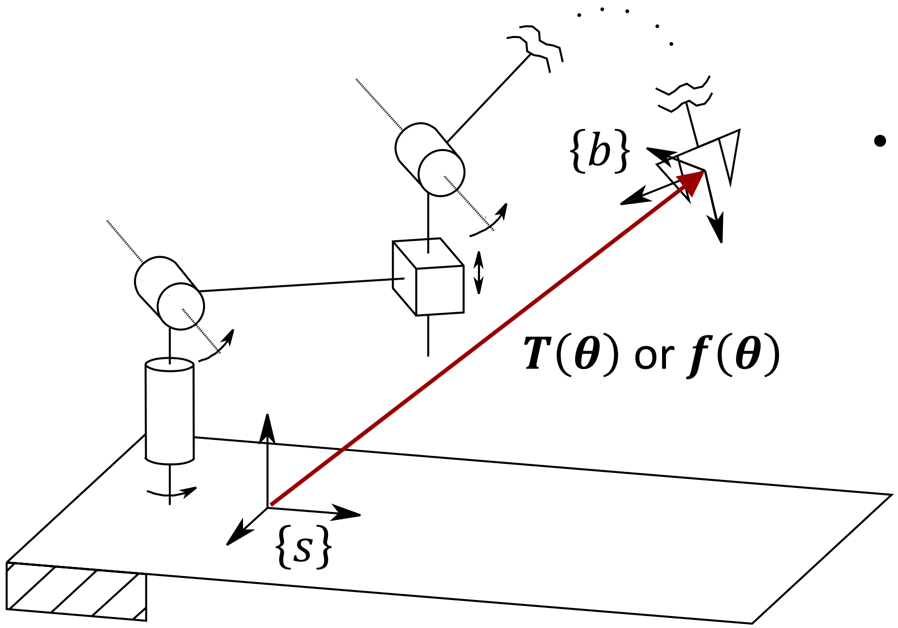
# Ch7: Inverse Kinematics

# Inverse Kinematics

# Inverse Kinematics

The inverse kinematics of a robot refers to the calculation of the joint coordinates $\boldsymbol{\theta}$ from the position and orientation (**pose**) of its end-effector frame.




- "Geometric" inverse kinematics:

$$\text{Given } \boldsymbol{T}_{sb} = \boldsymbol{T}(\boldsymbol{\theta}) \in SE(3), \text{ Find } \boldsymbol{\theta} \in \mathbb{R}^n$$

$$\boldsymbol{T}: \mathbb{R}^n \rightarrow SE(3)$$

- "Minimum-Coordinate" inverse kinematics:

$$\text{Given } \boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta}) \in \mathbb{R}^r, \text{ Find } \boldsymbol{\theta} \in \mathbb{R}^n$$

$$\boldsymbol{f}: \mathbb{R}^n \rightarrow \mathbb{R}^r$$

# Complexities of Inverse Kinematics

- The equations to solve are in general nonlinear. Thus, it is not always possible to find a closed-form solution.
- Multiple (finite) solutions may exist.
- Infinite solutions may exist (e.g., in the case of a kinematically redundant manipulator).
- There might be no admissible solutions (e.g., when the given EE pose does not belong to the manipulator dexterous workspace.).

▶ Solving Inverse Kinematics Problems:

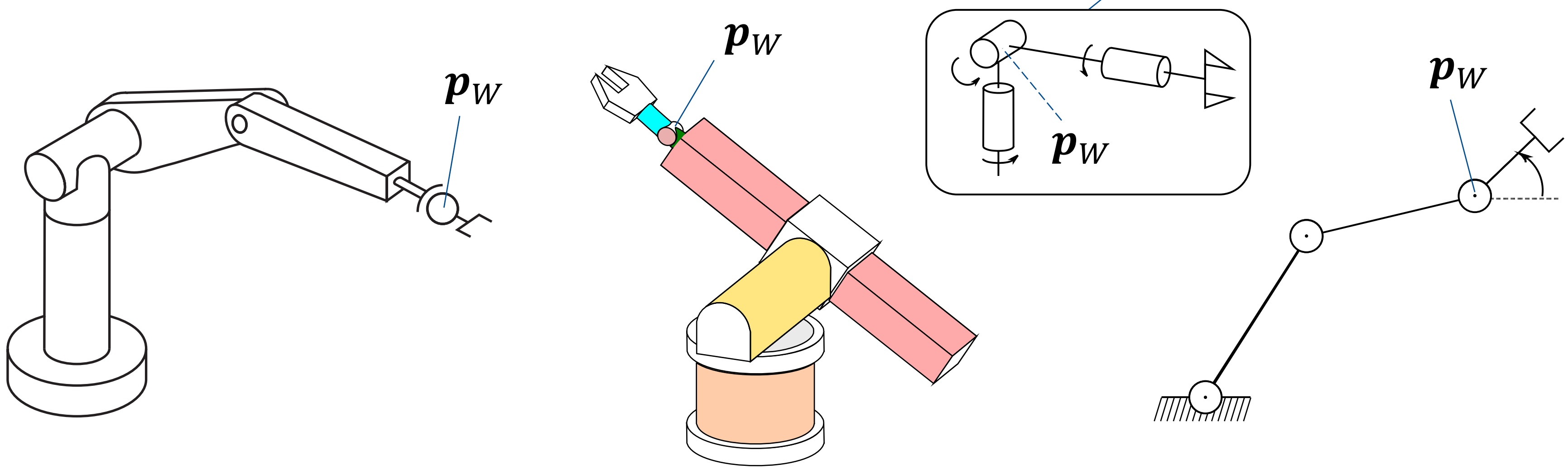
- **Analytic Methods**: Finding closed-form solutions using algebraic intuition or geometric intuition.

- **Iterative Numerical Methods**: When there are no (or it is difficult to find) closed-form solutions.

# Analytic Methods

# Analytic Inverse Kinematics

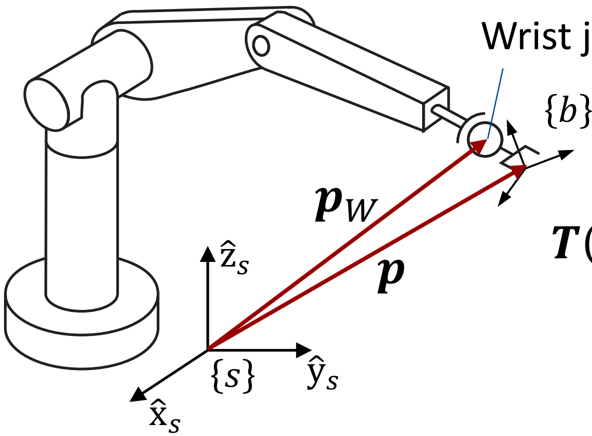
Most of the existing manipulators are typically formed by an **arm** and a **spherical wrist** (where three consecutive revolute joint axes intersect at a common point $\boldsymbol{p}_W$). Thus, we can decouple the solution for the position (i.e., point $\boldsymbol{p}_W$ at the intersection of the three revolute axes) from that for the orientation.




PUMA Arm (6R) Stanford Arm (RRPRRR) 3R Planar Arm

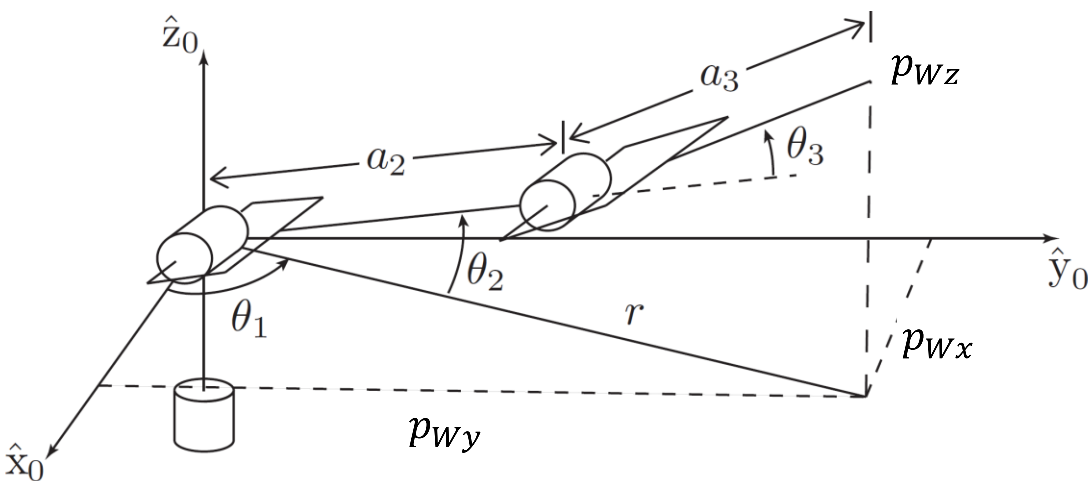
$*$ Therefore, it is possible to solve the inverse kinematics for the arm separately from the inverse kinematics for the spherical wrist.

# Example 1: 6R PUMA-Type Arms



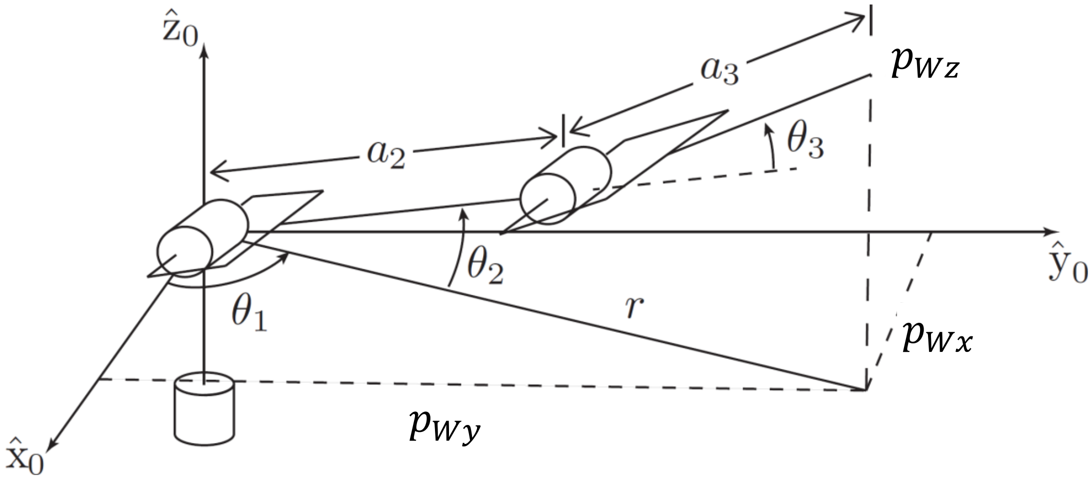

Wrist joints intersect orthogonally at a common point $\boldsymbol{p}_W$

$$\boldsymbol{T}(\boldsymbol{\theta}) = \begin{bmatrix} \boldsymbol{R}_{sb} & \boldsymbol{p} \\ \boldsymbol{0} & 1 \end{bmatrix} = e^{[\boldsymbol{S}_1]\theta_1} e^{[\boldsymbol{S}_2]\theta_2} e^{[\boldsymbol{S}_3]\theta_3} e^{[\boldsymbol{S}_4]\theta_4} e^{[\boldsymbol{S}_5]\theta_5} e^{[\boldsymbol{S}_6]\theta_6} \boldsymbol{M}$$




By having $\boldsymbol{p}$, we can find $\boldsymbol{p}_W = (p_{Wx}, p_{Wy}, p_{Wz})$, and then, we can find $(\theta_1, \theta_2, \theta_3)$ as follows.

# Example 1: 6R PUMA-Type Arms (cont.)




$$p_{Wx} = c_1(a_2c_2 + a_3c_{23}) = c_1 r$$
$$p_{Wy} = s_1(a_2c_2 + a_3c_{23}) = s_1 r$$
$$p_{Wz} = a_2s_2 + a_3s_{23}$$

❖ Inverse position problem of finding $(\theta_1, \theta_2, \theta_3)$ using algebraic intuition:

$$p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2 = a_2^2 + a_3^2 + 2a_2a_3c_3$$

$$c_3 = \frac{p_{Wx}^2 + p_{Wy}^2 + p_{Wz}^2 - a_2^2 - a_3^2}{2a_2a_3}$$
$$s_3 = \pm\sqrt{1 - c_3^2}$$

⇨ $\theta_3 = \text{atan2}(s_3, c_3)$ ⇨ $\theta_{3,\text{I}} \in [-\pi, \pi]$, $\theta_{3,\text{II}} = -\theta_{3,I}$

# Example 1: 6R PUMA-Type Arms (cont.)

$$p_{Wx}^2 + p_{Wy}^2 = (a_2 c_2 + a_3 c_{23})^2 \longrightarrow a_2 c_2 + a_3 c_{23} = \pm\sqrt{p_{Wx}^2 + p_{Wy}^2} = \pm r$$

$$p_{Wz} = a_2 s_2 + a_3 s_{23}$$

$$s_{23} = s_2 c_3 + s_3 c_2$$

$$c_{23} = c_2 c_3 - s_2 s_3$$

$\Rightarrow$

$$c_2 = \frac{\pm\sqrt{p_{Wx}^2 + p_{Wy}^2}(a_2 + a_3 c_3) + p_{Wz} a_3 s_3}{a_2^2 + a_3^2 + 2 a_2 a_3 c_3}$$

$$s_2 = \frac{p_{Wz}(a_2 + a_3 c_3) - \left(\pm\sqrt{p_{Wx}^2 + p_{Wy}^2}\, a_3 s_3\right)}{a_2^2 + a_3^2 + 2 a_2 a_3 c_3}$$

$\Rightarrow$ $\theta_2 = \text{atan2}(s_2, c_2)$

For each $\theta_3$, we have two solutions for $\theta_2$:

$$\begin{cases} \theta_{3,\text{I}} & \begin{cases} \theta_{2,\text{I}} & (+r) \\ \theta_{2,\text{II}} & (-r) \end{cases} \\ \theta_{3,\text{II}} & \begin{cases} \theta_{2,\text{III}} & (+r) \\ \theta_{2,\text{IV}} & (-r) \end{cases} \end{cases}$$

$\theta_{3,\text{I}} \rightarrow (\theta_{2,\text{I}}, \theta_{2,\text{II}})$

$\theta_{3,\text{II}} \rightarrow (\theta_{2,\text{III}}, \theta_{2,\text{IV}})$

# Example 1: 6R PUMA-Type Arms (cont.)

$$p_{Wx} = c_1(a_2c_2 + a_3c_{23})$$
$$p_{Wy} = s_1(a_2c_2 + a_3c_{23})$$
$$a_2c_2 + a_3c_{23} = \pm\sqrt{p_{Wx}^2 + p_{Wy}^2} = \pm r$$

$\Rightarrow$

$$p_{Wx} = \pm c_1\sqrt{p_{Wx}^2 + p_{Wy}^2}$$
$$p_{Wy} = \pm s_1\sqrt{p_{Wx}^2 + p_{Wy}^2}$$

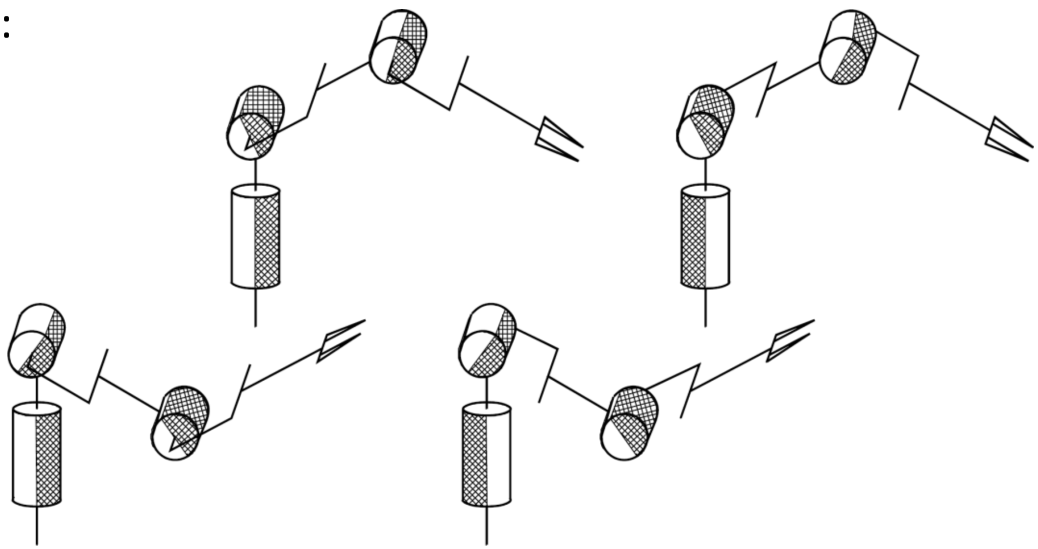
$\Rightarrow$

$$\theta_{1,\mathrm{I}} = \operatorname{atan2}(p_{Wy}, p_{Wx})$$
$$\theta_{1,\mathrm{II}} = \operatorname{atan2}(-p_{Wy}, -p_{Wx})$$

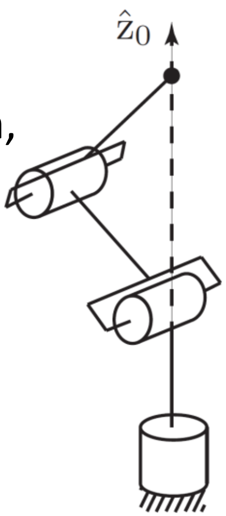
Thus, in total, there exist four solutions:

- $\theta_{3,\mathrm{I}}$
  - $\theta_{2,\mathrm{I}}$ $(+r)$ → $\theta_{1,\mathrm{I}}$
  - $\theta_{2,\mathrm{II}}$ $(-r)$ → $\theta_{1,\mathrm{II}}$
- $\theta_{3,\mathrm{II}}$
  - $\theta_{2,\mathrm{III}}$ $(+r)$ → $\theta_{1,\mathrm{I}}$
  - $\theta_{2,\mathrm{IV}}$ $(-r)$ → $\theta_{1,\mathrm{II}}$

$$(\theta_{1,\mathrm{I}}, \theta_{2,\mathrm{I}}, \theta_{3,\mathrm{I}})$$
$$(\theta_{1,\mathrm{I}}, \theta_{2,\mathrm{III}}, \theta_{3,\mathrm{II}})$$
$$(\theta_{1,\mathrm{II}}, \theta_{2,\mathrm{II}}, \theta_{3,\mathrm{I}})$$
$$(\theta_{1,\mathrm{II}}, \theta_{2,\mathrm{IV}}, \theta_{3,\mathrm{II}})$$



# Example 1: 6R PUMA-Type Arms (cont.)




**Note**: When $p_{Wx} = p_{Wy} = 0$, the arm is in a kinematically singular configuration, and there are infinitely many possible solutions for $\theta_1$.

- Inverse orientation problem of finding $(\theta_4, \theta_5, \theta_6)$ after finding $(\theta_1, \theta_2, \theta_3)$:

$$e^{[\boldsymbol{S}_4]\theta_4} e^{[\boldsymbol{S}_5]\theta_5} e^{[\boldsymbol{S}_6]\theta_6} = e^{-[\boldsymbol{S}_3]\theta_3} e^{-[\boldsymbol{S}_2]\theta_2} e^{-[\boldsymbol{S}_1]\theta_1} \boldsymbol{T}(\boldsymbol{\theta}) \boldsymbol{M}^{-1} = \boldsymbol{T}' = (\boldsymbol{R}', \boldsymbol{p}')$$

known

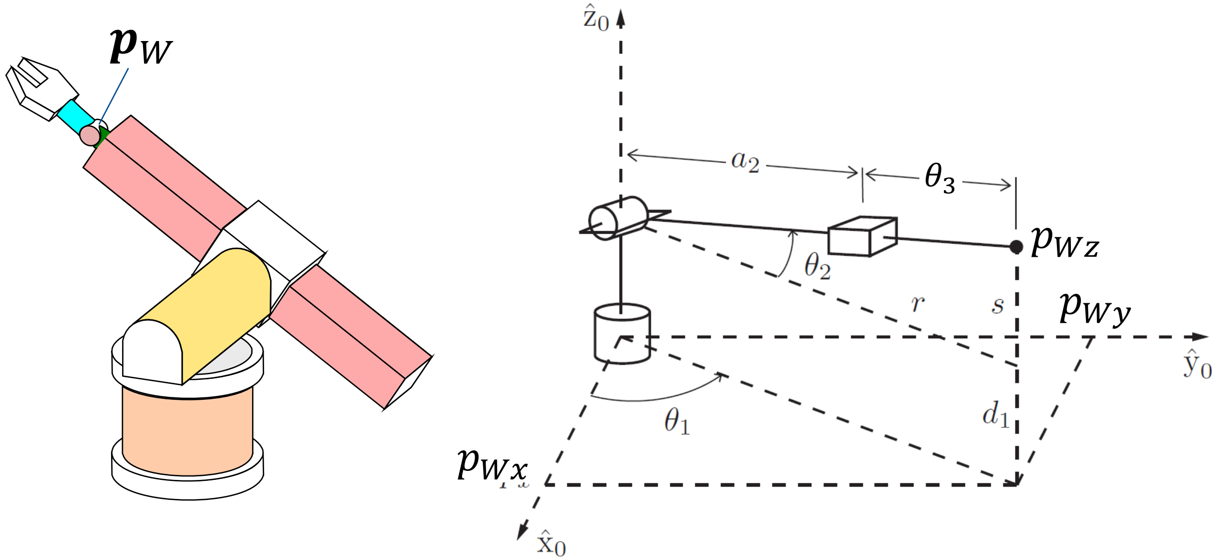
Assume that the joint axes $(\boldsymbol{S}_4, \boldsymbol{S}_5, \boldsymbol{S}_6)$ of the spherical wrist are aligned in the $(\hat{z}_s, \hat{y}_s, \hat{x}_s)$ directions, respectively:

$$\boldsymbol{S}_{\omega_4} = (0,0,1), \quad \boldsymbol{S}_{\omega_5} = (0,1,0), \quad \boldsymbol{S}_{\omega_6} = (1,0,0)$$

⇨ $\mathrm{Rot}(\hat{\mathrm{z}}, \theta_4)\mathrm{Rot}(\hat{\mathrm{y}}, \theta_5)\mathrm{Rot}(\hat{\mathrm{x}}, \theta_6) = \boldsymbol{R}'$ ⇨ This corresponds to the ZYX Euler angles.

⇨ $(\theta_4, \theta_5, \theta_6)$

# Example 2: Stanford-Type Arms




$$r^2 = p_{Wx}^2 + p_{Wy}^2$$
$$s = p_{Wz} - d_1$$

❖ Inverse position problem of finding $(\theta_1, \theta_2, \theta_3)$ using geometric intuition:

If $p_{Wx}, p_{Wy} \neq 0$: $\begin{cases} \theta_1 = \text{atan2}(p_{Wy}, p_{Wx}) \\ \theta_2 = \text{atan2}(s, r) \end{cases}$, $\begin{cases} \theta_1 = \pi + \text{atan2}(p_{Wy}, p_{Wx}) \\ \theta_2 = \pi - \text{atan2}(s, r) \end{cases}$

$(\theta_3 + a_2)^2 = r^2 + s^2$ ⟶ $\theta_3 = \sqrt{r^2 + s^2} - a_2 = \sqrt{p_{Wx}^2 + p_{Wy}^2 + (p_{Wz} - d_1)^2} - a_2$

$\Rightarrow$ Thus, there are 2 solutions to the inverse kinematics problem.

❖ Inverse orientation problem of finding $(\theta_4, \theta_5, \theta_6)$ is similar to PUMA.

# Iterative Numerical Methods

# Numerical Method:
# The Simplest IK Method Using IVK

Velocity kinematics equation $\boldsymbol{\mathcal{V}} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$ can be used to tackle the inverse kinematics problem. Suppose that the end-effector motion $\boldsymbol{\mathcal{V}}_d(t)$ and the initial robot configuration $\boldsymbol{\theta}(0)$ are given. The aim is to determine a feasible joint position and velocity $\left(\boldsymbol{\theta}(t), \dot{\boldsymbol{\theta}}(t)\right)$ that reproduces the given end-effector motion $\boldsymbol{\mathcal{V}}_d(t)$.

From Inverse Velocity Kinematics (IVK): $\dot{\boldsymbol{\theta}} = \boldsymbol{J}^{+}(\boldsymbol{\theta})\boldsymbol{\mathcal{V}}_d$ then, $\boldsymbol{\theta}(t) = \int_0^t \dot{\boldsymbol{\theta}}(\varsigma)d\varsigma + \boldsymbol{\theta}(0)$.

Using Euler integration method and an integration interval $\Delta t = t_{k+1} - t_k$:

$$\boldsymbol{\theta}(t_{k+1}) = \boldsymbol{\theta}(t_k) + \dot{\boldsymbol{\theta}}(t_k)\Delta t = \boldsymbol{\theta}(t_k) + \boldsymbol{J}^{+}\left(\boldsymbol{\theta}(t_k)\right)\boldsymbol{\mathcal{V}}_d(t_k)\Delta t$$

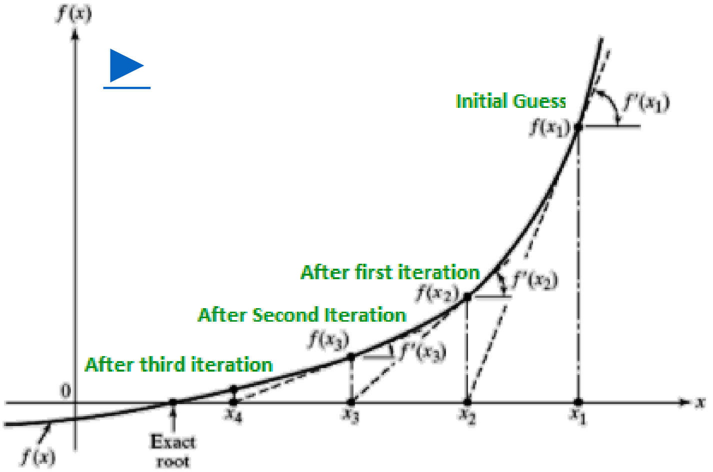
However, due to **drift phenomena** in numerical integration, small velocity errors are likely to <u>accumulate over time</u>, resulting in increasing position error $\boldsymbol{\theta}$ and the end-effector pose corresponding to the computed joint variables differs from the desired one.

Thus, an end-effector pose feedback in algorithm is required to keep the end-effector following the desired pose/motion.

# Jacobian (Pseudo-)Inverse Method

# Preliminary: Newton–Raphson Method

**Newton–Raphson Method** is an iterative method for numerically finding the roots of a nonlinear equation $f(x) = 0$ where $f: \mathbb{R} \to \mathbb{R}$ is differentiable.




If $x^0$ is an initial guess for the solution, Taylor expansion of $f(x)$ at $x^0$ is

$$f(x) = f(x^0) + \frac{\mathrm{d}f}{\mathrm{d}x}(x^0)(x - x^0) + \text{higher-order terms (h.o.t)} \quad (\cong 0) \xrightarrow{f(x)=0} x = x^0 - \left(\frac{\mathrm{d}f}{\mathrm{d}x}(x^0)\right)^{-1} f(x^0)$$

Using $x$ as the new guess for the solution and repeating:

$$x^{k+1} = x^k - \left(\frac{\mathrm{d}f}{\mathrm{d}x}(x^k)\right)^{-1} f(x^k)$$

The iteration is repeated until some stopping criterion is satisfied, e.g., $\dfrac{|f(x^{k+1}) - f(x^k)|}{|f(x^k)|} \le \epsilon$

$\epsilon$: a given threshold value

# Jacobian (Pseudo-)Inverse Method
## (Minimum-Coordinate IK – Configuration Level)

Assume that the EE pose is represented by the minimum number of coordinates, i.e., $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta}) \in \mathbb{R}^r$, $\boldsymbol{\theta} \in \mathbb{R}^n$ ($\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^r$). Thus, given a desired EE pose $\boldsymbol{x}_d$, the goal is to find joint coordinates $\boldsymbol{\theta} = \boldsymbol{\theta}_d$ such that

$$\boldsymbol{x}_d = \boldsymbol{f}(\boldsymbol{\theta}_d)$$

(Assumption: $\boldsymbol{f}$ is differentiable)

- We use a method similar to the Newton–Raphson method for nonlinear root-finding.

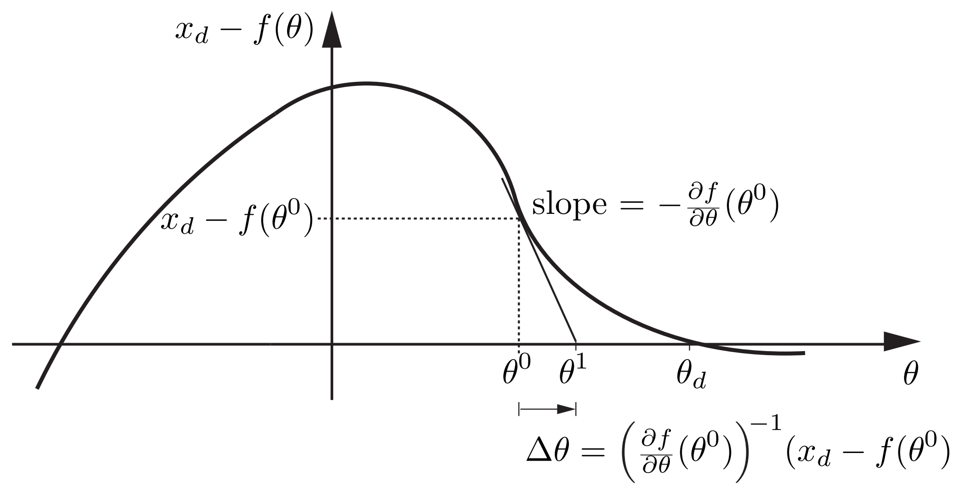
Given an initial guess $\boldsymbol{\theta}^0$ which is "close to" a solution $\boldsymbol{\theta}_d$, and using the Taylor expansion:

$$\boldsymbol{x}_d = \boldsymbol{f}(\boldsymbol{\theta}) = \boldsymbol{f}(\boldsymbol{\theta}^0) + \underbrace{\left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}^0}}_{\boldsymbol{J}_a(\boldsymbol{\theta}^0) \in \mathbb{R}^{r \times n}} \underbrace{(\boldsymbol{\theta} - \boldsymbol{\theta}^0)}_{\Delta\boldsymbol{\theta}} + \text{h.o.t.}$$

Analytical Jacobian at $\boldsymbol{\theta}^0$

Approximately: (h.o.t. = **0**) →

$$\boldsymbol{J}_a(\boldsymbol{\theta}^0)\Delta\boldsymbol{\theta} = \boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}^0)$$



# Jacobian (Pseudo-)Inverse Method
## (Minimum-Coordinate IK – Configuration Level)

* If $\boldsymbol{J}_a$ is square ($r = n$) and invertible: $\Delta\boldsymbol{\theta} = \boldsymbol{J}_a{}^{-1}(\boldsymbol{\theta}^0)\left(\boldsymbol{x}_d - \boldsymbol{f}(\theta^0)\right)$

$$\Rightarrow \quad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \lambda \boldsymbol{J}_a{}^{-1}(\boldsymbol{\theta}^k)\left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}^k)\right), \qquad k = 0,1,2,\dots$$

where $0 < \lambda \leq 1$ is the step length. $\boldsymbol{\theta}^0, \boldsymbol{\theta}^1, \boldsymbol{\theta}^2, \dots \rightarrow \boldsymbol{\theta}_d$

* If $\boldsymbol{J}_a$ is not square or not invertible (due to singularity): $\Delta\boldsymbol{\theta} = \boldsymbol{J}_a{}^{+}(\boldsymbol{\theta}^0)\left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}^0)\right)$

$\boldsymbol{J}_a{}^{+}$: Moore–Penrose pseudoinverse

$$\Rightarrow \quad \boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \lambda \boldsymbol{J}_a{}^{+}(\boldsymbol{\theta}^k)\left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}^k)\right), \qquad k = 0,1,2,\dots$$

**Note**: If robot is redundant ($n > r$) and $\boldsymbol{J}_a$ is full rank ($\mathrm{rank}(\boldsymbol{J}_a) = \min(r, n)$), i.e., the robot is not at a singularity:

$$\boldsymbol{J}_a{}^{+} = \boldsymbol{J}_a{}^{T}\left(\boldsymbol{J}_a \boldsymbol{J}_a{}^{T}\right)^{-1}$$

# Remarks

- The step length $\lambda$ can be adjusted to aid convergence. It may be chosen as a scalar $\lambda \in \mathbb{R}$ or as a diagonal matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ (to scale each component of the configuration $\boldsymbol{\theta}$ separately).

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^{k} + \boldsymbol{\Lambda}\boldsymbol{J}_a{}^{+}(\boldsymbol{\theta}^{k})\left(\boldsymbol{x}^{d} - \boldsymbol{f}(\boldsymbol{\theta}^{k})\right), \qquad k = 0,1,2,\ldots$$

The step length $\lambda$ or $\boldsymbol{\Lambda}$ can be either a constant or as a function of $k$.

- If there are multiple inverse kinematics solutions, the iterative process tends to converge to the solution that is "closest" to the initial guess $\boldsymbol{\theta}^0$.

- **Methods of optimization** are needed in situations where an exact solution may not exist and we seek the closest approximate solution; or, conversely, an infinity of inverse kinematics solutions exists (i.e., if the robot is kinematically redundant) and we seek a solution that is optimal with respect to some criterion/constraints.

# Algorithm for Minimum-Coordinate Representation

**a) Initialization**: Given $\boldsymbol{x}_d \in \mathbb{R}^r$ and an initial guess $\boldsymbol{\theta}^0 \in \mathbb{R}^n$, set $k = 0$.

**b) Iteration**: Set $\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}^k)$. While $\|\boldsymbol{e}\| > \epsilon$ for some small $\epsilon \in \mathbb{R}$:

- Set $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \lambda\, \boldsymbol{J}^+(\boldsymbol{\theta}^i)\boldsymbol{e}$.
- Increment $k$.

$0 < \lambda \leq 1$: step length parameter

Algorithm in MATLAB:

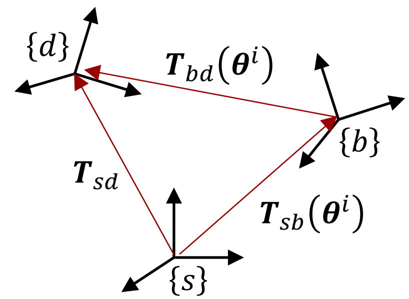
```
max_iterations = 20;
k = 0;
lambda = 1;
Theta = Theta_0;
e = X_d - FK(Theta);
while norm(e) > epsilon && k < max_iterations
    Theta = Theta + lambda * pinv(J(Theta)) * e;
    k = k + 1;
    e = X_d - FK(Theta);
end
```

**Note**: For the motion of a robot along a given desired trajectory, a good choice for the initial guess $\boldsymbol{\theta}^0$ is to use the solution to the IK at the previous time step.

# Algorithm for Transformation Matrix Representation

Assume that the EE pose is represented by a Transformation Matrix, i.e., $\boldsymbol{T}_{sb} = \boldsymbol{T}(\boldsymbol{\theta}) \in SE(3)$, $\boldsymbol{\theta} \in \mathbb{R}^n$. Thus, given a desired EE pose $\boldsymbol{T}_{sd}$, the goal is to find joint coordinates $\boldsymbol{\theta} = \boldsymbol{\theta}_d$ such that

$$\boldsymbol{T}_{sd} = \boldsymbol{T}(\boldsymbol{\theta}_d)$$




Algorithm in Body Frame:

a) **Initialization**: Given $\boldsymbol{T}_{sd} \in SE(3)$ and an initial guess $\boldsymbol{\theta}^0 \in \mathbb{R}^n$, set $k = 0$.

b) **Iteration**: Set $[\boldsymbol{\mathcal{E}}_b] = \log\left(\boldsymbol{T}_{bd}(\boldsymbol{\theta}^k)\right) = \log(\boldsymbol{T}_{sb}^{-1}(\boldsymbol{\theta}^k)\boldsymbol{T}_{sd})$. While $\|\boldsymbol{\mathcal{E}}_{b,\omega}\| > \epsilon_\omega$ or $\|\boldsymbol{\mathcal{E}}_{b,v}\| > \epsilon_v$ for some small $\epsilon_\omega, \epsilon_v \in \mathbb{R}$, where $\boldsymbol{\mathcal{E}}_b = (\boldsymbol{\mathcal{E}}_{b,\omega}, \boldsymbol{\mathcal{E}}_{b,v})$:
- Set $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \lambda \boldsymbol{J}_b^{+}(\boldsymbol{\theta}^k)\boldsymbol{\mathcal{E}}_b$.
- Increment $k$.

($\boldsymbol{\mathcal{E}}_b$ is the twist that takes $\boldsymbol{T}_{sb}$ to $\boldsymbol{T}_{sd}$ in 1s)

($\epsilon_\omega$ has the unit of radian and the dimension of $\epsilon_v$ is length)

$(0 < \lambda \leq 1)$

Algorithm in Space Frame:

a) **Initialization**: Given $\boldsymbol{T}_{sd} \in SE(3)$ and an initial guess $\boldsymbol{\theta}^0 \in \mathbb{R}^n$, set $k = 0$.

b) **Iteration**: Set $[\boldsymbol{\mathcal{E}}_s] = [\mathrm{Ad}_{\boldsymbol{T}_{sb}}]\log\left(\boldsymbol{T}_{bd}(\boldsymbol{\theta}^k)\right) = [\mathrm{Ad}_{\boldsymbol{T}_{sb}}]\log(\boldsymbol{T}_{sb}^{-1}(\boldsymbol{\theta}^k)\boldsymbol{T}_{sd})$. While $\|\boldsymbol{\mathcal{E}}_{s,\omega}\| > \epsilon_\omega$ or $\|\boldsymbol{\mathcal{E}}_{s,v}\| > \epsilon_v$ for some small $\epsilon_\omega, \epsilon_v \in \mathbb{R}$, where $\boldsymbol{\mathcal{E}}_s = (\boldsymbol{\mathcal{E}}_{s,\omega}, \boldsymbol{\mathcal{E}}_{s,v})$:
- Set $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k + \lambda \boldsymbol{J}_s^{+}(\boldsymbol{\theta}^k)\boldsymbol{\mathcal{E}}_s$.
- Increment $k$.

($\boldsymbol{\mathcal{E}}_s$ is the twist that takes $\boldsymbol{T}_{sb}$ to $\boldsymbol{T}_{sd}$ in 1s)

# Jacobian (Pseudo-)Inverse Method
## (Minimum-Coordinate IK – Velocity Level)

Assume that the end-effector pose is represented by the minimum number of coordinates, i.e., $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{\theta}) \in \mathbb{R}^r$, $\boldsymbol{\theta} \in \mathbb{R}^n$ ($\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^r$), and $\dot{\boldsymbol{x}} = \boldsymbol{J}_a(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$. Let $\boldsymbol{x}_d(t)$ be the desired end-effector trajectory. Thus, the end-effector pose error, and its derivative are defined as

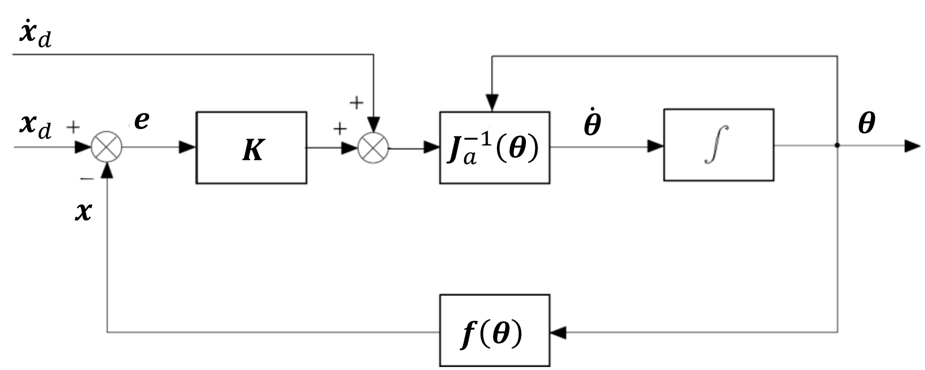
$$\boldsymbol{e} = \boldsymbol{x}_d - \boldsymbol{x} = \boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}) \qquad \dot{\boldsymbol{e}} = \dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}} = \dot{\boldsymbol{x}}_d - \boldsymbol{J}_a(\boldsymbol{\theta})\dot{\boldsymbol{\theta}}$$

On the assumption that matrix $\boldsymbol{J}_a$ is square ($n = r$) and nonsingular, the choice

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_a^{-1}(\boldsymbol{\theta})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}) \qquad (*)$$

where $\boldsymbol{K} \in \mathbb{R}^{r\times r}$ is a positive definite (usually diagonal) matrix, leads to the closed-loop system $\dot{\boldsymbol{e}} + \boldsymbol{K}\boldsymbol{e} = \boldsymbol{0}$ which is a linear system and is **asymptotically stable**.

Thus, the error $\boldsymbol{e}$ tends to zero along the trajectory with a convergence rate that depends on the eigenvalues of matrix $\boldsymbol{K}$ (the larger the eigenvalues, the faster the convergence).



# Jacobian (Pseudo-)Inverse Method

## (Minimum-Coordinate IK – Velocity Level)

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_a^{-1}(\boldsymbol{\theta})(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}) \quad \rightarrow$$

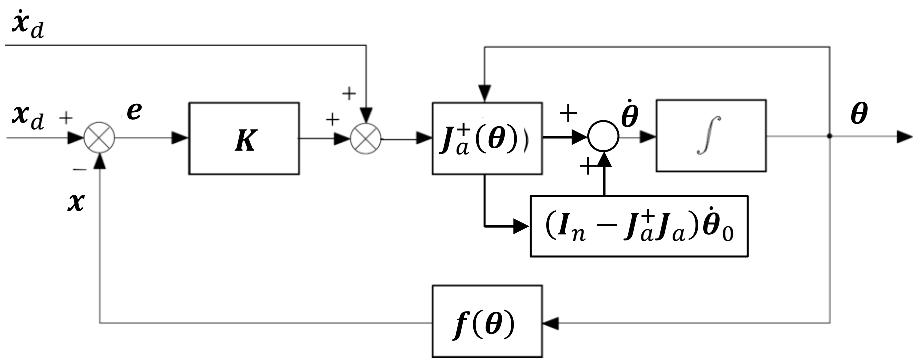
$$\boldsymbol{\theta}(t_{k+1}) = \boldsymbol{\theta}(t_k) + \dot{\boldsymbol{\theta}}(t_k)\Delta t = \boldsymbol{\theta}(t_k) + \boldsymbol{J}_a^{-1}(\boldsymbol{\theta}(t_k))\big(\dot{\boldsymbol{x}}_d(t_k) + \boldsymbol{K}\boldsymbol{e}(t_k)\big)\Delta t$$

$$= \boldsymbol{\theta}(t_k) + \boldsymbol{J}_a^{-1}(\boldsymbol{\theta}(t_k))\left(\dot{\boldsymbol{x}}_d(t_k) + \boldsymbol{K}\left(\boldsymbol{x}_d(t_k) - \boldsymbol{f}(\boldsymbol{\theta}(t_k))\right)\right)\Delta t \qquad k = 0,1,2,\ldots$$

**Note**: This equation for $\dot{\boldsymbol{x}}_d = \boldsymbol{0}$ (i.e., a constant end-effector pose $\boldsymbol{x}_d$) corresponds to the configuration-level IK based on Newton–Raphson Method.

**Note**: In the case of a **redundant manipulator**, the solution $(*)$ can be generalized into

$$\dot{\boldsymbol{\theta}} = \boldsymbol{J}_a^{+}(\dot{\boldsymbol{x}}_d + \boldsymbol{K}\boldsymbol{e}) + (\boldsymbol{I}_n - \boldsymbol{J}_a^{+}\boldsymbol{J}_a)\dot{\boldsymbol{\theta}}_0$$



# Jacobian Transpose Method

# Jacobian Transpose Method
## (Minimum-Coordinate IK – Configuration Level)

Let's define an optimization problem as $\min_{\boldsymbol{\theta}} F(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{2}\left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta})\right)^T \left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta})\right)$

The gradient of the cost function $F(\boldsymbol{\theta}) \in \mathbb{R}$ is $\boldsymbol{\nabla} F(\boldsymbol{\theta}) = -\boldsymbol{J}_a^T(\boldsymbol{\theta})(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}))$.

A **Gradient Descent** algorithm to minimize $F(\boldsymbol{\theta})$ is

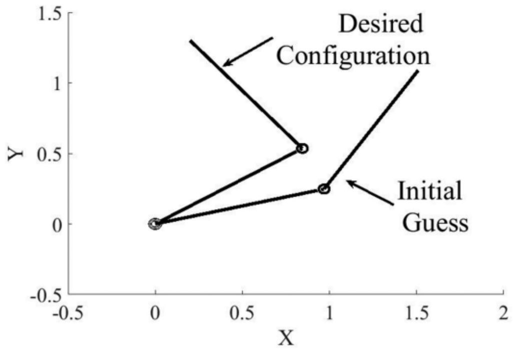
$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \lambda\, \boldsymbol{\nabla} F(\boldsymbol{\theta}_k) = \boldsymbol{\theta}^k + \lambda\, \boldsymbol{J}_a^T(\boldsymbol{\theta}^k)\left(\boldsymbol{x}_d - \boldsymbol{f}(\boldsymbol{\theta}^k)\right)$$

where $0 < \lambda \leq 1$ is the step length where can be adjusted to aid convergence.

# Jacobian Transpose vs Jacobian Inverse

- Jacobian transpose method is computationally more efficient to compute than the Jacobian inverse method.
- Jacobian transpose does not suffer from kinematic singularities.
- The convergence of Jacobian transpose, in terms of number of iterations, may be slower than the Jacobian inverse method.

Consider the following 2R robot where the desired end-effector coordinate is $\boldsymbol{x}_d = (0.2, 1.3)$, the joint variables corresponding to $\boldsymbol{x}_d$ are $\theta_1 = 0.5650$ and $\theta_2 = 0.7062$, the initial guess are $\theta_1 = 0.25$ and $\theta_2 = 0.75$, and the step size is $0.75$.




| Iteration | $\theta_1$ | $\theta_2$ |
|---|---|---|
| 1 | −0.33284 | 2.6711 |
| 2 | 0.80552 | 2.1025 |
| 3 | 0.46906 | 1.9316 |
| 4 | 0.53554 | 1.7697 |
| 5 | 0.55729 | 1.7227 |
| 6 | 0.56308 | 1.7104 |
| 7 | 0.56455 | 1.7073 |
| 8 | 0.56492 | 1.7065 |
| 9 | 0.56501 | 1.7063 |
| 10 | 0.56503 | 1.7062 |

IK using Jacobian inverse

| Iteration | $\theta_1$ | $\theta_2$ |
|---|---|---|
| 1 | 1.8362 | 1.3412 |
| 2 | 0.4667 | 1.1025 |
| 3 | 1.1215 | 1.6233 |
| 4 | 0.45264 | 1.415 |
| 5 | 0.83519 | 1.7273 |
| 26 | 0.56522 | 1.7063 |
| 27 | 0.56492 | 1.7061 |
| 28 | 0.56514 | 1.7063 |
| 29 | 0.56498 | 1.7062 |
| 30 | 0.5650 | 1.7062 |

IK using Jacobian transpose

# Orientation Error

# Orientation Error for Minimum-Coordinate Representation

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{e}_R \\ \boldsymbol{e}_p \end{bmatrix} = \begin{bmatrix} \boldsymbol{e}_R \\ \boldsymbol{p}_d - \boldsymbol{p} \end{bmatrix}$$

Computation of $\boldsymbol{e}_R$ depends on the particular representation of end-effector orientation, namely, Euler angles, exponential coordinates (angle and axis), and unit quaternion:

**(1) Euler Angles**: Method 1: $\boldsymbol{e}_R = \boldsymbol{\phi}_d - \boldsymbol{\phi} \in \mathbb{R}^3$

Method 2: $\boldsymbol{e}_R = \text{EulerAngles}\left(\boldsymbol{R}_{sb}^T \boldsymbol{R}_{sd}\right) = \text{EulerAngles}(\boldsymbol{R}_{bd}) \in \mathbb{R}^3$

Assumption: There is no kinematic or representation singularities.

# Orientation Error for Minimum-Coordinate Representation

**(2) Exponential Coordinates (Angle and Axis)**:

$$\boldsymbol{R}_{bd} = \boldsymbol{R}_{sb}^T \boldsymbol{R}_{sd}\ ,\quad \log(\boldsymbol{R}_{bd}) = [\hat{\boldsymbol{\omega}}_b]\theta \quad ,$$

(in EE frame)

$$\boldsymbol{e}_R := \hat{\boldsymbol{\omega}}_b \theta$$ (in EE frame)

$$\boldsymbol{e}_R := \boldsymbol{R}_{sb} \hat{\boldsymbol{\omega}}_b \theta$$ (in base frame)

**(3) Unit Quaternion**:

$$\boldsymbol{R}_{bd} = \boldsymbol{R}_{sb}^T \boldsymbol{R}_{sd}\ ,\ \mathrm{UnitQuat}(\boldsymbol{R}_{bd}) = \begin{bmatrix} \cos\theta/2 \\ \sin\theta/2\ \hat{\boldsymbol{\omega}}_b \end{bmatrix} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

(in EE frame)