

Robotics algorithms

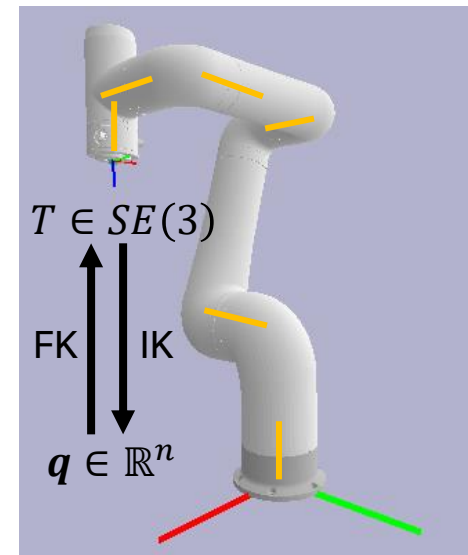
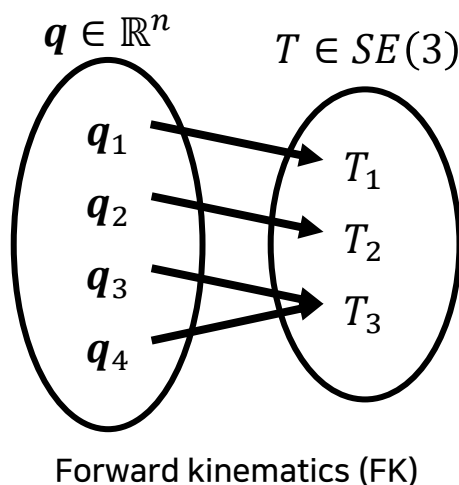
Lee Yongseok, Jung Hyunseo

dldydtjr2000@postech.ac.kr

1. Numerical IK

Inverse kinematics (IK)

The inverse kinematics (IK) is to find the joint position (called as joint variables) $\mathbf{q} \in \mathbb{R}^n$ for a given transformation matrix $T \in SE(3)$. The IK problem is a nonlinear problem; the existence and uniqueness of the solution are not always guaranteed, while the forward kinematics (FK) is **surjective**.

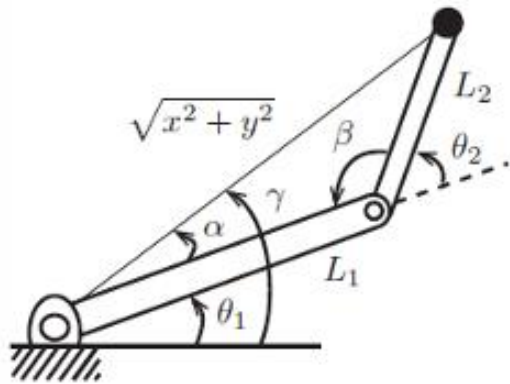


Inverse kinematics has two approaches: **closed-form solutions** and **numerical methods**. The closed-form solution uses geometry and polynomial equations, but there is no general solution for a 6-dof robotic system due to its complex structure. The numerical method has a computational burden but is easy to apply, and we will deal with the numerical methods in this lecture.

1. Numerical IK

Closed-form IK (a.k.a. analytic IK) example with 2R planar robot

Given (x, y) , Find $\theta_{1,2}$



- Using the law of cosines, we have

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

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

- Using $\gamma = \text{atan2}(y, x) = \tan^{-1} \frac{y}{x}$ in the range $(-\pi, \pi]$, the righty solution becomes

$$\theta_1 = \gamma - \alpha$$

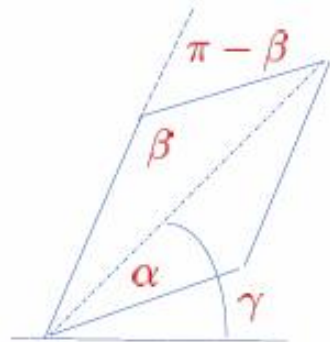
$$\theta_2 = \pi - \beta$$

- The lefty solution is

$$\theta_1 = \gamma + \alpha$$

$$\theta_2 = -\pi + \beta$$

- If $x^2 + y^2$ lies outside the range $[L_1 - L_2, L_1 + L_2]$, then no solution exists.



1. Numerical IK

Numerical IK: Newton-Raphson method

For numerical solutions, we can resort to the Newton-Raphson method. Let the target task pose \mathbf{p}^{des} and current joint variables $\mathbf{q}^{(k)}$. Here, our goal is to find a desired joint variables \mathbf{q}^{des} in next timestep $k + 1$, i.e.

$$\mathbf{p}^{des} = f(\mathbf{q}^{(k+1)}) = f(\mathbf{q}^{(k)}) + \left. \frac{\partial f}{\partial \mathbf{q}^T} \right|_{\mathbf{q}^{(k)}} \cdot (\mathbf{q}^{(k+1)} - \mathbf{q}^{(k)}) + \dots$$

Here, the partial derivative of $f(\mathbf{q})$ is a jacobian matrix $J(\mathbf{q})$, so

$$\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + J^+(\mathbf{q}^{(k)}) \cdot (\mathbf{p}^{des} - f(\mathbf{q}^{(k)}))$$

where J^+ is a **pseudo-inverse** of jacobian (ex. Moore-Penrose, Damped least-square).

The convergence rate of the Newton-Raphson method can be adjusted by multiplying step size κ

$$\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + \kappa J^+(\mathbf{q}^{(k)}) \cdot (\mathbf{p}^{des} - f(\mathbf{q}^{(k)}))$$

1. Numerical IK

Numerical IK: Newton-Raphson method

However, since the target pose is given as 4x4 homogeneous transformation matrix $T \in SE(3)$, the following part causes a problem.

$$\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + \kappa J^+(\mathbf{q}^{(k)}) \cdot \left(\mathbf{p}^{des} - f(\mathbf{q}^{(k)}) \right)$$

Red part is the error between desired pose and current pose, but $SE(3)$ is not closed under the 'matrix summation'. Therefore, we need to define a new representation of error between two transformation matrices.

Recall the jacobian matrix $J_r \in \mathbb{R}^{6 \times n}$ in the lecture.

$$\begin{bmatrix} \dot{x} & \dot{y} & \dot{z} & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T = J_r(\mathbf{q}) \cdot \dot{\mathbf{q}}$$

For a numerical IK, we can use the pose error term as follow:

$$\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + \kappa J_r^+(\mathbf{q}^{(k)}) \cdot \left[(\text{position error})^T \ (\text{orientation error})^T \right]^T$$

where $\mathbf{r}^{err} = \begin{bmatrix} x^{des} - x^{(k)} & y^{des} - y^{(k)} & z^{des} - z^{(k)} \end{bmatrix}^T$ is position error and $\boldsymbol{\theta}^{err} = \begin{bmatrix} \phi^{des} - \phi^{(k)} & \theta^{des} - \theta^{(k)} & \psi^{des} - \psi^{(k)} \end{bmatrix}^T$ is orientation error.

1. Numerical IK

Numerical IK: Newton-Raphson method

Since Euler angle suffer from singularity issue, we introduce the concept of matrix logarithm.

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} 0 & -\sin \phi & \cos \phi \sin \theta \\ 0 & \cos \phi & \sin \phi \sin \theta \\ 1 & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

$\sin \theta = 0 \rightarrow \text{singular}$

Orientational error can be defined as the difference between current orientation and desired orientation as follows:

$$\begin{aligned} R^{err} &= R^{-1} R^{des} \\ &= R^T R^{des} \in SO(3) \end{aligned}$$

Every matrix in $SO(3)$ is bijective to $\xi \in \mathbb{R}^{3 \times 1}$ and the relation is called matrix exponential/logarithm. Python code for matrix exponential and logarithm will be provided.

$$\begin{aligned} R &= \text{expm}(\xi) \\ \xi &= \text{logm}(R) \end{aligned}$$

Using matrix logarithm, numerical IK can be formulated as follows:

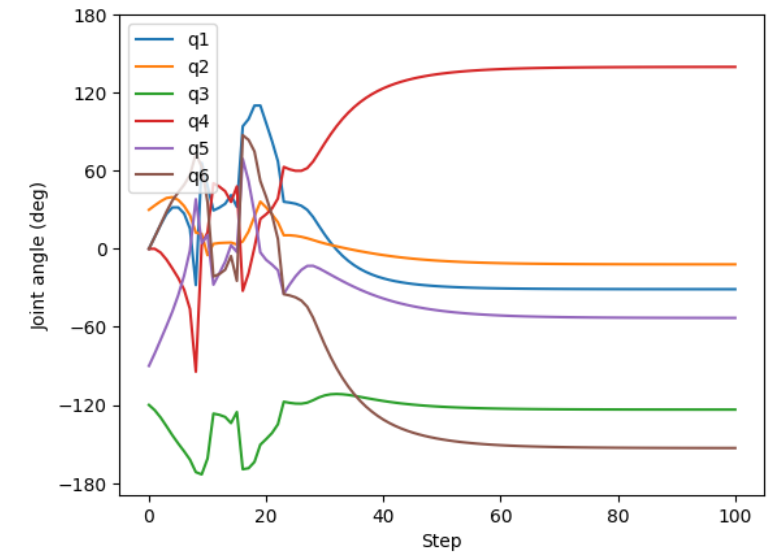
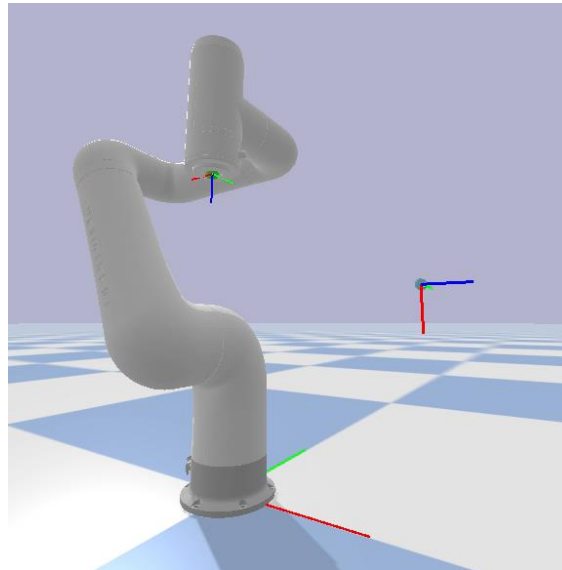
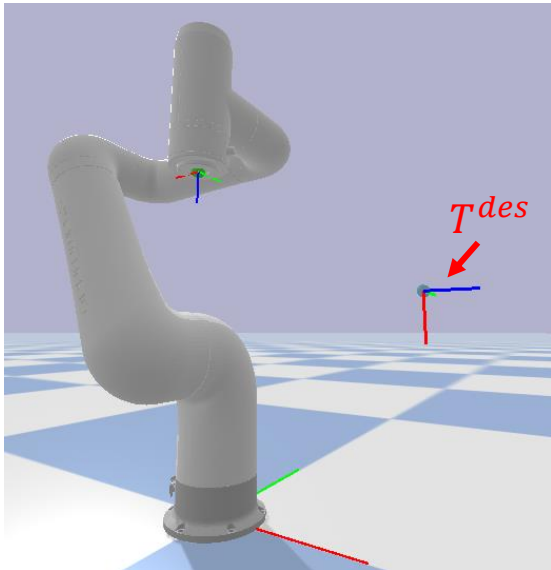
$$\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + \kappa J_v^+(\mathbf{q}^{(k)}) \cdot \begin{bmatrix} \mathbf{r}^{err} \\ \xi^{err} \end{bmatrix}$$

1. Numerical IK

Example

$$\mathbf{q}^{(0)} = [0 \ \pi/6 \ -2\pi/3 \ 0 \ -\pi/2 \ 0]^\top \quad \mathbf{T}^{des} = \begin{bmatrix} 0 & 0 & 1 & 0.4 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{q}^{(k+1)} \leftarrow \mathbf{q}^{(k)} + \kappa \mathbf{J}_v^+(\mathbf{q}^{(k)}) \cdot \begin{bmatrix} \mathbf{r}^{err} \\ \xi^{err} \end{bmatrix} \quad \kappa = 0.1$$



2. Robot Dynamics

Lagrangian approach

The dynamics of the mechanical system can be derived through Lagrangian $L = T - U$.

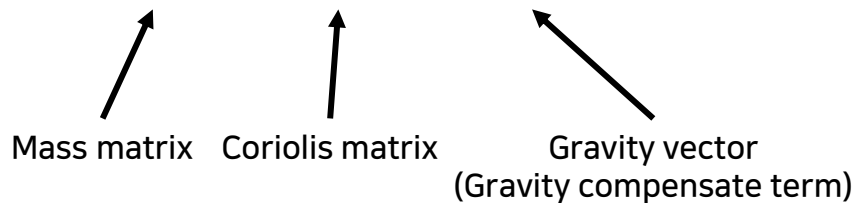
$$L = T - U = \frac{1}{2} \dot{\mathbf{q}}^\top \mathbf{M}(\mathbf{q}) \dot{\mathbf{q}} - U(\mathbf{q})$$

$$\boldsymbol{\tau} = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\mathbf{q}}} \right) - \frac{\partial L}{\partial \mathbf{q}}$$

$$= \frac{d}{dt} (\mathbf{M}(\mathbf{q}) \dot{\mathbf{q}}) - \frac{1}{2} \dot{\mathbf{q}}^\top \frac{\partial \mathbf{M}}{\partial \mathbf{q}} \dot{\mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}}$$

$$= \mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} + \left(\dot{\mathbf{M}}(\mathbf{q}, \dot{\mathbf{q}}) - \frac{1}{2} \dot{\mathbf{q}}^\top \frac{\partial \mathbf{M}}{\partial \mathbf{q}} \right) \dot{\mathbf{q}} + \frac{\partial U}{\partial \mathbf{q}}$$

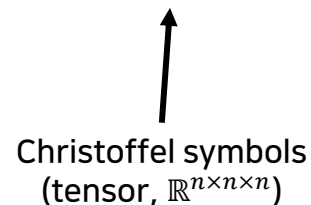
$$= \mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) \dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$$


 Mass matrix Coriolis matrix Gravity vector
 (Gravity compensate term)

$$L = \frac{1}{2} \sum_{j,k} m_{jk} \dot{q}_j \dot{q}_k - U(\mathbf{q})$$

$$\begin{aligned}
 \tau_i &= \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{q}_i} \right) - \frac{\partial L}{\partial q_i} = \frac{d}{dt} \left(\frac{1}{2} \sum_{i,k} m_{ik} \dot{q}_k + \frac{1}{2} \sum_{j,i} m_{ji} \dot{q}_j \right) - \frac{1}{2} \sum_{j,k} \frac{\partial m_{jk}}{\partial q_i} \dot{q}_j \dot{q}_k + \frac{\partial U}{\partial q_i} \\
 &= \sum_j m_{ij} \ddot{q}_j + \sum_{j,k} \frac{\partial m_{ij}}{\partial q_k} \dot{q}_j \dot{q}_k - \frac{1}{2} \sum_{j,k} \frac{\partial m_{jk}}{\partial q_i} \dot{q}_j \dot{q}_k + \frac{\partial U}{\partial q_i} \\
 &= \sum_j m_{ij} \ddot{q}_j + \frac{1}{2} \sum_{j,k} \left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{jk}}{\partial q_i} \right) \dot{q}_j \dot{q}_k + \frac{\partial U}{\partial q_i} \\
 &= \sum_j m_{ij} \ddot{q}_j + \sum_{j,k} \Gamma_{ijk}(\mathbf{q}) \dot{q}_j \dot{q}_k + \frac{\partial U}{\partial q_i}
 \end{aligned}$$

$$\boldsymbol{\tau} = \mathbf{M}(\mathbf{q}) \ddot{\mathbf{q}} + \dot{\mathbf{q}}^\top \boldsymbol{\Gamma}(\mathbf{q}) \dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})$$


 Christoffel symbols
 (tensor, $\mathbb{R}^{n \times n \times n}$)

2. Robot Dynamics

Inverse dynamics & Forward dynamics

Inverse dynamics (for a robot control): $q, \dot{q}, \ddot{q} \rightarrow \tau$

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q)$$

Forward dynamics (for a robot simulation): $q, \dot{q}, \tau \rightarrow \ddot{q}$

$$\ddot{q} = M(q)^{-1}(\tau - C(q, \dot{q})\dot{q} - g(q))$$

In practice, the inverse dynamics is computed recursively using recursive Newton-Euler method (RNE). It is out of the scope of this course, so please refer the [1], [2] if you want to learn more.

[1] Lynch, Kevin M., and Frank C. Park. Modern robotics. Cambridge University Press, 2017.

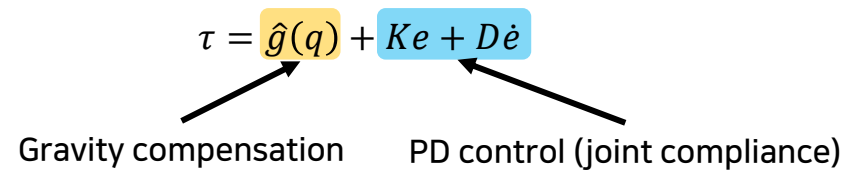
[2] Murray, Richard M., Zexiang Li, and S. Shankar Sastry. A mathematical introduction to robotic manipulation. CRC press, 2017.

3. Robot Control

Simple joint compliance control

The goal of joint compliance control is to get $e = q_{des} - q \rightarrow 0$ when $\dot{\theta}_{des} = 0$ and $\tau_{ext} = 0$.

The control law is very simple.

$$\tau = \hat{g}(q) + Ke + D\dot{e}$$


Gravity compensation PD control (joint compliance)

This is a typical controller in industrial robots. It compensates for gravity and uses simple PD control. This controller, however, works very well, and this controller is exponentially stable when the gravity is perfectly compensated.

(it means the nominal gravity vector \hat{g} is equal to the real gravity vector g ; $\hat{g} = g$).

When gravity compensation is imperfect or absent, assessing stability becomes more complex. The equilibrium is dislocated away from $e = 0$ (which means a nonzero steady state error). We can introduce integral action to reject constant disturbance due to imperfect gravity compensation, so the modified control law is

$$\tau = \hat{g}(q) + Ke + D\dot{e} + I \int e dt$$

4. Practice

Today, you have to do the followings:

1. Implement IK using XYZ position error and ZYZ Euler angle error (Hint: use Jr_i)
2. Implement IK using XYZ position error and $so(3)$ orientation error (Hint: use Jv_i)
3. Draw a circle with radius $R = 0.1$ at center $(x, y, z) = (0.4, 0, 0.4)$, maintaining orientation ZYZ Euler angle $(\phi, \theta, \psi) = (0^\circ, 90^\circ, 0^\circ)$

If you are not done with this during today's class, you can finish it until next Monday!