components.
  - The first three columns form the 3×3 rotation matrix R, describing orientation change from frame $i-1$ to frame i.
  - The fourth column is the translation vector $\mathbf{d} = [a_i \cos\theta_i,\ a_i \sin\theta_i,\ d_i]^T$, describing the position offset between the frames.
- **Row 4:** Always [0 0 0 1], this row preserves homogeneous coordinates for matrix multiplication.

Using the Transformation Matrices:
To find the pose of the end-effector relative to the base frame (frame 0), multiply all individual transformations from the base up to the last joint n:

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

- Each T_i^{i-1} is the DH transformation matrix for joint i. - The product T_n^0 gives the full forward kinematics: the position and orientation of the end-effector in the base frame. - This matrix multiplication combines all joint rotations and link translations in sequence.

Summary:
- The DH transformation matrix succinctly captures both rotation and translation between two consecutive robot links. - It allows a modular, repeatable approach for serial manipulators of any length. - By chaining these matrices, one can compute the overall robot configuration efficiently.

6 What is Inverse Kinematics?

Definition: Inverse Kinematics (IK) is the process of computing the joint parameters (angles or displacements) needed for the robot's end-effector to reach a specified pose — that is, a desired position and orientation in space.
Why It Matters:

- Enables robots to plan and execute movements to achieve specific goals.
- Essential for motion planning, control, and teleoperation tasks.

Inputs: Desired end-effector pose — both position and orientation.
Outputs: Joint variables:

- Joint angles for revolute joints.
- Linear displacements for prismatic joints.

Forward Kinematics vs. Inverse Kinematics:

- **Forward Kinematics (FK):** Maps joint variables to end-effector pose (straightforward, deterministic).
- **Inverse Kinematics (IK):** Maps end-effector pose back to joint variables (more complex and often challenging).

Challenges in Inverse Kinematics – The Solution Space

IK is inherently more complex due to the nature of its solution space:

1. **Multiple Solutions:** A single end-effector pose may correspond to multiple valid joint configurations. *Example:* A 2-link planar arm can reach the same point with an elbow-up or elbow-down posture.
2. **No Solution:** The target pose might lie outside the robot's reachable workspace, making the IK problem unsolvable for that pose.
3. **Infinite Solutions (Redundancy):** In robots with more degrees of freedom (DOF) than task dimensions (e.g., a 7-DOF arm performing a 6-DOF task), infinite solutions exist for the same end-effector pose.
4. **Nonlinear Equations:** IK involves solving nonlinear trigonometric equations that may have multiple roots or require iterative numerical methods. These nonlinearities make finding solutions challenging and computationally intensive.

This complexity motivates the development of various IK solution methods, such as analytical approaches, numerical solvers, and optimization-based techniques.

7 Inverse Kinematics – Methods Overview

Inverse Kinematics (IK) can be approached using various methods depending on the robot's complexity, degrees of freedom, and required precision:

1. **Geometric (Trigonometric) Method:**
   - Solves IK by applying classical geometry using laws of sines and cosines.
   - Best suited for simple planar robots, such as a 2-link robotic arm.
   - Advantages: Intuitive, visual, and computationally efficient.
2. **Algebraic Method:**
   - Involves symbolically inverting the Forward Kinematics equations.
   - Requires solving systems of nonlinear equations involving trigonometric functions.
   - Well-suited for robots with low degrees of freedom.
   - Scalable and often supported by symbolic computation software.
3. **Numerical (Iterative) Methods:**
   - Use numerical techniques such as the Jacobian inverse, Jacobian pseudoinverse, or optimization algorithms.
   - Can handle complex robot geometries, redundant DOFs, and constraints.
   - Typically require an initial guess and iterative refinement.
   - May not always converge or guarantee a global solution.
4. **Learning-Based Methods:**
   - Employ neural networks, regression models, or other machine learning techniques to approximate IK mappings.
   - Offer fast inference after training, useful for real-time control or robots with many DOFs.
   - May suffer from limitations in accuracy and generalization to unseen poses.