Kinematics

Hsiu-Chin Lin

September 8, 2020

Welcome

Lecture Recordings This lecture will be recorded. By attending the live sessions, you agree to the recording, and you understand that your image, voice, and name may be disclosed to classmates

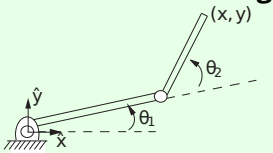
During the lecture
I will mute the audience
Ask questions through 'Chat'

Outline

Outline

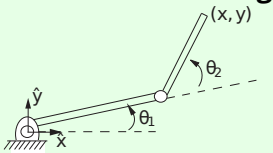
Robots

A robot is a set of **rigid bodies** connected by a set of **joints**



Robots

A robot is a set of **rigid bodies** connected by a set of **joints**



Rigid-body

A rigid-body is a solid object in which deformation can be neglected

Configuration Space

Configuration $\mathbf{q}$
a set of quantities that provides a complete specification of the position of every point of the robot

Degree-of-freedom $\mathcal{N}$
the smallest number that is needed to represent the configuration space

(a.k.a. generalized coordinates, joint-space, state space)

Configuration Space

Configuration $\mathbf{q}$
a set of quantities that provides a complete specification of the position of every point of the robot

Degree-of-freedom $\mathcal{N}$
the smallest number that is needed to represent the configuration space

The first thing to describe your robot

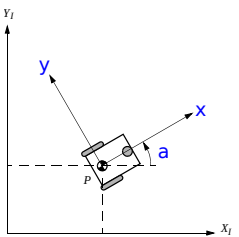
$$\mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_{\mathcal{N}} \end{bmatrix}$$

Configuration Space

Configuration q
a set of quantities that provides a complete specification of the position of every point of the robot

Degree-of-freedom $\mathcal{N}$
the smallest number that is needed to represent the configuration space

Example (Mobile Robots)



Origin: P
Configuration: x-y position, orientation

$$\mathbf{q} = \begin{bmatrix} x \\ y \\ a \end{bmatrix}$$

Configuration Space

Configuration q
a set of quantities that provides a complete specification of the position of every point of the robot

Degree-of-freedom $\mathcal{N}$
the smallest number that is needed to represent the configuration space

Example (Quadrotor)

Origin: P
Configuration:
x, y, z, roll, pitch, yaw

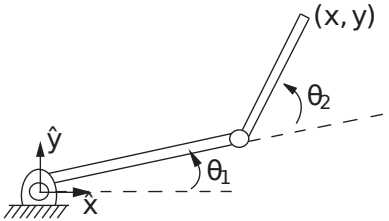
$$\mathbf{q} = \begin{bmatrix} x \\ y \\ a \\ \rho \\ \theta \\ \psi \end{bmatrix}$$

Configuration Space

Configuration q
a set of quantities that provides a complete specification of the position of every point of the robot

Degree-of-freedom $\mathcal{N}$
the smallest number that is needed to represent the configuration space

Example (Planar Robot Arm)



Origin: **0**
Configuration: joint positions

$$\mathbf{q} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

Task Space

Task space

space in which the robot's task can be naturally expressed
may not be the same with the configuration space

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{\mathcal{P}} \end{bmatrix}$$

$\mathcal{P}$ is the dimensionality of the task

a robot can have more than one task space

Task space

space in which the robot's task can be naturally expressed
may not be the same with the configuration space

Example (Manipulation)

move hand from position A to position B

- Configuration space
  - 7 joint position $\mathbf{q} \in \mathbb{R}^7$
- Task space
  - end-effector (hand) position $\mathbf{x} \in \mathbb{R}^3$

Task space

space in which the robot's task can be naturally expressed
may not be the same with the configuration space

Example (Quadruped Locomotion)

walking from point A to point B

- Configuration space
  - Base position and orientation
  - 12 Joint positions $\mathbf{q} \in \mathbb{R}^{18}$
- Task Space
  - Body position $\mathbf{x}_{body} \in \mathbb{R}^3$
  - Feet position $\mathbf{x}_{feet} \in \mathbb{R}^3$

Positions/Translations

Displacement from a reference point

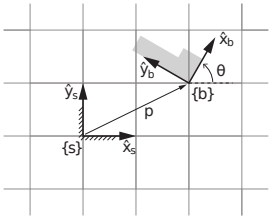
$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix} \in \mathbb{R}^2 \qquad \mathbf{p} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ \mathbf{p}_z \end{bmatrix} \in \mathbb{R}^3$$

Positions/Translations

Displacement from a reference point

$$\mathbf{p} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{bmatrix} \in \mathbb{R}^2 \qquad \mathbf{p} = \begin{bmatrix} \mathbf{p}_x \\ \mathbf{p}_y \\ \mathbf{p}_z \end{bmatrix} \in \mathbb{R}^3$$

Example (2D displacement)



Reference frame {s}
Robot frame {b}

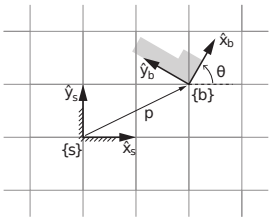
Orientations

- **angles**: rotation with respect to a fixed coordinate system
- **unit quaternions**: alternative representation in 3D
- **rotation matrices**: matrices that rotate vectors

Orientations

- **angles**: rotation with respect to a fixed coordinate system
- **unit quaternions**: alternative representation in 3D
- **rotation matrices**: matrices that rotate vectors

Example (Orientations in 2D)



Angle: θ
(w.r.t. x-axis)

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Orientations

- **angles**: rotation with respect to a fixed coordinate system
- **unit quaternions**: alternative representation in 3D
- **rotation matrices**: matrices that rotate vectors

Rotation Matrices

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \in \mathbb{SO}(2) \qquad \mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in \mathbb{SO}(3)$$

$\mathbb{SO}(2)$ and $\mathbb{SO}(3)$: **special orthogonal** group in $\mathbb{R}^2$ and $\mathbb{R}^3$

Homogeneous Transformation Matrix
Represent orientation and position of a rigid body

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{SE}(.) : \text{ special euclidean group}$$

Homogeneous Transformation Matrix
Represent orientation and position of a rigid body

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{SE}(.) : \text{ special euclidean group}$$

Representations in $\mathbb{R}^2$ and $\mathbb{R}^3$

$$\mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & p_x \\ r_{21} & r_{22} & p_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}(2) \qquad \mathbf{T} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{SE}(3)$$

Properties of Homogeneous Transformation

Question: When do I use it?

- to represent the position and orientation of a rigid body
- to change the reference frame
- to displace a vector or frame

Properties of Homogeneous Transformation

The inverse of a transformation matrix is a transformation matrix

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$$

Properties of Homogeneous Transformation

The inverse of a transformation matrix is a transformation matrix

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^{\top} & -\mathbf{R}^{\top}\mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Properties of Homogeneous Transformation

The inverse of a transformation matrix is a transformation matrix

$$\mathbf{T}^{-1} = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{R}^\top & -\mathbf{R}^\top \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix}$$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1 \mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1 \mathbf{R}_2 & \mathbf{R}_1 \mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

The multiplication of transformation matrices is associative

$$(\mathbf{T}_1 \mathbf{T}_2)\mathbf{T}_3 = \mathbf{T}_1(\mathbf{T}_2 \mathbf{T}_3)$$

Outline

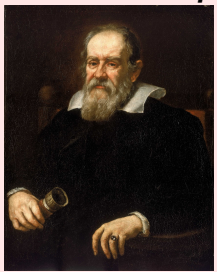
Kinematics is a geometric description of motion

- Positions, velocities, accelerations
- Visually observable

Kinematics is a geometric description of motion

- Positions, velocities, accelerations
- Visually observable

Galileo's Experiments on the Tower of Pisa



Question: Given current configurations, what is the task space?
Configurations provide a complete specification of the position of every point of the robot. How do we compute it.

Question: Given current configurations, what is the task space?
Configurations provide a complete specification of the position of every point of the robot. How do we compute it.

Forward Kinematics
Relates the configuration space to the task space

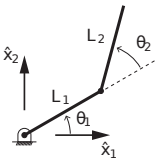
Forward Kinematics at Position Level

$$\mathbf{x} = \mathcal{F}(\mathbf{q})$$

Forward Kinematics at Position Level

$$\mathbf{x} = \mathcal{F}(\mathbf{q})$$

Example (2D planar arm)



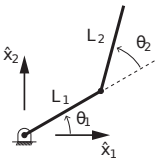
configuration $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

task $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$

Forward Kinematics at Position Level

$$\mathbf{x} = \mathcal{F}(\mathbf{q})$$

Example (2D planar arm)



$$\text{configuration } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\text{task } \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$$

$$\mathbf{x}_1 = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$

$$\mathbf{x}_2 = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2)$$

Forward Kinematics using Homogenous Transformation Matrix

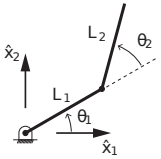
The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Example (2D planar arm)



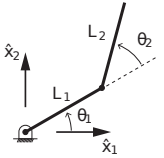
configuration $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

task $\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Example (2D planar arm)



$$\text{configuration } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

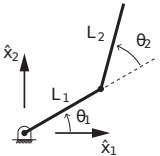
$$\text{task } \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$$

$$\mathbf{T}_{01} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Example (2D planar arm)



$$\text{configuration } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

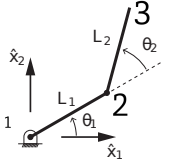
$$\text{task } \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$$

$$\mathbf{T}_{01} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{12} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & L_1 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Example (2D planar arm)



configuration $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

task $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

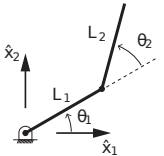
points?

$$\mathbf{T}_{01} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{12} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & L_1 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{23} = \begin{bmatrix} 1 & 0 & L_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The product of two transformation matrices is a transformation matrix

$$\mathbf{T}_1\mathbf{T}_2 = \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2 & \mathbf{R}_1\mathbf{p}_2 + \mathbf{p}_1 \\ \mathbf{0} & 1 \end{bmatrix}$$

Example (2D planar arm)



$$\text{configuration } \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\text{task } \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}$$

$$\mathbf{T}_{01} = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{12} = \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & L_1 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{T}_{23} = \begin{bmatrix} 1 & 0 & L_2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{T}_{03} = \mathbf{T}_{01}\mathbf{T}_{12}\mathbf{T}_{23}$$

Question: What about velocity?
Velocity is the time derivative of position

Forward Kinematics

$$\frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial t}$$ (time derivative of forward kineamtics)

Forward Kinematics

$$\frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial t}$$ (time derivative of forward kineamtics)

$$\dot{\mathbf{x}} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t}$$ (chain rule)

Forward Kinematics

$$\frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial t} \qquad \text{(time derivative of forward kineamtics)}$$

$$\dot{\mathbf{x}} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} \qquad \text{(chain rule)}$$

$$= \begin{bmatrix} \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_\mathcal{N}} \\ \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_\mathcal{N}} \\ \vdots & & & \\ \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_\mathcal{N}} \end{bmatrix} \dot{\mathbf{q}}$$

(partial derivative of a vector over a vector gives you a matrix)

Forward Kinematics

$$\frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial t} \qquad \text{(time derivative of forward kineamtics)}$$

$$\dot{\mathbf{x}} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} \qquad \text{(chain rule)}$$

$$= \begin{bmatrix} \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_\mathcal{N}} \\ \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_\mathcal{N}} \\ \vdots & & & \\ \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_\mathcal{N}} \end{bmatrix} \dot{\mathbf{q}}$$

(partial derivative of a vector over a vector gives you a matrix)

$$= \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$$\frac{\partial \mathbf{x}}{\partial t} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial t} \qquad \text{(time derivative of forward kineamtics)}$$

$$\dot{\mathbf{x}} = \frac{\partial \mathcal{F}(\mathbf{q})}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} \qquad \text{(chain rule)}$$

$$= \begin{bmatrix} \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_1(\mathbf{q})}{\partial q_\mathcal{N}} \\ \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_2(\mathbf{q})}{\partial q_\mathcal{N}} \\ \vdots & & & \\ \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_1} & \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_2} & \cdots & \frac{\partial \mathcal{F}_\mathcal{P}(\mathbf{q})}{\partial q_\mathcal{N}} \end{bmatrix} \dot{\mathbf{q}}$$

(partial derivative of a vector over a vector gives you a matrix)

$$= \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

where $\mathbf{J} \in \mathbb{R}^{\mathcal{P} \times \mathcal{N}}$ is called the **jacobian** matrix

Forward Kinematics at Velocity Level

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$\mathbf{J} \in \mathbb{R}^{\mathcal{P}\times\mathcal{N}}$ is called the **jacobian** matrix

Forward Kinematics at Velocity Level

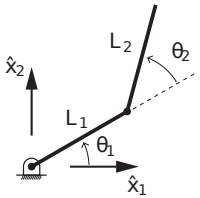
$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

$\mathbf{J} \in \mathbb{R}^{\mathcal{P}\times\mathcal{N}}$ is called the **jacobian** matrix

Let $\mathbf{J}_i$ be the i^{th} row of $\mathbf{J}$.

$$\dot{\mathbf{x}}_1 = \mathbf{J}_1\dot{\boldsymbol{\theta}}$$
$$\dot{\mathbf{x}}_2 = \mathbf{J}_2\dot{\boldsymbol{\theta}}$$
$$\vdots$$
$$\dot{\mathbf{x}}_{\mathcal{P}} = \mathbf{J}_{\mathcal{P}}\dot{\boldsymbol{\theta}}$$

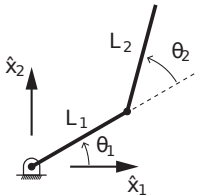
Example (2D planar arm)



configuration $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

task $\mathbf{x} = \begin{bmatrix} \mathsf{x}_1 \\ \mathsf{x}_2 \end{bmatrix}$

Example (2D planar arm)



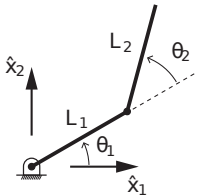
configuration $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

task $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Take time-derivative on both sides

$$\dot{x}_1 = -L_1\dot{\theta}_1 \sin(\theta_1) - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$$
$$\dot{x}_2 = L_1\dot{\theta}_1 \cos(\theta_1) + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2)$$

Example (2D planar arm)



configuration $\boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

task $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Take time-derivative on both sides

$$\dot{x}_1 = -L_1\dot{\theta}_1 \sin(\theta_1) - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$$
$$\dot{x}_2 = L_1\dot{\theta}_1 \cos(\theta_1) + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2)$$

Rearrange into the form of $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -L_1 \sin(\theta_1) - L_2 \sin(\theta_1 + \theta_2), & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2), & L_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix}$$

Forward Kinematics at Acceleration Level

$$\frac{\partial \dot{\mathbf{x}}}{\partial t} = \frac{\partial \left[\mathbf{J}\dot{\mathbf{q}}\right]}{\partial t}$$

Forward Kinematics at Acceleration Level

$$\frac{\partial \dot{\mathbf{x}}}{\partial t} = \frac{\partial \left[\mathbf{J}\dot{\mathbf{q}}\right]}{\partial t}$$

$$\ddot{\mathbf{x}} = \mathbf{J}\frac{\partial \dot{\mathbf{q}}}{\partial t} + \frac{\partial \mathbf{J}}{\partial t}\dot{\mathbf{q}}$$

Forward Kinematics at Acceleration Level

$$\begin{aligned}\frac{\partial \dot{\mathbf{x}}}{\partial t} &= \frac{\partial \left[\mathbf{J}\dot{\mathbf{q}}\right]}{\partial t} \\ \ddot{\mathbf{x}} &= \mathbf{J}\frac{\partial \dot{\mathbf{q}}}{\partial t} + \frac{\partial \mathbf{J}}{\partial t}\dot{\mathbf{q}} \\ &= \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}\end{aligned}$$

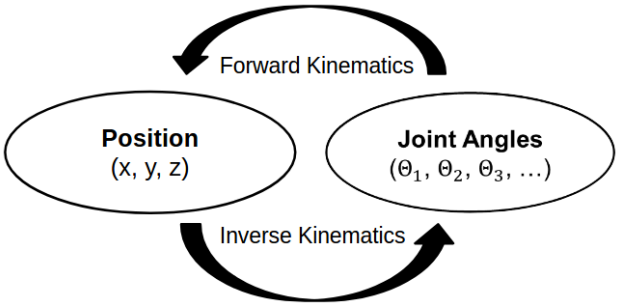
Forward Kinematics

Question: Too much math?

Question: Too much math?
most robotics library offers forward kinematics and Jacobian

Outline

Inverse Kinematics



Question: Given the task-space velocities, what are the configuration space velocities?

Question: Given the task-space velocities, what are the configuration space velocities?

Normally, it is easiler to plan or control at the task-space rather than the configuration space. If we know the task-space velocities, we need to convert it into configuration velocities

Question: Given the task-space velocities, what are the configuration space velocities?

Normally, it is easiler to plan or control at the task-space rather than the configuration space. If we know the task-space velocities, we need to convert it into configuration velocities

Question: We know $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$, can we do $\dot{\mathbf{q}} = \mathbf{J}^{-1}\dot{\mathbf{x}}$?
No! $\mathbf{J} \in \mathbb{R}^{\mathcal{P} \times \mathcal{N}}$ is rank deficient, and the inverse does not exist

Inverse Kinemaitcs

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}} \quad \text{(forward kinematics)}$$

$$\Downarrow$$

$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{x}}$ where $\mathbf{J}^{\dagger}$ is the **pseudo inverse** of $\mathbf{J}$

Inverse Kinemaitcs

$$\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}} \quad \text{(forward kinematics)}$$

$$\Downarrow$$

$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{x}}$ where $\mathbf{J}^{\dagger}$ is the **pseudo inverse** of $\mathbf{J}$

Pseudo-inverse has properties that are similar to inverse

$$\begin{cases} \mathbf{J}^{\dagger} \text{ exists for all } \mathbf{J} \\ \mathbf{J}\mathbf{J}^{\dagger} = \mathbf{I} \\ \mathbf{J}\mathbf{J}^{\dagger}\mathbf{J} = \mathbf{J} \end{cases}$$

Question:
How to calculate pseudo inverse?

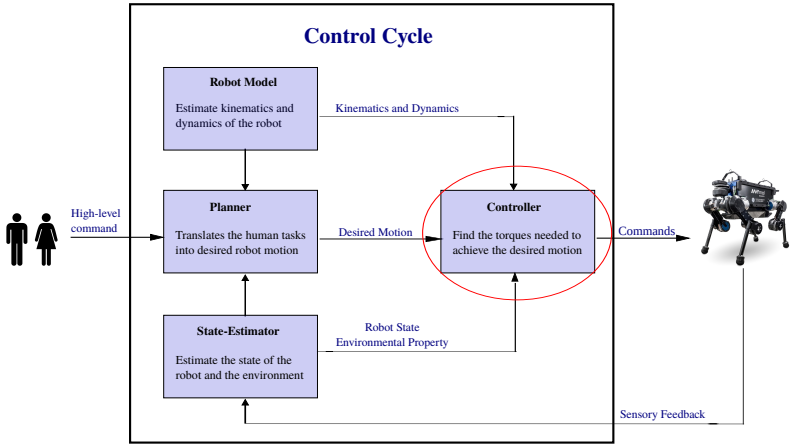
Question:
How to calculate pseudo inverse?

- most tools (matlab, python, c++) have a function to calculate pseudo inverse

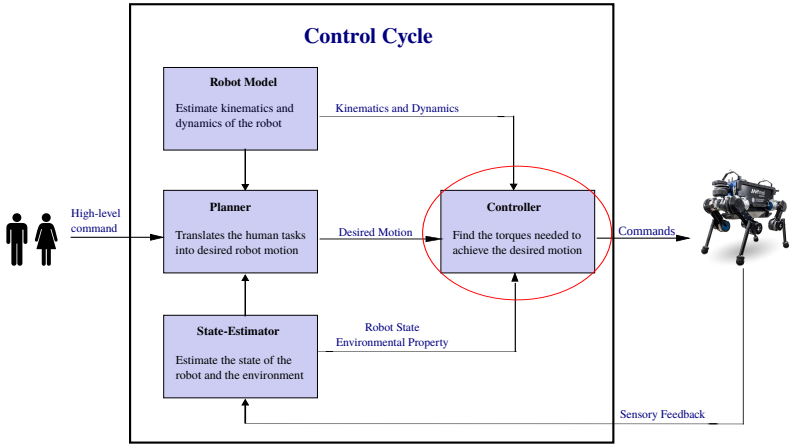
Question:
How to calculate pseudo inverse?

- most tools (matlab, python, c++) have a function to calculate pseudo inverse
- If $\mathbf{J}$ is full row rank, $\mathbf{J}^{\dagger} = \mathbf{J}(\mathbf{J}^{\top}\mathbf{J})^{-1}$ has an analytical solution

Controllers



Controllers



Commands

- positions
- velocities
- torques

Inverse Kinematics Controller

Desired position $\mathbf{x}^* \in \mathbb{R}^3$

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}_x^* \\ \mathbf{x}_y^* \\ \mathbf{x}_z^* \end{bmatrix}$$

provided by the planner

Inverse Kinematics Controller

Desired position $\mathbf{x}^* \in \mathbb{R}^3$

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}_x^* \\ \mathbf{x}_y^* \\ \mathbf{x}_z^* \end{bmatrix}$$

provided by the planner

current position $\mathbf{x} \in \mathbb{R}^3$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_x \\ \mathbf{x}_y \\ \mathbf{x}_z \end{bmatrix}$$

calculated by forward kinematics

Inverse Kinematics Controller

Desired position $\mathbf{x}^* \in \mathbb{R}^3$

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{x}_x^* \\ \mathbf{x}_y^* \\ \mathbf{x}_z^* \end{bmatrix}$$

provided by the planner

current position $\mathbf{x} \in \mathbb{R}^3$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_x \\ \mathbf{x}_y \\ \mathbf{x}_z \end{bmatrix}$$

calculated by forward kinematics

Question: How to move from $\mathbf{x}$ to $\mathbf{x}^*$
The desired task space velocity: $\dot{\mathbf{x}} = \mathbf{x}^* - \mathbf{x}$
The desired configuration space velocity: $\dot{\mathbf{q}} = \mathbf{J}^\dagger \dot{\mathbf{x}}$

Matlab Example

Example (Kinova Jaco arm)

Move hand from position A to position B
$\mathbf{q} \in \mathbb{R}^7$: joints
$\mathbf{x} \in \mathbb{R}^3$: hand position
$\mathbf{J} \in \mathbb{R}^{3\times 7}$: Jacobian

Blue: initial
Red: target
Green: trajectory

Outline

What is Redundancy?

[0] © Ministry of Silly Walks

What is Redundancy?

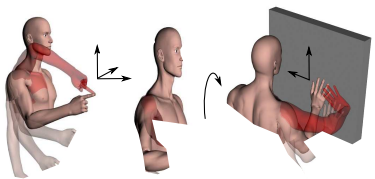
Redundancy
More degree-of-freedom than what is required for your task
There are more than one solutions to achieve the desired task

What is Redundancy?

Redundancy
More degree-of-freedom than what is required for your task
There are more than one solutions to achieve the desired task

Example

- keep your finger tip and the same position and move your elbows



- joint-space velocity but no task-space velocity

Redundancy

Mathematically....

Redundancy

Mathematically....

- There are many solutions to $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$

Redundancy

Mathematically....

- There are many solutions to $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$
- Jacobian is rank deficient $\mathrm{r}(\mathbf{J}) \leq \mathcal{P} < \mathcal{N}$

Redundancy

Mathematically....

- There are many solutions to $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$
- Jacobian is rank deficient $\mathrm{r}(\mathbf{J}) \leq \mathcal{P} < \mathcal{N}$
- There exist a nullspace $\mathcal{N}(\mathbf{J}) \neq 0$

Redundancy

Mathematically....

- There are many solutions to $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$
- Jacobian is rank deficient $\mathrm{r}(\mathbf{J}) \leq \mathcal{P} < \mathcal{N}$
- There exist a nullspace $\mathcal{N}(\mathbf{J}) \neq 0$

Nullspace
The nullspace of a matrix $\mathbf{A}$ consists of all the vectors $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0}$

Redundancy

Mathematically....

- There are many solutions to $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$
- Jacobian is rank deficient $\mathrm{r}(\mathbf{J}) \leq \mathcal{P} < \mathcal{N}$
- There exist a nullspace $\mathcal{N}(\mathbf{J}) \neq 0$

Nullspace
The nullspace of a matrix $\mathbf{A}$ consists of all the vectors $\mathbf{x}$ such that $\mathbf{A}\mathbf{x} = \mathbf{0}$

so... $\mathbf{J}\dot{\mathbf{q}}^0 = \mathbf{0}$ for some $\dot{\mathbf{q}}^0$

Inverse Kinemaitcs Controller with Redudnacy Resolution

$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{x} + \mathbf{N}\dot{\mathbf{q}}^{0}$ where

$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is the nullspace projection matrix

Inverse Kinemaitcs Controller with Redudnacy Resolution

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{x} + \mathbf{N}\dot{\mathbf{q}}^{0} \text{ where}$$

$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is the nullspace projection matrix

Projection Matrix

A projection matrix $\mathbf{N}$ projects a vector onto its subspace

Inverse Kinemaitcs Controller with Redudnacy Resolution

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{x} + \mathbf{N}\dot{\mathbf{q}}^{0} \text{ where}$$

$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is the nullspace projection matrix

Projection Matrix
A projection matrix $\mathbf{N}$ projects a vector onto its subspace

- $\mathbf{N}$ projects $\dot{\mathbf{q}}^{0}$ onto the nullspace of the jacobian.

Inverse Kinemaitcs Controller with Redudnacy Resolution

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{x} + \mathbf{N}\dot{\mathbf{q}}^{0} \text{ where}$$

$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is the nullspace projection matrix

Projection Matrix
A projection matrix $\mathbf{N}$ projects a vector onto its subspace

- $\mathbf{N}$ projects $\dot{\mathbf{q}}^{0}$ onto the nullspace of the jacobian.
- $\dot{\mathbf{q}}^{0}$ can be ANY arbitrary vector. $\mathbf{N}\dot{\mathbf{q}}^{0}$ has **NO** effect on the $\dot{\mathbf{x}}$

Inverse Kinemaitcs Controller with Redudnacy Resolution

$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{x} + \mathbf{N}\dot{\mathbf{q}}^{0}$ where

$\mathbf{N} = \mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}$ is the nullspace projection matrix

Projection Matrix

A projection matrix $\mathbf{N}$ projects a vector onto its subspace

- $\mathbf{N}$ projects $\dot{\mathbf{q}}^{0}$ onto the nullspace of the jacobian.
- $\dot{\mathbf{q}}^{0}$ can be ANY arbitrary vector. $\mathbf{N}\dot{\mathbf{q}}^{0}$ has **NO** effect on the $\dot{\mathbf{x}}$
  $\mathbf{J}\mathbf{N}\dot{\mathbf{q}}^{0} = \mathbf{J}(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J})\dot{\mathbf{q}}^{0} = (\mathbf{J} - \mathbf{J}\mathbf{J}^{\dagger}\mathbf{J})\dot{\mathbf{q}}^{0} = (\mathbf{J} - \mathbf{J})\dot{\mathbf{q}}^{0} = \mathbf{0}$

Redundancy

Ideally, we like to move our arms in a comfortable way, or reduce the amount of energy we consume for a given task.

Redundancy

Ideally, we like to move our arms in a comfortable way, or reduce the amount of energy we consume for a given task.

Move the redundant dimension to some default positions

$$\dot{\mathbf{q}}^0 = \mathbf{q}^0 - \mathbf{q}$$

$\mathbf{q}^0$ is the home configuration

$\mathbf{q}$ is the current configuration

Matlab Example

Without Redundancy Resolution

With Redundancy Resolution