Inverse Kinematics

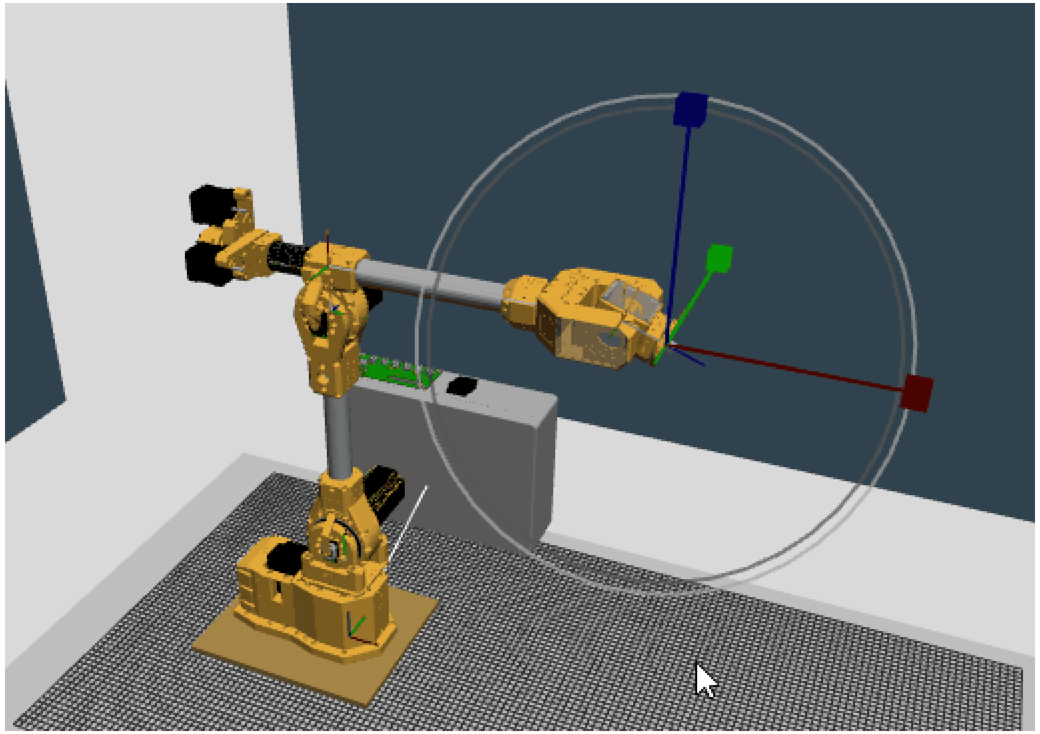
CS 6301 Special Topics: Introduction to Robot Manipulation and Navigation

Professor Yu Xiang

The University of Texas at Dallas

Robot Kinematics

- The relationship between a robot's joint coordinates and its spatial layout



https://www.marginallyclever.com/2020/04/gradient-descent-inverse-kinematics-for-6dof-robot-arms/

Inverse Kinematics

- Calculation of the joint coordinates given the position and orientation of its end-effector

End-effector transformation

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

→

Joint coordinates θ

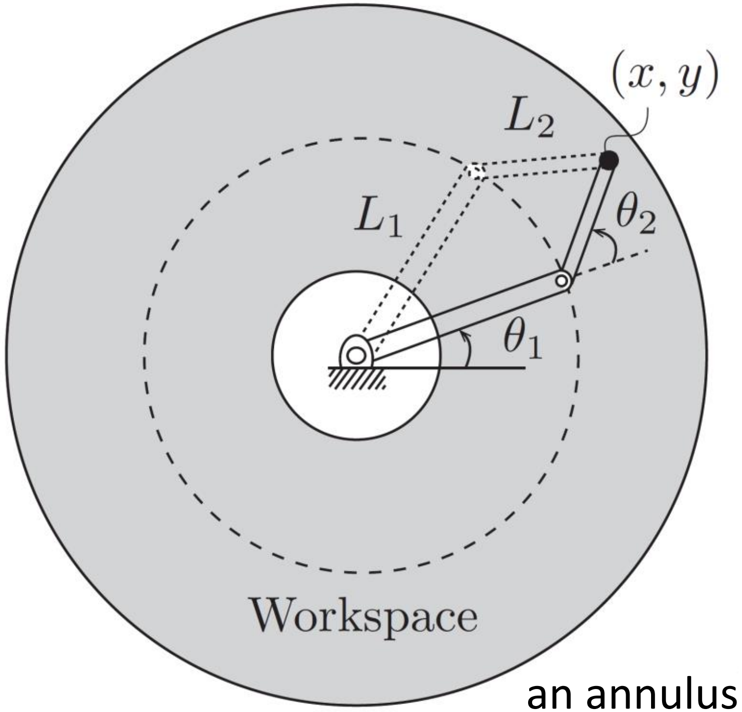
Inverse Kinematics

- For a n degree-of-freedom open chain with forward kinematics
  $T(\theta) \quad \theta \in \mathbb{R}^n$

- Given a homogenous transformation $X \in SE(3)$

- Find solutions θ such that $T(\theta) = X$

Inverse Kinematics



an annulus

Two-link planar open chain

Forward kinematics

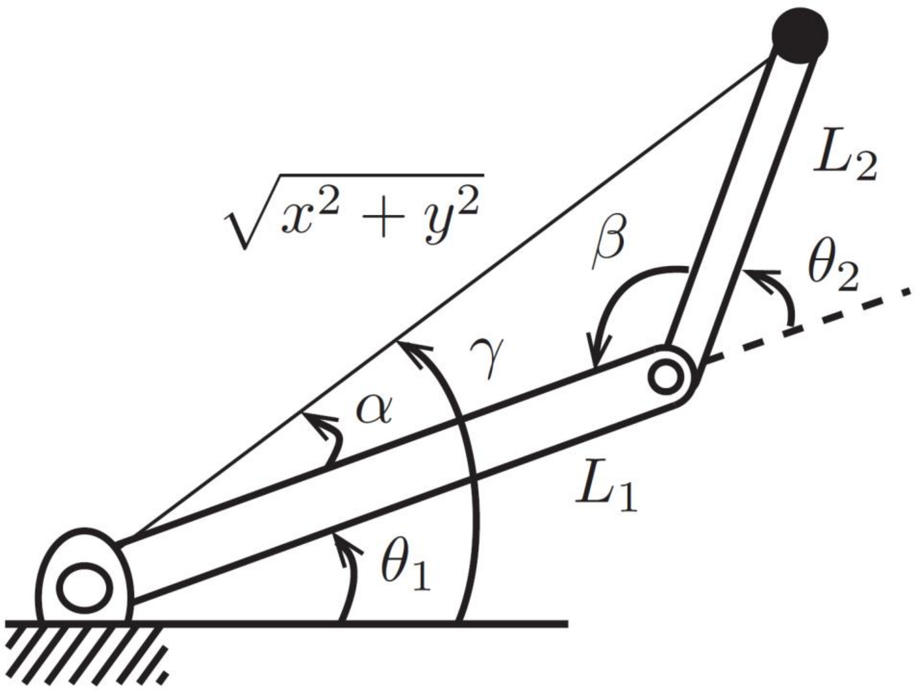
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) \\ L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2) \end{bmatrix}$$

Assuming $L_1 > L_2$

Give (x, y) Find solutions for (θ_1, θ_2)

There can be zero, one,
or two solutions for (θ_1, θ_2)

Inverse Kinematics



Two-link planar open chain

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

$$L_1^2 + L_2^2 - 2L_1L_2\cos\beta = x^2 + y^2$$

$$\beta = \cos^{-1}\left(\frac{L_1^2 + L_2^2 - x^2 - y^2}{2L_1L_2}\right)$$

$$\alpha = \cos^{-1}\left(\frac{x^2 + y^2 + L_1^2 - L_2^2}{2L_1\sqrt{x^2+y^2}}\right)$$

$$\gamma = \text{atan2}(y, x) \quad (-\pi, \pi]$$

righty solution $\theta_1 = \gamma - \alpha, \quad \theta_2 = \pi - \beta$

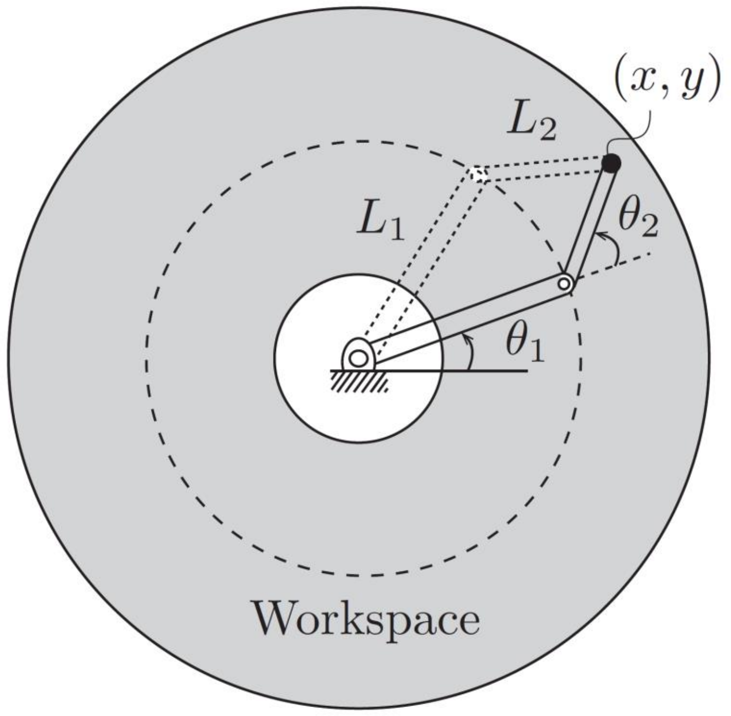
lefty solution $\theta_1 = \gamma + \alpha, \quad \theta_2 = \beta - \pi$

Inverse Kinematics

- IK can have multiple solutions
- FK only has a single solution
- Find solutions θ such that

$$T(\theta) = X$$

- Finding the roots of a nonlinear equation



Analytic Inverse Kinematics

Newton-Raphson Method

- Solve $g(\theta) = 0$ $\quad g : \mathbb{R} \rightarrow \mathbb{R}$ differentiable
- Initial guess θ^0
- Taylor expansion

$$g(\theta) = g(\theta^0) + \frac{\partial g}{\partial \theta}(\theta^0)(\theta - \theta^0) + \text{higher-order terms (h.o.t)}$$

$$\text{set } g(\theta) = 0 \qquad \theta = \theta^0 - \left(\frac{\partial g}{\partial \theta}(\theta^0)\right)^{-1} g(\theta^0)$$

$$\theta^{k+1} = \theta^k - \left(\frac{\partial g}{\partial \theta}(\theta^k)\right)^{-1} g(\theta^k)$$

Newton-Raphson Method

- When g is multi-dimensional $g : \mathbb{R}^n \to \mathbb{R}^n$

$$\frac{\partial g}{\partial \theta}(\theta) = \begin{bmatrix} \frac{\partial g_1}{\partial \theta_1}(\theta) & \cdots & \frac{\partial g_1}{\partial \theta_n}(\theta) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_n}{\partial \theta_1}(\theta) & \cdots & \frac{\partial g_n}{\partial \theta_n}(\theta) \end{bmatrix} \in \mathbb{R}^{n \times n}$$

Jacobian matrix

Numerical Inverse Kinematics Algorithm

- Forward kinematics $x = f(\theta)$ $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$
- Desired end-effector coordinates x_d
- Objective function for the Newton-Raphson method

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

- Goal

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

- Initial guess θ^0

Numerical Inverse Kinematics Algorithm

- Taylor expansion

$$x_d = f(\theta_d) = f(\theta^0) + \underbrace{\left.\frac{\partial f}{\partial \theta}\right|_{\theta^0}}_{J(\theta^0)} \underbrace{(\theta_d - \theta^0)}_{\Delta\theta} + \text{h.o.t.},$$

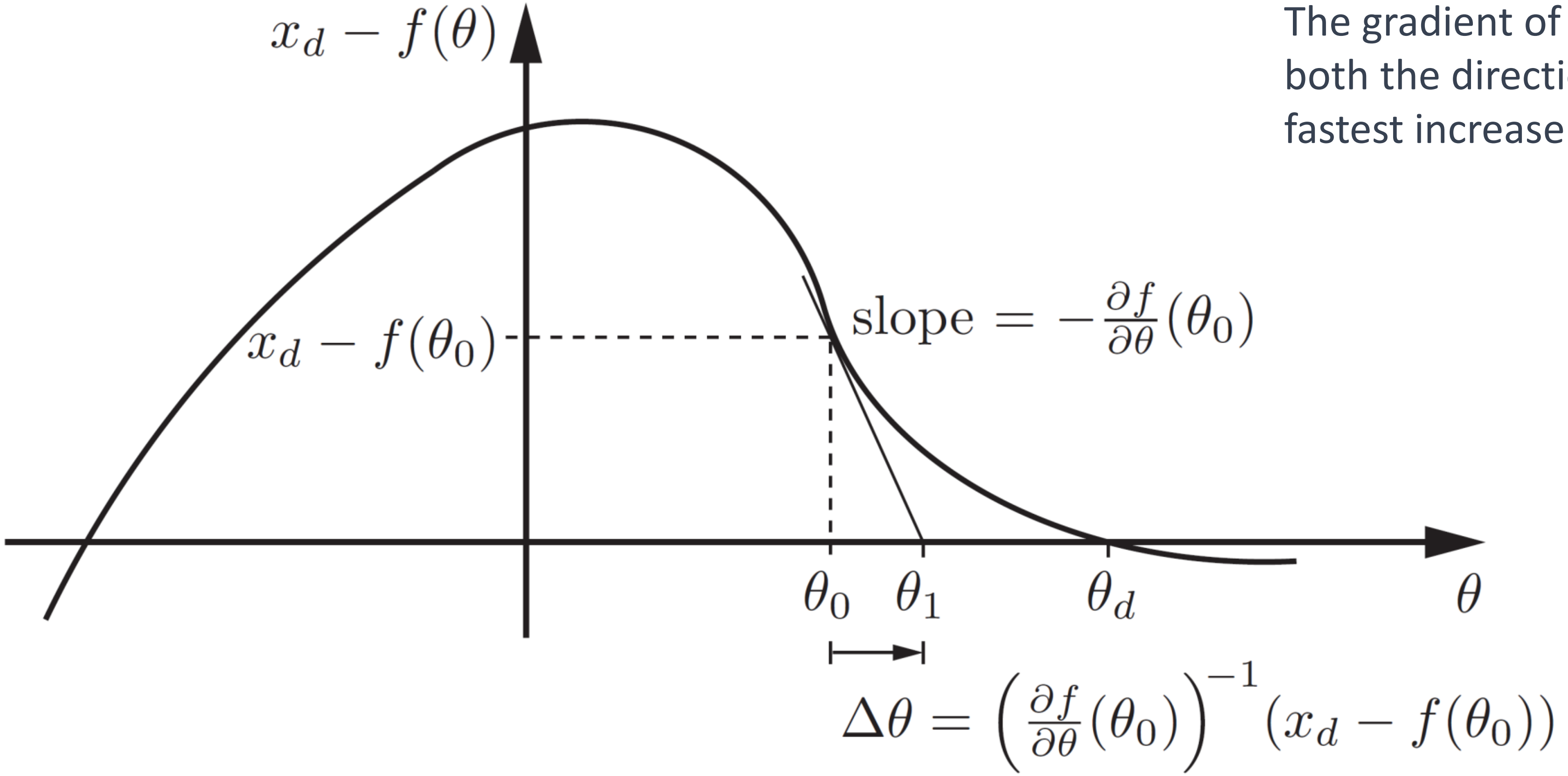
$J(\theta^0) \in \mathbb{R}^{m \times n}$ Jacobian

$$J(\theta^0)\Delta\theta = x_d - f(\theta^0)$$

Whe $J(\theta^0)$ is square and invertible $\Delta\theta = J^{-1}(\theta^0)\left(x_d - f(\theta^0)\right)$

$$\theta^1 = \theta^0 + \Delta\theta$$

Numerical Inverse Kinematics Algorithm



The gradient of a function represents both the direction and rate of the fastest increase of that function.

Numerical Inverse Kinematics Algorithm

- When J is not invertible, use pseudoinverse $J^{\dagger}$

$$Jy = z \qquad J \in \mathbb{R}^{m \times n},\ y \in \mathbb{R}^{n},\ \text{and}\ z \in \mathbb{R}^{m}$$

$$y^* = J^{\dagger} z$$

$$J^{\dagger} = J^{\mathrm{T}}(JJ^{\mathrm{T}})^{-1} \qquad \text{if } J \text{ is fat, } n > m \text{ (called a right inverse since } JJ^{\dagger} = I)$$

$$J^{\dagger} = (J^{\mathrm{T}}J)^{-1}J^{\mathrm{T}} \qquad \text{if } J \text{ is tall, } n < m \text{ (called a left inverse since } J^{\dagger}J = I).$$

$$\Delta\theta = J^{\dagger}(\theta^0)\left(x_d - f(\theta^0)\right)$$

Numerical Inverse Kinematics Algorithm

- Newton-Raphson iterative algorithm for inverse kinematics

- Initialization: given $x_d \in \mathbb{R}^m$, initial guess $\theta^0 \in \mathbb{R}^n$

- Set $e = x_d - f(\theta^i)$ While $||e|| > \epsilon$ for some small ϵ:
  - Set $\theta^{i+1} = \theta^i + J^\dagger(\theta^i)e$
  - Increment i

Numerical Inverse Kinematics Algorithm

- How to achieve a desired end-effector configuration $T_{sd} \in SE(3)$

- Given current configuration $T_{sb}(\theta^i)$

- We cannot simply compute error as $T_{sd} - T_{sb}(\theta^i)$

- Consider error $e = x_d - f(\theta^i)$ as a velocity vector
- Similarly, a body twist $\mathcal{V}_b \in \mathbb{R}^6$ will cause a motion $T_{sb}(\theta^i)$ to T_{sd}

Numerical Inverse Kinematics Algorithm

- How to achieve a desired end-effector configuration $T_{sd} \in SE(3)$
- Current configuration $T_{sb}(\theta^i)$

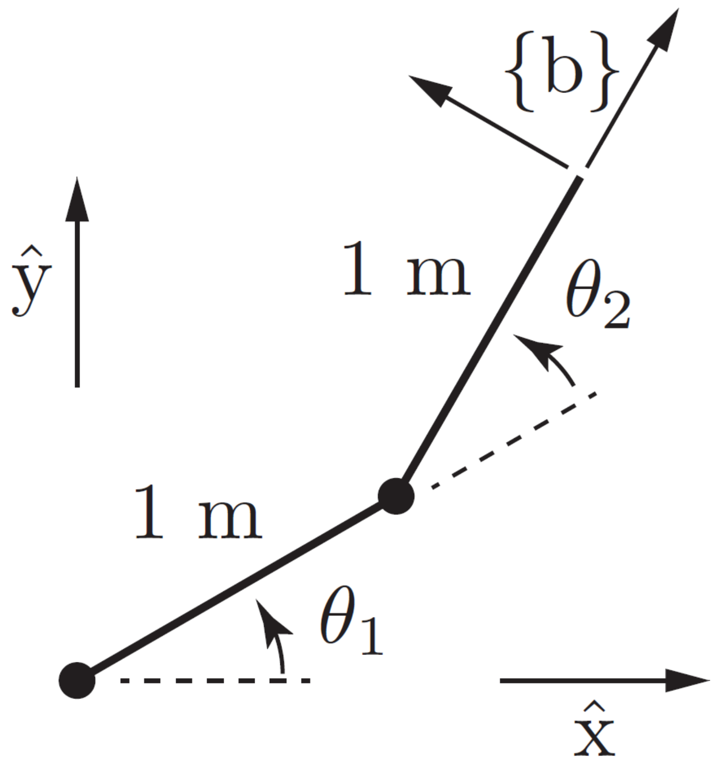
- Desired configuration $T_{bd}(\theta^i) = T_{sb}^{-1}(\theta^i) T_{sd} = T_{bs}(\theta^i) T_{sd}$
- Body twist $[\mathcal{V}_b] = \log T_{bd}(\theta^i)$

- Updating rule

Pseudoinverse of the body Jacobian

$$\theta^{i+1} = \theta^i + J_b^{\dagger}(\theta^i) \mathcal{V}_b$$

Numerical Inverse Kinematics



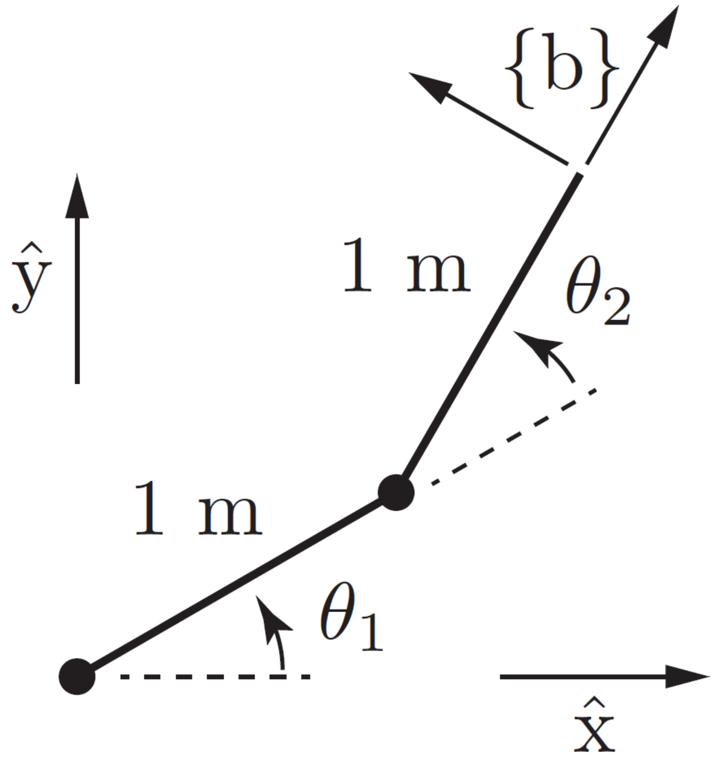
A 2R robot

Goal

$$(x, y) = (0.366 \text{ m}, 1.366 \text{ m})$$

$$T_{sd} = \begin{bmatrix} -0.5 & -0.866 & 0 & 0.366 \\ 0.866 & -0.5 & 0 & 1.366 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Numerical Inverse Kinematics



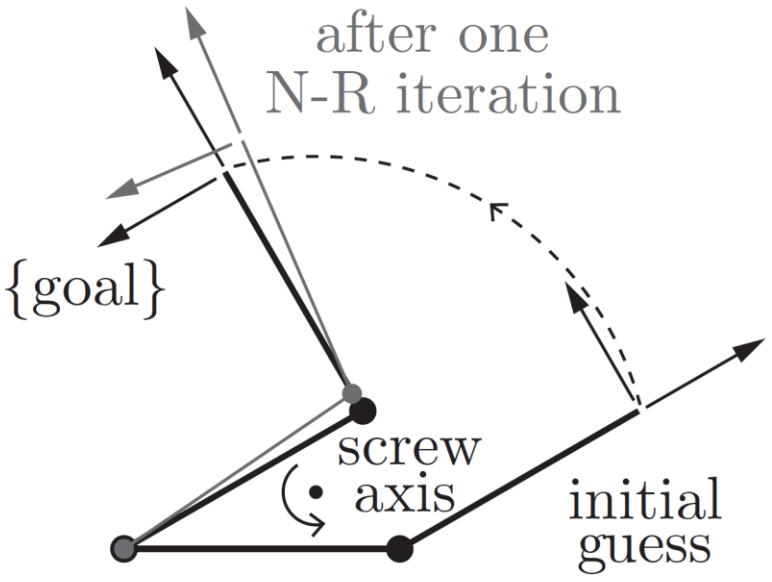
- Forward kinematics

$$M = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathcal{B}_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 2 \\ 0 \end{bmatrix} \quad \mathcal{B}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$T(\theta) = Me^{[\mathcal{B}]_1\theta_1}e^{[\mathcal{B}]_2\theta_2}$$

- Initial guess $\theta^0 = (0, 30^\circ)$

Numerical Inverse Kinematics



- Compute body twist

$$T_{bd}(\theta^i) = T_{sb}^{-1}(\theta^i)T_{sd} = T_{bs}(\theta^i)T_{sd}$$

$$[\mathcal{V}_b] = \log T_{bd}(\theta^i)$$

- Compute body Jacobian $J_b(\theta) \in \mathbb{R}^{6\times n}$

$$J_{bi}(\theta) = \mathrm{Ad}_{e^{-[\mathcal{B}_n]\theta_n} \cdots e^{-[\mathcal{B}_{i+1}]\theta_{i+1}}}(\mathcal{B}_i)$$

- Update $\theta^{i+1} = \theta^i + J_b^\dagger(\theta^i)\mathcal{V}_b$

i	(θ_1, θ_2)	(x, y)	$\mathcal{V}_b = (\omega_{zb}, v_{xb}, v_{yb})$	$\lVert\omega_b\rVert$	$\lVert v_b\rVert$
0	$(0.00, 30.00°)$	$(1.866, 0.500)$	$(1.571, 0.498, 1.858)$	1.571	1.924
1	$(34.23°, 79.18°)$	$(0.429, 1.480)$	$(0.115, -0.074, 0.108)$	0.115	0.131
2	$(29.98°, 90.22°)$	$(0.363, 1.364)$	$(-0.004, 0.000, -0.004)$	0.004	0.004
3	$(30.00°, 90.00°)$	$(0.366, 1.366)$	$(0.000, 0.000, 0.000)$	0.000	0.000

$$\theta_d = (30°, 90°)$$

Inverse Velocity Kinematics

- Find the joint velocity $\dot{\theta}$ to follow a desired end-effector trajectory $T_{sd}(t)$
- Method 1: uses inverse kinematics to compute $\theta_d(k\Delta t)$

Joint velocity $\dot{\theta} = \big(\theta_d(k\Delta t) - \theta((k-1)\Delta t)\big)/\Delta t$

interval $[(k-1)\Delta t, k\Delta t]$

- Method 2: uses $J\dot{\theta} = \mathcal{V}_d$

$$\dot{\theta} = J^{\dagger}(\theta)\mathcal{V}_d$$

Body twist $T_{sd}^{-1}(t)\dot{T}_{sd}(t)$ Spatial twist $\dot{T}_{sd}(t)T_{sd}^{-1}(t)$

Summary

- Inverse kinematics
- Newton-Raphson Method
- Numerical Inverse Kinematics Algorithm

Further Reading

- Chapter 6 and Appendix D in Kevin M. Lynch and Frank C. Park. Modern Robotics: Mechanics, Planning, and Control. 1st Edition, 2017.