深蓝学院
shenlanxueyuan.com

Foundations of Robotics

Lec 6: Inverse Kinematics

主讲人 滕瀚哲
美国加州大学河滨分校
ARCS实验室博士

Outline

1. Inverse Kinematics
2. Analytic Approach (Trigonometry)
3. Numerical Approach (Newton’s Method)
4. Homework

Outline

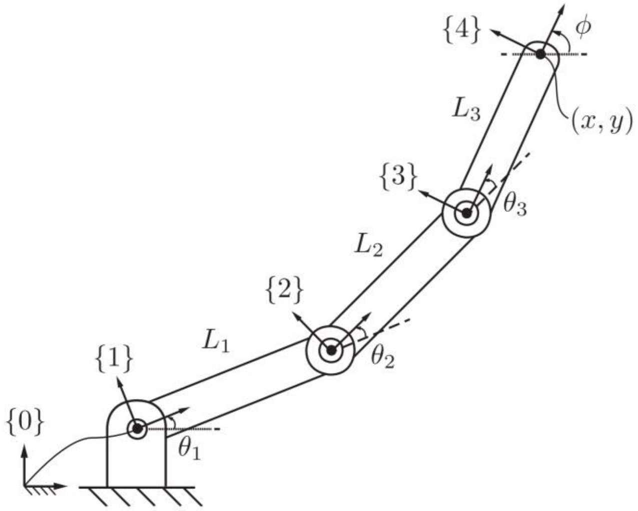
1. Inverse Kinematics
2. Analytic Approach (Trigonometry)
3. Numerical Approach (Newton’s Method)
4. Homework

Inverse Kinematics

Consider a general n degree-of-freedom open chain.

Forward Kinematics
- $\theta \in \mathbb{R}^n \quad \rightarrow \quad T(\theta) \in SE(3)$
- Given joint coordinates θ, calculate the configuration (position and orientation) $T(\theta)$ of the end-effector frame.

Inverse Kinematics
- $T(\theta) \in SE(3) \quad \rightarrow \quad \theta \in \mathbb{R}^n$
- Given a homogeneous transform $X \in SE(3)$ and forward kinematics $T(\theta)$, find solutions θ that satisfy $T(\theta) = X$.



Inverse Kinematics

Example: a 2R planar open-chain manipulator

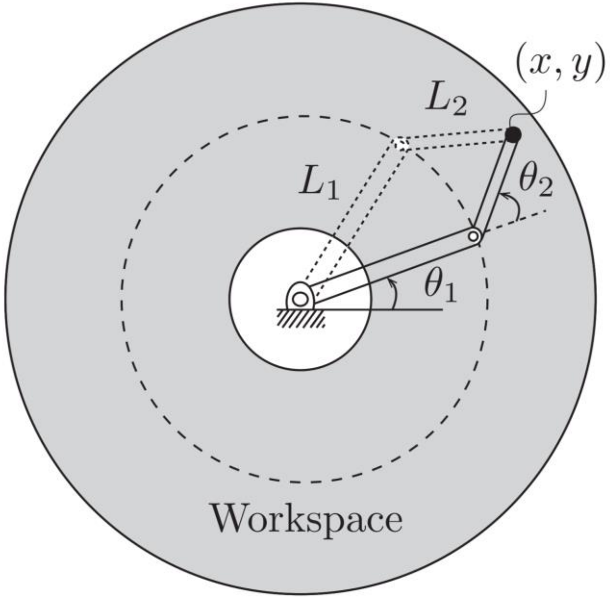
$$T(\theta) \in SE(2) \quad \rightarrow \quad \theta \in \mathbb{R}^2$$

Considering only the position of end-effector and ignoring its orientation, the forward kinematics can be expressed as

$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$$

Provided a desired end-effector position (x, y), there will be either zero, one or two solutions depending on where (x, y) lies.

Assuming $L_1 > L_2$

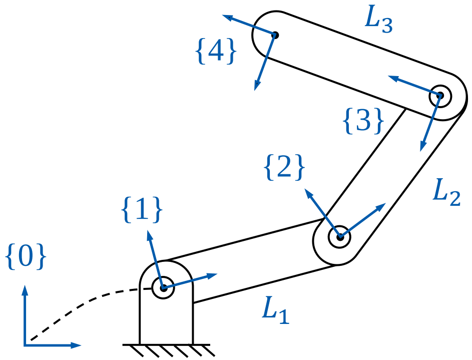


Inverse Kinematics

Example: a 3R planar open-chain manipulator

$$T(\theta) \in SE(2) \quad \rightarrow \quad \theta \in \mathbb{R}^3$$

Considering only the position of end-effector and ignoring its orientation, the forward kinematics can be expressed as



$$x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) + L_3 \cos(\theta_1 + \theta_2 + \theta_3)$$
$$y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) + L_3 \sin(\theta_1 + \theta_2 + \theta_3)$$

Provided a desired end-effector position (x, y), there will be either zero, one or multiple solutions depending on where (x, y) lies.

Inverse Kinematics

Two ways to solve inverse kinematics:

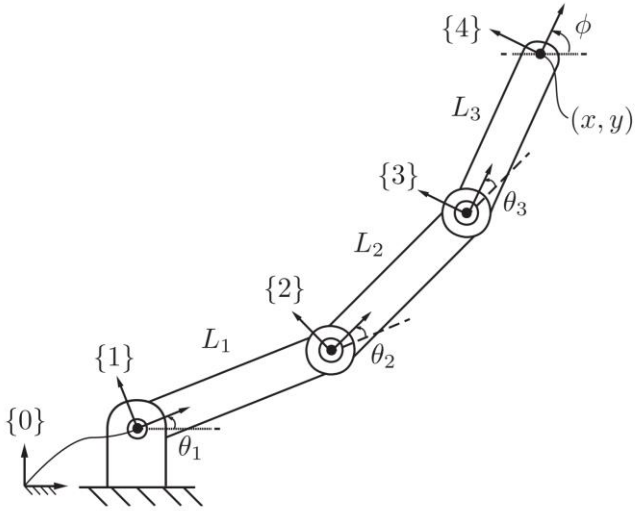
Analytic Approach

- Derive equations based on the geometric structure (e.g., trigonometry) of the mechanism
- Provide zero, one or multiple solutions in closed-form

Numerical Approach

- Iteratively solve a non-linear optimization problem (e.g., by Newton-Raphson method)
- Require an initial guess; always produce one solution (best approximation, may not be optimal)

In practice, they both can be applied: analytic solution can serve as the initial guess for numerical methods.

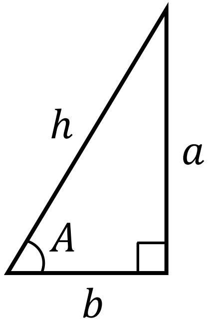


Outline

1. Inverse Kinematics
2. Analytic Approach (Trigonometry)
3. Numerical Approach (Newton’s Method)
4. Homework

Analytic Approach (Trigonometry)

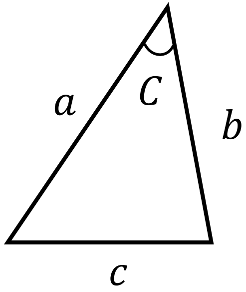
Math tools from trigonometry:



$\sin A$, $\cos A$, $\tan A$

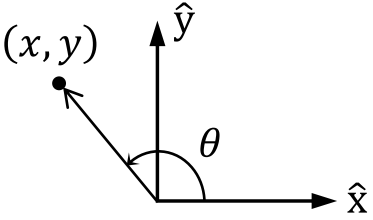
$\operatorname{asin}\frac{a}{h}, \operatorname{acos}\frac{b}{h}, \operatorname{atan}\frac{a}{b}$

Trigonometric ratios



$c^2 = a^2 + b^2 - 2ab\cos C$

Law of cosines



$\theta = \operatorname{atan2}(y, x) \in (-\pi, \pi]$

Two-argument arctangent

Analytic Approach (Trigonometry)

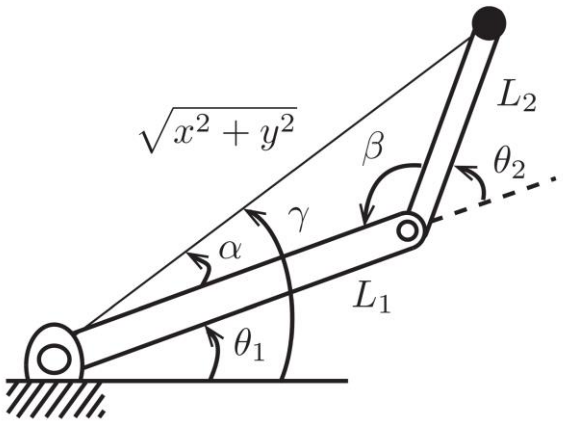
Example: A 2R planar open-chain manipulator

In the current "elbow-down" setup, as shown in the figure, joint angles can be obtained as follows.

- Compute $\gamma = \text{atan2}(y, x)$
- Compute α from law of cosines
- Obtain $\theta_1 = \gamma - \alpha$
- Compute β from law of cosines
- Obtain $\theta_2 = \pi - \beta$

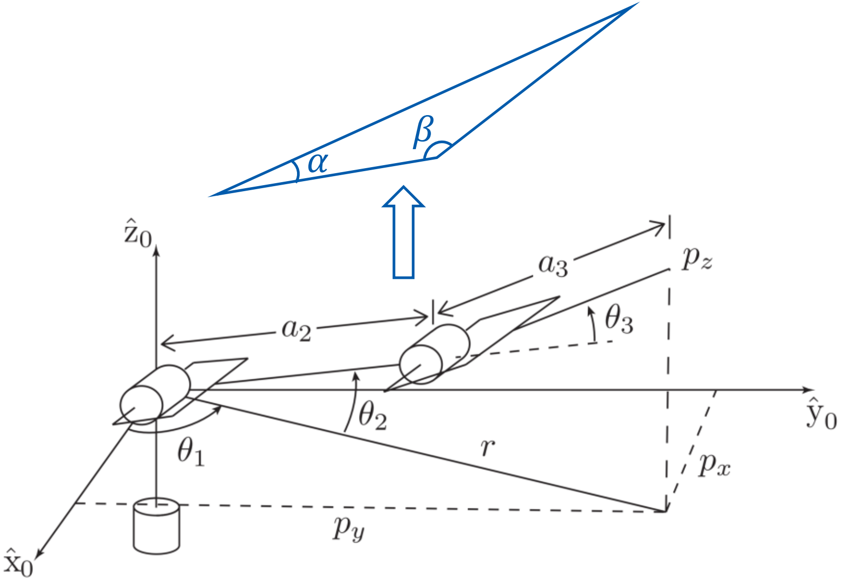
The other solution in the "elbow-up" setup can be obtained similarly: $\theta_1 = \gamma + \alpha$, $\theta_2 = \beta - \pi$

Again, no solution exists if (x, y) is not reachable, and exactly one solution exists if (x, y) lies on the boundary.



Analytic Approach (Trigonometry)

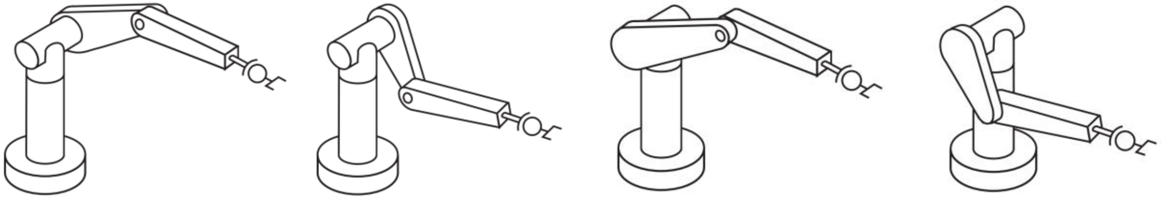
Example: A 3R spatial open-chain manipulator



One possible solution:

- Obtain $\theta_1 = \mathrm{atan2}(p_y, p_x)$
- Compute $\gamma = \mathrm{atan}(p_z/r)$
- Compute α from law of cosines
- Obtain $\theta_2 = \gamma - \alpha$
- Compute β from law of cosines
- Obtain $\theta_3 = \pi - \beta$

A more challenging case: what if joint axes are not aligned with coordinate axes (i.e. with an offset)?



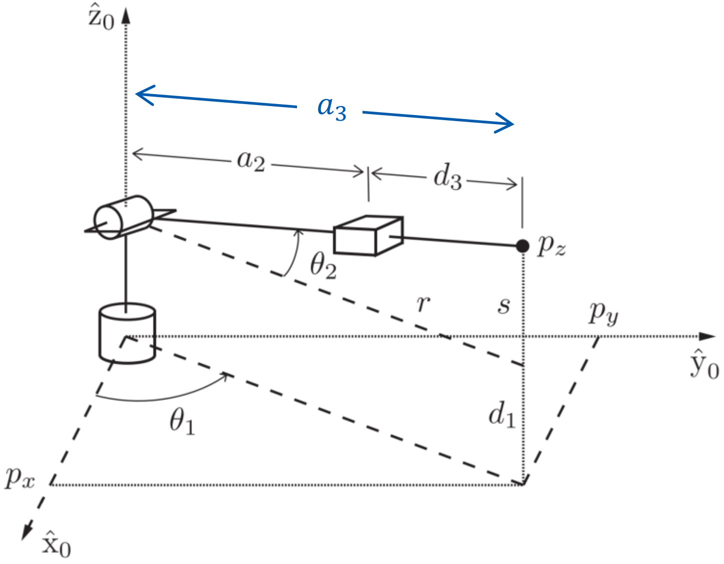
Analytic Approach (Trigonometry)

Example: An RRP spatial open-chain manipulator

One possible solution:

- Obtain $\theta_1 = \text{atan2}(p_y, p_x)$
- Compute $s = p_z - d_1$
- Compute $r = \sqrt{p_x^2 + p_y^2}$
- Obtain $\theta_2 = \text{atan}\, s/r$
- Compute $a_3 = \sqrt{r^2 + s^2}$
- Obtain $d_3 = a_3 - a_2$

Again, the solution can be more complicated if the shoulder joint is placed with an offset.



Outline

1. Inverse Kinematics
2. Analytic Approach (Trigonometry)
3. Numerical Approach (Newton’s Method)
4. Homework

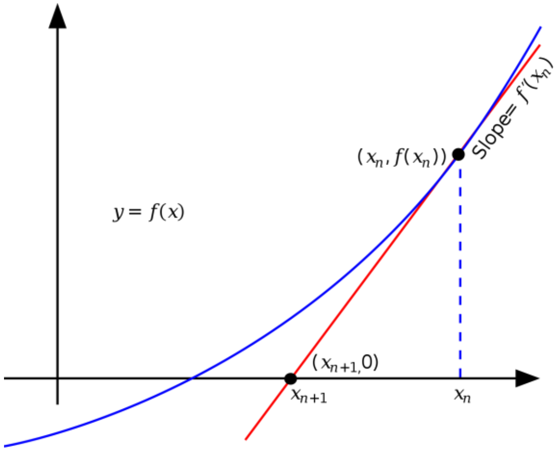
Numerical Approach (Newton's Method)

Newton-Raphson Method (a.k.a. Newton's Method)

- An iterative approach for finding the roots of a nonlinear equation.

Example of a scalar function

- Problem to solve: $f(x) = 0$
- Initial guess: $x = x_0$
- At the nth iteration: $x = x_n$
- Taylor expansion of $f(x)$ at x_n:
  $f(x) = f(x_n) + f'(x_n)(x - x_n) + h.o.t.$
- Ignore higher-order terms, set $f(x) = 0$ and solve for x to obtain $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$
- Repeat until some stopping criterion is satisfied



* Figure taken from wikipedia: https://en.wikipedia.org/wiki/Newton's_method

Numerical Approach (Newton's Method)

Newton's Method in Optimization

- Find the roots of the derivative of a twice-differentiable function (solution to $f'(x) = 0$).

Example of a scalar function

- Problem to solve

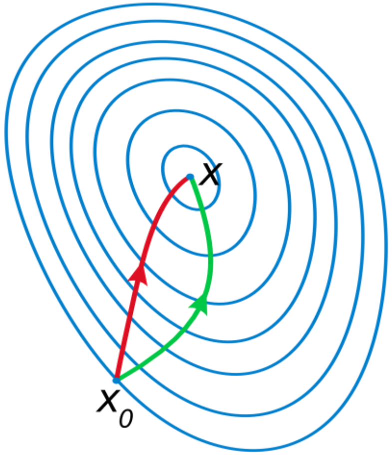
$$\min f(x)$$

- In each iteration, Taylor expansion of $f(x)$ at x_n up to second order terms

$$f'(x) = f'(x_n) + f''(x_n)(x - x_n) = 0$$

- Update equation

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}$$



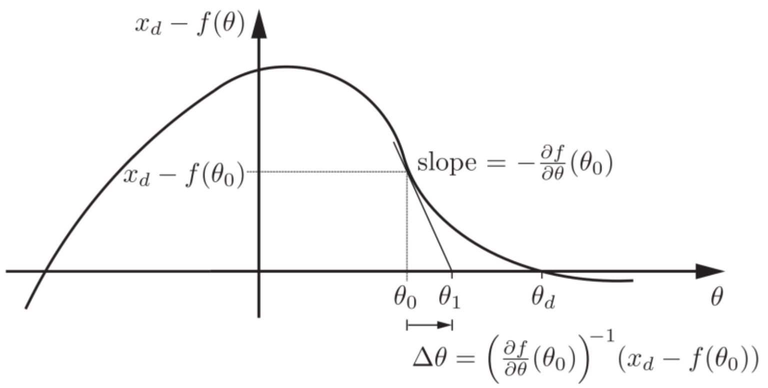
Green: Gradient Descent; Red: Newton's Method

* Figure taken from wikipedia: https://en.wikipedia.org/wiki/Newton's_method_in_optimization

Numerical Approach (Newton's Method)

Newton's Method in Inverse Kinematics

- Inverse kinematics: find solution to $T(\theta) = X$
- Define $g(\theta) = x_d - f(\theta) = X - T(\theta)$
- Solve $g(\theta) = 0$ using Newton's method



Example of a vector function

- Problem to solve

$$g(\theta) = x_d - f(\theta) = 0$$

- Taylor expansion

$$x_d = f(\theta) = f(\theta^k) + J(\theta^k)(\theta - \theta^k) + h.o.t.$$

- Update equation

$$\theta^{k+1} = \theta^k + J^{-1}(\theta^k)(x_d - f(\theta^k))$$

$$J(\theta^k) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1^k} & \cdots & \frac{\partial f_1}{\partial \theta_n^k} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1^k} & \cdots & \frac{\partial f_m}{\partial \theta_n^k} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Note: J^{-1} exists only when J is square and nonsingular!

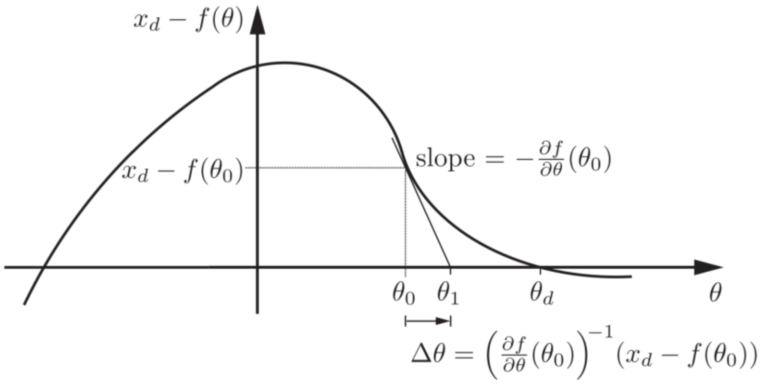
Numerical Approach (Newton's Method)

In cases where J is not invertible, we can apply pseudoinverse $J^{\dagger}$ instead.

1. The robot has more joints n than the end-effector coordinates m ("fat" Jacobian)
   - $J^{\dagger} = J^{\top}\left(JJ^{\top}\right)^{-1}$
   - Called a right inverse since $JJ^{\dagger} = I$

2. The robot has fewer joints n than the end-effector coordinates m ("tall" Jacobian)
   - $J^{\dagger} = \left(J^{\top}J\right)^{-1}J^{\top}$
   - Called a left inverse since $J^{\dagger}J = I$

In practice, we often use linear solvers to avoid the computation of inverse matrices.



$$J(\theta^k) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1^k} & \cdots & \frac{\partial f_1}{\partial \theta_n^k} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1^k} & \cdots & \frac{\partial f_m}{\partial \theta_n^k} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

Numerical Approach (Newton's Method)

Algorithm to reach a provided 3D point

1. Initialization
   - Given an end-effector position $x_d \in \mathbb{R}^m$
   - Given an initial guess $\theta^0 \in \mathbb{R}^n$
   - Set $k = 0$
2. While $\left\|x_d - f\left(\theta^k\right)\right\| > \epsilon$ for some small ϵ:
   - Compute $\Delta\theta = J^\dagger\left(\theta^k\right)\left(x_d - f\left(\theta^k\right)\right)$
   - Update $\theta^{k+1} = \theta^k + \Delta\theta$
   - Increment k

Right inverse: $J^\dagger = J^\top\left(JJ^\top\right)^{-1}$

Left inverse: $J^\dagger = \left(J^\top J\right)^{-1}J^\top$

$$J\left(\theta^k\right) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1^k} & \cdots & \frac{\partial f_1}{\partial \theta_n^k} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1^k} & \cdots & \frac{\partial f_m}{\partial \theta_n^k} \end{bmatrix} \in \mathbb{R}^{m\times n}$$

Numerical Approach (Newton's Method)

Algorithm to reach a provided 6D pose

1. Initialization
   - Given an end-effector pose $T_{sd} \in SE(3)$
   - Given an initial guess $\theta^0 \in \mathbb{R}^n$
   - Set $k = 0$
2. While $\|\omega_b\| > \epsilon_\omega$ or $\|v_b\| > \epsilon_v$ for some small ϵ_ω and ϵ_v :
   - Compute $[\mathcal{V}_b] = \log\left(T_{sb}^{-1}(\theta^k)T_{sd}\right)$
   - Compute $\Delta\theta = J^\dagger(\theta^k)\mathcal{V}_b$
   - Update $\theta^{k+1} = \theta^k + \Delta\theta$
   - Increment k

Jacobian should act on body twist

desired state: T_{sd}
current state: $T_{sb}(\theta^k)$
difference: $T_{bd}(\theta^k) = T_{bs}(\theta^k)T_{sd} = T_{sb}^{-1}(\theta^k)T_{sd}$

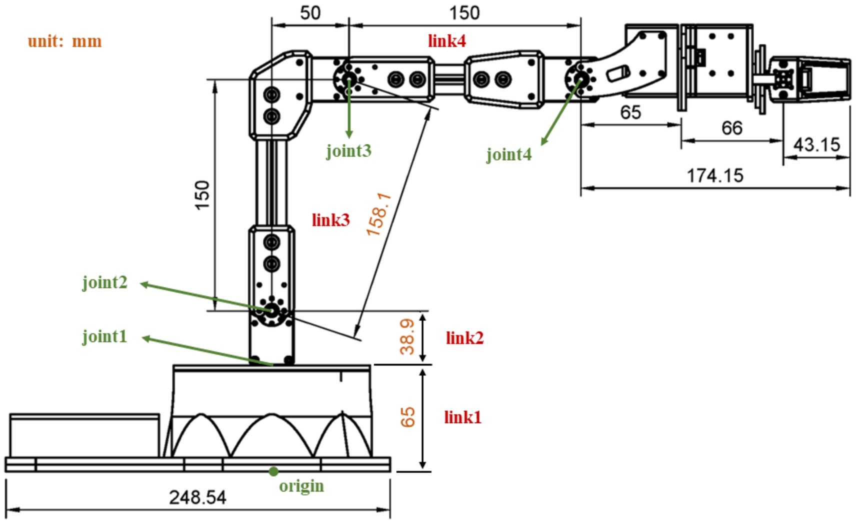
Right inverse: $J^\dagger = J^\top(JJ^\top)^{-1}$

Left inverse: $J^\dagger = (J^\top J)^{-1}J^\top$

$$J(\theta^k) = \begin{bmatrix} \frac{\partial f_1}{\partial \theta_1^k} & \cdots & \frac{\partial f_1}{\partial \theta_n^k} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial \theta_1^k} & \cdots & \frac{\partial f_m}{\partial \theta_n^k} \end{bmatrix} \in \mathbb{R}^{m\times n}$$

Outline

Homework



(1) Lab Assignments: Inverse Kinematics

- Provided the schematic of a 3R manipulator, write two scripts (in C++ or Python) to implement analytic and numerical methods to solve the inverse kinematics.
- The scripts will be tested using a few test cases (input: the position of the end effector; expected output: joint variables)

深蓝学院
shenlanxueyuan.com

Thanks for Listening！