CTU

CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

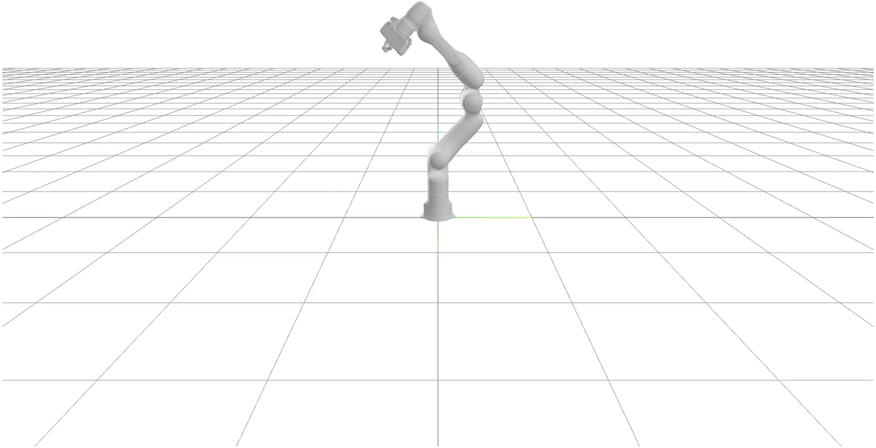
Robotics: Differential Kinematics and Statics

Vladimír Petrík

vladimir.petrik@cvut.cz

20.10.2023

Motivation



Differential kinematics

- We know how to compute end-effector pose from the configuration
  - forward kinematics
  - $\boldsymbol{x}(t) = f_{\mathsf{fk}}(\boldsymbol{q}(t))$
  - $\boldsymbol{x}(t)$ is expressed in task-space, *i.e.* $SE(2)$, $SE(3)$, or $\mathbb{R}^2$, $\mathbb{R}^3$ for position only
  - $\boldsymbol{q}(t) \in \mathbb{R}^N$ is configuration (joint space)
  - t represents time

Differential kinematics

- We know how to compute end-effector pose from the configuration
  - forward kinematics
  - $\boldsymbol{x}(t) = f_{\mathsf{fk}}(\boldsymbol{q}(t))$
  - $\boldsymbol{x}(t)$ is expressed in task-space, *i.e.* $SE(2)$, $SE(3)$, or $\mathbb{R}^2$, $\mathbb{R}^3$ for position only
  - $\boldsymbol{q}(t) \in \mathbb{R}^N$ is configuration (joint space)
  - t represents time
- Differential kinematics
  - relates end-effector velocity to joint velocities
  - $\dot{\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} \in \mathbb{R}^M$
  - Jacobian of the manipulator is core structure in the analysis

Jacobian

Forward kinematics:

$$\boldsymbol{x}(t) = f_{\mathsf{fk}}(\boldsymbol{q}(t))$$

Jacobian:

$$\dot{\boldsymbol{x}} = \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t}$$

Jacobian

Forward kinematics:

$$\boldsymbol{x}(t) = f_{\mathsf{fk}}(\boldsymbol{q}(t))$$

Jacobian:

$$\begin{aligned}\dot{\boldsymbol{x}} &= \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} \\ &= \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}(t)}{\mathrm{d}t}\end{aligned}$$

Jacobian

Forward kinematics:

$$\boldsymbol{x}(t) = f_{\mathsf{fk}}(\boldsymbol{q}(t))$$

Jacobian:

$$\begin{aligned}\dot{\boldsymbol{x}} &= \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} \\ &= \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}(t)}{\mathrm{d}t} \\ &= \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}\end{aligned}$$

Jacobian

Forward kinematics:

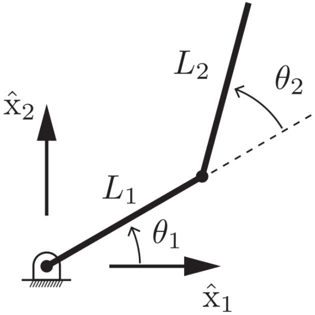
$$\boldsymbol{x}(t) = f_{\mathsf{fk}}(\boldsymbol{q}(t))$$

Jacobian:

$$\begin{aligned}
\dot{\boldsymbol{x}} &= \frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} \\
&= \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \frac{\mathrm{d}\boldsymbol{q}(t)}{\mathrm{d}t} \\
&= \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} \\
&= J(\boldsymbol{q})\dot{\boldsymbol{q}}
\end{aligned}$$

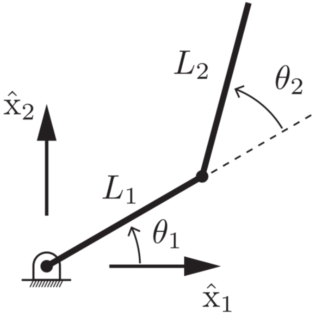
$$J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$$

Planar robot example



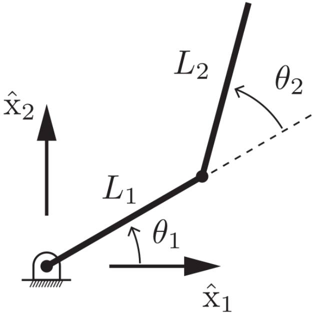
- FK: $\boldsymbol{q} = (\theta_1, \theta_2)^\top \to (x, y)^\top$

Planar robot example



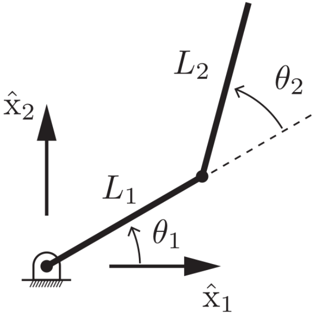
- FK: $\boldsymbol{q} = (\theta_1, \theta_2)^\top \rightarrow (x, y)^\top$
  - $x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$
  - $y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$
- $\dot{\boldsymbol{x}} = ?$

Planar robot example



- FK: $\boldsymbol{q} = (\theta_1, \theta_2)^\top \rightarrow (x, y)^\top$
  - $x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$
  - $y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$
- $\dot{\boldsymbol{x}} = ?$
  - $\dot{x}_1 = -L_1\dot{\theta}_1 \sin\theta_1 - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$
  - $\dot{y}_1 = L_1\dot{\theta}_1 \cos\theta_1 + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2)$

Planar robot example



- FK: $\boldsymbol{q} = (\theta_1, \theta_2)^\top \to (x, y)^\top$
  - $x = L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2)$
  - $y = L_1 \sin\theta_1 + L_2 \sin(\theta_1 + \theta_2)$
- $\dot{\boldsymbol{x}} = ?$
  - $\dot{x}_1 = -L_1\dot{\theta}_1 \sin\theta_1 - L_2(\dot{\theta}_1 + \dot{\theta}_2)\sin(\theta_1 + \theta_2)$
  - $\dot{y}_1 = L_1\dot{\theta}_1 \cos\theta_1 + L_2(\dot{\theta}_1 + \dot{\theta}_2)\cos(\theta_1 + \theta_2)$
  - $J(\boldsymbol{q}) = \begin{pmatrix} -L_1 \sin\theta_1 - L_2 \sin(\theta_1 + \theta_2) & -L_2 \sin(\theta_1 + \theta_2) \\ L_1 \cos\theta_1 + L_2 \cos(\theta_1 + \theta_2) & L_2 \cos(\theta_1 + \theta_2) \end{pmatrix}$
  - Jacobian depends on the configuration $\boldsymbol{q}$

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$
- 2 DoF robot with translation task space:

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$
- 2 DoF robot with translation task space: 2×2
- 2 DoF robot with $SE(2)$ task space:

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$
- 2 DoF robot with translation task space: 2×2
- 2 DoF robot with $SE(2)$ task space: 3×2
- 5 DoF robot with $SE(2)$ task space:

Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$
- 2 DoF robot with translation task space: 2×2
- 2 DoF robot with $SE(2)$ task space: 3×2
- 5 DoF robot with $SE(2)$ task space: 3×5
- 6 DoF robot with $SE(3)$ task space:

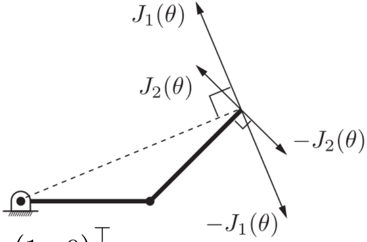
Jacobian dimension

- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$
- 2 DoF robot with translation task space: 2×2
- 2 DoF robot with $SE(2)$ task space: 3×2
- 5 DoF robot with $SE(2)$ task space: 3×5
- 6 DoF robot with $SE(3)$ task space: 6×6
- 7 DoF robot with $SE(3)$ task space:

Jacobian dimension

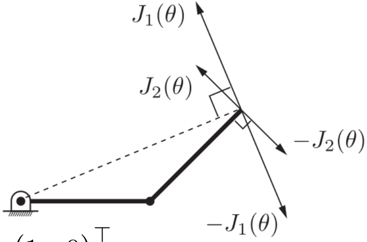
- $J(\boldsymbol{q}) = \frac{\partial f_{\mathsf{fk}}(\boldsymbol{q})}{\partial \boldsymbol{q}} \in \mathbb{R}^{M \times N}$
- M task-space DoF
- N joint-space DoF
- Redundant robots: $N > M$
- Under-actuated robots: $N < M$
- 2 DoF robot with translation task space: 2×2
- 2 DoF robot with $SE(2)$ task space: 3×2
- 5 DoF robot with $SE(2)$ task space: 3×5
- 6 DoF robot with $SE(3)$ task space: 6×6
- 7 DoF robot with $SE(3)$ task space: 6×7

Jacobian properties



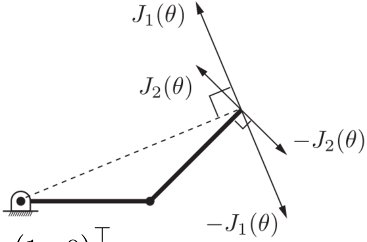
- $J(\boldsymbol{q}) = \begin{pmatrix} J_1(\boldsymbol{q}) & J_2(\boldsymbol{q}) \end{pmatrix}$
- First column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top$
- Second column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top$
- $\dot{\boldsymbol{x}} = \boldsymbol{v}_{\text{tip}} = J_1(\boldsymbol{q})\dot{\theta}_1 + J_2(\boldsymbol{q})\dot{\theta}_2$

Jacobian properties



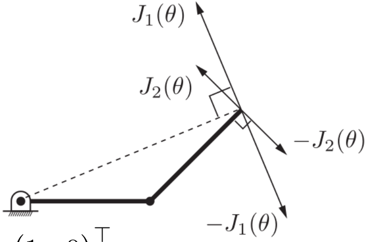
- $J(\boldsymbol{q}) = \begin{pmatrix} J_1(\boldsymbol{q}) & J_2(\boldsymbol{q}) \end{pmatrix}$
- First column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top$
- Second column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top$
- $\dot{\boldsymbol{x}} = \boldsymbol{v}_{\text{tip}} = J_1(\boldsymbol{q})\dot{\theta}_1 + J_2(\boldsymbol{q})\dot{\theta}_2$
- We can generate tip velocity in any direction if $J_1(\boldsymbol{q})$ and $J_2(\boldsymbol{q})$ are not collinear
  - when they are collinear?

Jacobian properties



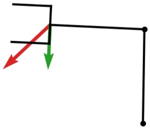
- $J(\boldsymbol{q}) = \begin{pmatrix} J_1(\boldsymbol{q}) & J_2(\boldsymbol{q}) \end{pmatrix}$
- First column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top$
- Second column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top$
- $\dot{\boldsymbol{x}} = \boldsymbol{v}_{\text{tip}} = J_1(\boldsymbol{q})\dot{\theta}_1 + J_2(\boldsymbol{q})\dot{\theta}_2$
- We can generate tip velocity in any direction if $J_1(\boldsymbol{q})$ and $J_2(\boldsymbol{q})$ are not collinear
  - when they are collinear?*e.g.* $\theta_2 = 0$

Jacobian properties



- $J(\boldsymbol{q}) = \begin{pmatrix} J_1(\boldsymbol{q}) & J_2(\boldsymbol{q}) \end{pmatrix}$
- First column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 1 & 0 \end{pmatrix}^\top$
- Second column corresponds to the end-point velocity for $\dot{\boldsymbol{q}} = \begin{pmatrix} 0 & 1 \end{pmatrix}^\top$
- $\dot{\boldsymbol{x}} = \boldsymbol{v}_{\text{tip}} = J_1(\boldsymbol{q})\dot{\theta}_1 + J_2(\boldsymbol{q})\dot{\theta}_2$
- We can generate tip velocity in any direction if $J_1(\boldsymbol{q})$ and $J_2(\boldsymbol{q})$ are not collinear
  - when they are collinear? *e.g.* $\theta_2 = 0$
  - Jacobian is singular matrix $\rightarrow$ configurations are called **singularities**
  - rank of Jacobian is not maximal
  - end-effector is unable to generate velocity in a certain direction

Jacobian columns visualization



How to compute jacobian numerically

- Finite difference method
  - $f'(x_0) \approx \frac{f(x_0+\delta)-f(x_0)}{\delta}, \quad \delta \to 0$

How to compute jacobian numerically

- Finite difference method
  - $f'(x_0) \approx \frac{f(x_0+\delta)-f(x_0)}{\delta}, \quad \delta \to 0$
- $J = \begin{pmatrix} \frac{\partial x}{\partial q_0} & \frac{\partial x}{\partial q_1} & \dots \\ \frac{\partial y}{\partial q_0} & \frac{\partial y}{\partial q_1} & \dots \\ \frac{\partial \theta}{\partial q_0} & \frac{\partial \theta}{\partial q_1} & \dots \end{pmatrix}$

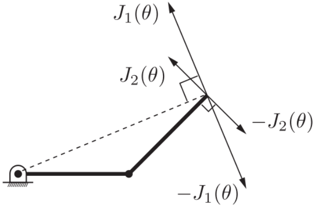
How to compute jacobian numerically

- Finite difference method
  - $f'(x_0) \approx \frac{f(x_0+\delta)-f(x_0)}{\delta}, \quad \delta \to 0$
- $J = \begin{pmatrix} \frac{\partial x}{\partial q_0} & \frac{\partial x}{\partial q_1} & \cdots \\ \frac{\partial y}{\partial q_0} & \frac{\partial y}{\partial q_1} & \cdots \\ \frac{\partial \theta}{\partial q_0} & \frac{\partial \theta}{\partial q_1} & \cdots \end{pmatrix}$
- $\frac{\partial x}{\partial q_0}(\boldsymbol{q}) \approx \frac{f_{\mathsf{fk,x}}(\boldsymbol{q}+\boldsymbol{\delta})-f_{\mathsf{fk,x}}(\boldsymbol{q})}{\delta}, \quad \boldsymbol{\delta} = \begin{pmatrix} \delta & 0 & \cdots \end{pmatrix}^\top$
- Repeat for every element of J

How to compute jacobian numerically

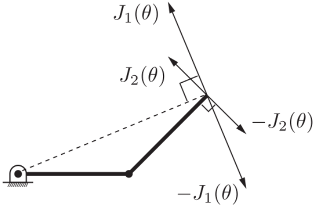
- Finite difference method
  - $f'(x_0) \approx \frac{f(x_0+\delta)-f(x_0)}{\delta}, \quad \delta \to 0$
- $J = \begin{pmatrix} \frac{\partial x}{\partial q_0} & \frac{\partial x}{\partial q_1} & \cdots \\ \frac{\partial y}{\partial q_0} & \frac{\partial y}{\partial q_1} & \cdots \\ \frac{\partial \theta}{\partial q_0} & \frac{\partial \theta}{\partial q_1} & \cdots \end{pmatrix}$
- $\frac{\partial x}{\partial q_0}(\boldsymbol{q}) \approx \frac{f_{\mathsf{fk,x}}(\boldsymbol{q}+\boldsymbol{\delta})-f_{\mathsf{fk,x}}(\boldsymbol{q})}{\delta}, \quad \boldsymbol{\delta} = \begin{pmatrix}\delta & 0 & \cdots\end{pmatrix}^\top$
- Repeat for every element of J
- Slow to compute, easy to implement $\rightarrow$ used in testing

How to compute jacobian analytically



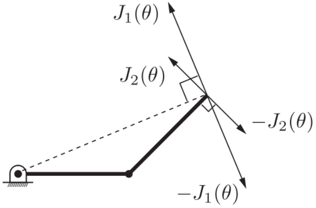
- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column $(\boldsymbol{n}_S)$ is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector

How to compute jacobian analytically



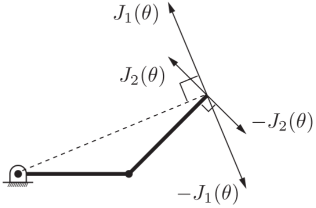
- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column $(\boldsymbol{n}_S)$ is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector
  - S - reference frame, J - frame attached to i-th joint, E end-effector frame

How to compute jacobian analytically



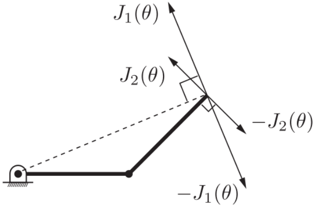
- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column ($\boldsymbol{n}_S$) is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector
  - S - reference frame, J - frame attached to i-th joint, E end-effector frame
  - $\boldsymbol{t}_{JE}$ - translation part of $T_{JE} \in SE(2)$

How to compute jacobian analytically



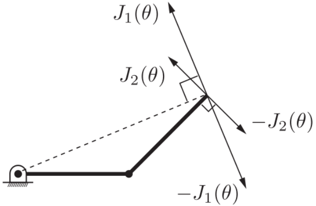
- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column ($\boldsymbol{n}_S$) is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector
  - S - reference frame, J - frame attached to i-th joint, E end-effector frame
  - $\boldsymbol{t}_{JE}$ - translation part of $T_{JE} \in SE(2)$
  - $\boldsymbol{n} = R(90)\boldsymbol{t}_{JE}$ - perpendicular vector

How to compute jacobian analytically



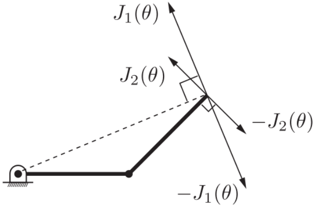
- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column $(\boldsymbol{n}_S)$ is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector
  - S - reference frame, J - frame attached to i-th joint, E end-effector frame
  - $\boldsymbol{t}_{JE}$ - translation part of $T_{JE} \in SE(2)$
  - $\boldsymbol{n} = R(90)\boldsymbol{t}_{JE}$ - perpendicular vector
  - $\boldsymbol{n}_S = R_{SJ}\boldsymbol{n}$ - change of reference frame

How to compute jacobian analytically



- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column ($\boldsymbol{n}_S$) is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector
  - S - reference frame, J - frame attached to i-th joint, E end-effector frame
  - $\boldsymbol{t}_{JE}$ - translation part of $T_{JE} \in SE(2)$
  - $\boldsymbol{n} = R(90)\boldsymbol{t}_{JE}$ - perpendicular vector
  - $\boldsymbol{n}_S = R_{SJ}\boldsymbol{n}$ - change of reference frame
  - For prismatic joints: $\boldsymbol{n}_S = R_{SJ}\boldsymbol{a}$
  - $\boldsymbol{a}$ is axis of translation

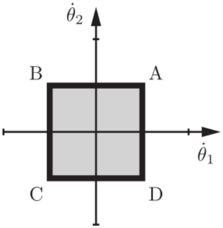
How to compute jacobian analytically



- $J = \begin{pmatrix} J_v & J_w \end{pmatrix}^\top$ *i.e.* translation and rotation part
- Translation part:
  - i-th column ($\boldsymbol{n}_S$) is perpendicular to vector $\boldsymbol{t}$, connecting i-th joint to end-effector
  - S - reference frame, J - frame attached to i-th joint, E end-effector frame
  - $\boldsymbol{t}_{JE}$ - translation part of $T_{JE} \in SE(2)$
  - $\boldsymbol{n} = R(90)\boldsymbol{t}_{JE}$ - perpendicular vector
  - $\boldsymbol{n}_S = R_{SJ}\boldsymbol{n}$ - change of reference frame
  - For prismatic joints: $\boldsymbol{n}_S = R_{SJ}\boldsymbol{a}$
  - $\boldsymbol{a}$ is axis of translation
- Rotation part
  - 1 for revolute joints
  - 0 for prismatic joints

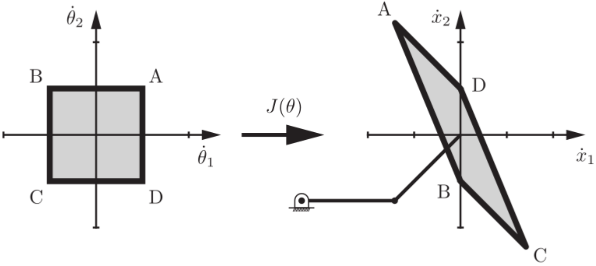
Jacobian application - velocity limits

- $\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
- Velocity limits are given for each joint
  - configuration independent



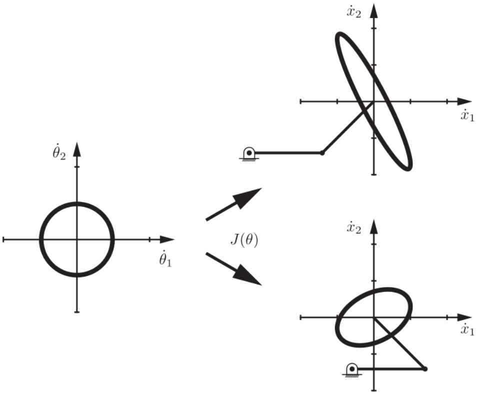
Jacobian application - velocity limits

- $\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
- Velocity limits are given for each joint
  - configuration independent
- What are the velocity we can achieve with end-effector?
  - depends on configuration
  - use jacobian to map joint-space velocity to task-space velocity



Manipulability ellipsoid

- Unit circle in joint velocity space, *i.e.* $\|\dot{\boldsymbol{q}}\| = 1$
- Mapping through Jacobian to ellipsoid in end-effector space
- Closer the ellipsoid is to sphere, more easily can end-effector move in arbitrary direction



How to compute manipulability ellipsoid

$$1 = \|\dot{\boldsymbol{q}}\|$$

How to compute manipulability ellipsoid

$$
\begin{aligned}
1 &= \|\dot{\boldsymbol{q}}\| \\
&= \dot{\boldsymbol{q}}^\top \dot{\boldsymbol{q}}
\end{aligned}
$$

How to compute manipulability ellipsoid

- If $J(\boldsymbol{q})$ is non-singular

$$\begin{aligned}
1 &= \|\dot{\boldsymbol{q}}\| \\
&= \dot{\boldsymbol{q}}^\top \dot{\boldsymbol{q}} \\
&= \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right)^\top \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right)
\end{aligned}$$

How to compute manipulability ellipsoid

- If $J(\boldsymbol{q})$ is non-singular

$$\begin{aligned} 1 &= \|\dot{\boldsymbol{q}}\| \\ &= \dot{\boldsymbol{q}}^\top \dot{\boldsymbol{q}} \\ &= \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right)^\top \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right) \\ &= \dot{\boldsymbol{x}}^\top J(\boldsymbol{q})^{-\top} J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}} \end{aligned}$$

How to compute manipulability ellipsoid

- If $J(\boldsymbol{q})$ is non-singular

$$
\begin{aligned}
1 &= \|\dot{\boldsymbol{q}}\| \\
&= \dot{\boldsymbol{q}}^\top \dot{\boldsymbol{q}} \\
&= \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right)^\top \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right) \\
&= \dot{\boldsymbol{x}}^\top J(\boldsymbol{q})^{-\top} J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^\top \left(J(\boldsymbol{q})J(\boldsymbol{q})^\top\right)^{-1} \dot{\boldsymbol{x}}
\end{aligned}
$$

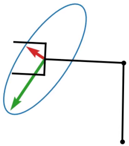
How to compute manipulability ellipsoid

- If $J(\boldsymbol{q})$ is non-singular
- Solution to $\boldsymbol{u}^\top A^{-1}\boldsymbol{u} = 1$ is ellipsoid
  - eigen vectors of A show directions of principal axes of the ellipsoid
  - square roots of eigen values are lengths of the principal axis

$$
\begin{aligned}
1 &= \|\dot{\boldsymbol{q}}\| \\
&= \dot{\boldsymbol{q}}^\top \dot{\boldsymbol{q}} \\
&= \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right)^\top \left(J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}}\right) \\
&= \dot{\boldsymbol{x}}^\top J(\boldsymbol{q})^{-\top} J(\boldsymbol{q})^{-1}\dot{\boldsymbol{x}} \\
&= \dot{\boldsymbol{x}}^\top \left(J(\boldsymbol{q})J(\boldsymbol{q})^\top\right)^{-1} \dot{\boldsymbol{x}}
\end{aligned}
$$

Manipulability ellipsoid example

- 2 DoF robot, translation only, $\mathrm{eig}(JJ^\top)$



How close we are to singularity?

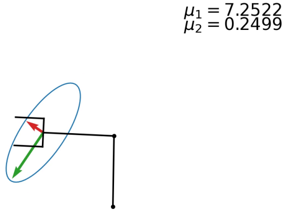
- Condition number of $JJ^\top$
  - $\mu_1 = \frac{\lambda_{\max}(JJ^\top)}{\lambda_{\min}(JJ^\top)} \geq 1$
  - λ is eigen value of a given matrix
  - the larger μ_1 is, the closer to singularity we are
  - **Small μ_1 is preferred**

How close we are to singularity?

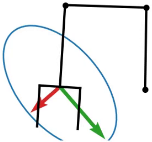
- Condition number of $JJ^\top$
  - $\mu_1 = \frac{\lambda_{\max}(JJ^\top)}{\lambda_{\min}(JJ^\top)} \geq 1$
  - λ is eigen value of a given matrix
  - the larger μ_1 is, the closer to singularity we are
  - **Small μ_1 is preferred**
- Volume of manipulability ellipsoid
  - the smaller volume is, the closer to singularity we are
  - $\mu_2 = \sqrt{\lambda_1 \lambda_2 \cdots} = \det\left(JJ^\top\right)$
  - **Large μ_2 is preferred**

How close we are to singularity?

- Condition number of $JJ^\top$
  - $\mu_1 = \frac{\lambda_{\max}(JJ^\top)}{\lambda_{\min}(JJ^\top)} \geq 1$
  - λ is eigen value of a given matrix
  - the larger μ_1 is, the closer to singularity we are
  - **Small μ_1 is preferred**
- Volume of manipulability ellipsoid
  - the smaller volume is, the closer to singularity we are
  - $\mu_2 = \sqrt{\lambda_1 \lambda_2 \cdots} = \det\left(JJ^\top\right)$
  - **Large μ_2 is preferred**



Redundant robots and singularities



Null-space of jacobian

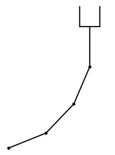
- $\mathsf{Null}(A) = \ker(A) = \{\boldsymbol{x} \mid A\boldsymbol{x} = \mathbf{0}\}$

Null-space of jacobian

- $\text{Null}(A) = \ker(A) = \{\boldsymbol{x} \mid A\boldsymbol{x} = \boldsymbol{0}\}$
- Find $\dot{\boldsymbol{q}}$ s.t. $\dot{\boldsymbol{x}} = \boldsymbol{0}$
  - $\dot{\boldsymbol{q}}_{\text{null}} \in \ker(J)$

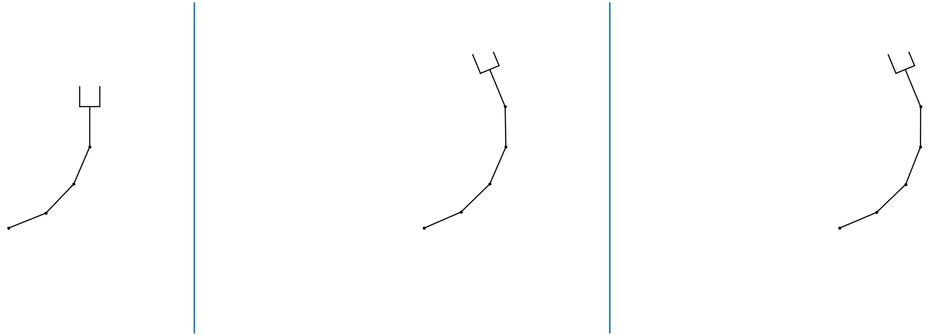
Null-space of jacobian

- $\mathsf{Null}(A) = \ker(A) = \{\boldsymbol{x} \mid A\boldsymbol{x} = \boldsymbol{0}\}$
- Find $\dot{\boldsymbol{q}}$ s.t. $\dot{\boldsymbol{x}} = \boldsymbol{0}$
  - $\dot{\boldsymbol{q}}_{\mathsf{null}} \in \ker(J)$



Null-space of jacobian

- $\mathrm{Null}(A) = \ker(A) = \{\boldsymbol{x} \mid A\boldsymbol{x} = \mathbf{0}\}$
- Find $\dot{\boldsymbol{q}}$ s.t. $\dot{\boldsymbol{x}} = \mathbf{0}$
  - $\dot{\boldsymbol{q}}_{\text{null}} \in \ker(J)$
  - there are multiple solutions if we have more DoF



Statics analysis

- Conservation of power: (power at the joints) = (power to move the robot) + (power at the end-effector)

Statics analysis

- Conservation of power: (power at the joints) = (power to move the robot) + (power at the end-effector)
- Static equilibrium: no power is used to move the robot, *i.e.* no motion

Statics analysis

- Conservation of power: (power at the joints) = (power to move the robot) + (power at the end-effector)
- Static equilibrium: no power is used to move the robot, *i.e.* no motion
  - (power at the joints) = (power at the end-effector)
  - $\boldsymbol{\tau}^\top \dot{\boldsymbol{q}} = \boldsymbol{F}^\top \dot{\boldsymbol{x}}$
    - $\boldsymbol{\tau}$ joint torques
    - $\dot{\boldsymbol{q}}$ joint velocities
    - $\boldsymbol{F}$ end-effector force
    - $\dot{\boldsymbol{x}}$ end-effector velocity

Statics analysis

- Conservation of power: (power at the joints) = (power to move the robot) + (power at the end-effector)
- Static equilibrium: no power is used to move the robot, *i.e.* no motion
  - (power at the joints) = (power at the end-effector)
  - $\boldsymbol{\tau}^\top \dot{\boldsymbol{q}} = \boldsymbol{F}^\top \dot{\boldsymbol{x}}$
    - $\boldsymbol{\tau}$ joint torques
    - $\dot{\boldsymbol{q}}$ joint velocities
    - $\boldsymbol{F}$ end-effector force
    - $\dot{\boldsymbol{x}}$ end-effector velocity
  - $\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
  - $\boldsymbol{\tau}^\top = \boldsymbol{F}^\top J(\boldsymbol{q})$

Statics analysis

- Conservation of power: (power at the joints) = (power to move the robot) + (power at the end-effector)
- Static equilibrium: no power is used to move the robot, *i.e.* no motion
  - (power at the joints) = (power at the end-effector)
  - $\boldsymbol{\tau}^\top \dot{\boldsymbol{q}} = \boldsymbol{F}^\top \dot{\boldsymbol{x}}$
    - $\boldsymbol{\tau}$ joint torques
    - $\dot{\boldsymbol{q}}$ joint velocities
    - $\boldsymbol{F}$ end-effector force
    - $\dot{\boldsymbol{x}}$ end-effector velocity
  - $\dot{\boldsymbol{x}} = J(\boldsymbol{q})\dot{\boldsymbol{q}}$
  - $\boldsymbol{\tau}^\top = \boldsymbol{F}^\top J(\boldsymbol{q})$
  - $\boldsymbol{\tau} = J(\boldsymbol{q})^\top \boldsymbol{F}$

Statics - compensating external force

- Consider external force applied to the end-effector is $-\boldsymbol{F}$.
- How to compute joint torques s.t. robot is static?

Statics - compensating external force

- Consider external force applied to the end-effector is $-\boldsymbol{F}$.
- How to compute joint torques s.t. robot is static?
  - $\boldsymbol{\tau}_{\text{ext}} = J(\boldsymbol{q})^\top \boldsymbol{F}$
  - end-effector needs to generate force $\boldsymbol{F}$ to compensate external $-\boldsymbol{F}$

Statics - compensating external force

- Consider external force applied to the end-effector is $-\boldsymbol{F}$.
- How to compute joint torques s.t. robot is static?
  - $\boldsymbol{\tau}_{\text{ext}} = J(\boldsymbol{q})^\top \boldsymbol{F}$
  - end-effector needs to generate force $\boldsymbol{F}$ to compensate external $-\boldsymbol{F}$
  - this equation assumes gravity does not act on a robot
  - $\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{ext}} + \boldsymbol{\tau}_g$
    - $\boldsymbol{\tau}_g$ compensates gravity acting on a robot

Statics - compensating external force

- Consider external force applied to the end-effector is $-\boldsymbol{F}$.
- How to compute joint torques s.t. robot is static?
  - $\boldsymbol{\tau}_{\text{ext}} = J(\boldsymbol{q})^\top \boldsymbol{F}$
  - end-effector needs to generate force $\boldsymbol{F}$ to compensate external $-\boldsymbol{F}$
  - this equation assumes gravity does not act on a robot
  - $\boldsymbol{\tau} = \boldsymbol{\tau}_{\text{ext}} + \boldsymbol{\tau}_g$
    - $\boldsymbol{\tau}_g$ compensates gravity acting on a robot
  - For Panda robot, you can directly command $\boldsymbol{\tau}_{\text{ext}}$

Force caused by given torques

- If J is invertible (when it is invertible?)
  - $\boldsymbol{F} = J(\boldsymbol{q})^{-\top}\boldsymbol{\tau}$

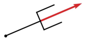
Force caused by given torques

- If J is invertible (when it is invertible?)
  - $\boldsymbol{F} = J(\boldsymbol{q})^{-\top}\boldsymbol{\tau}$
- Redundant robots
  - even for fixed end-effector we can have internal motion
  - static equilibrium assumption is not valid $\rightarrow$ dynamics needed

Force caused by given torques

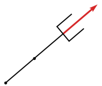
- If J is invertible (when it is invertible?)
  - $\boldsymbol{F} = J(\boldsymbol{q})^{-\top}\boldsymbol{\tau}$
- Redundant robots
  - even for fixed end-effector we can have internal motion
  - static equilibrium assumption is not valid $\rightarrow$ dynamics needed
- Under-actuated robots
  - fixed end-effector will immobilize the robot
  - robot cannot actively generate forces in null-space of $J^\top$: $\ker(J^\top) = \{\boldsymbol{F} \mid J^\top \boldsymbol{F} = \boldsymbol{0}\}$
  - however, robot can resist external force in the null-space without moving

Force caused by given torques



- If J is invertible (when it is invertible?)
  - $\boldsymbol{F} = J(\boldsymbol{q})^{-\top}\boldsymbol{\tau}$
- Redundant robots
  - even for fixed end-effector we can have internal motion
  - static equilibrium assumption is not valid $\rightarrow$ dynamics needed
- Under-actuated robots
  - fixed end-effector will immobilize the robot
  - robot cannot actively generate forces in null-space of $J^\top$: $\ker(J^\top) = \{\boldsymbol{F} \mid J^\top \boldsymbol{F} = \boldsymbol{0}\}$
  - however, robot can resist external force in the null-space without moving
  - red arrow shows null-space

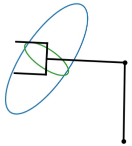
Force caused by given torques



- If J is invertible (when it is invertible?)
  - $\boldsymbol{F} = J(\boldsymbol{q})^{-\top}\boldsymbol{\tau}$
- Redundant robots
  - even for fixed end-effector we can have internal motion
  - static equilibrium assumption is not valid $\rightarrow$ dynamics needed
- Under-actuated robots
  - fixed end-effector will immobilize the robot
  - robot cannot actively generate forces in null-space of $J^\top$: $\ker(J^\top) = \{\boldsymbol{F} \,|\, J^\top \boldsymbol{F} = \boldsymbol{0}\}$
  - however, robot can resist external force in the null-space without moving
  - red arrow shows null-space
- Singularities (square J, but non-invertible)
  - non-zero null-space

Force ellipsoid

- How easy is to generate force in a given direction.
- Eigen analysis of $(JJ^\top)^{-1}$
  - Blue - manipulability ellipsoid (i.e. $JJ^\top$)
  - Green - force ellipsoid (i.e. $(JJ^\top)^{-1}$)
- Easy motion in a direction $\rightarrow$ difficult to compensate force in that direction
- Close to singularity:
  - area of manipulability ellipsoid $\rightarrow 0$
  - area of force ellipsoid $\rightarrow \infty$



Summary

- Differential kinematics
  - Jacobian and its properties
  - How to compute Jacobian
  - Manipulability ellipsoids
  - How to measure distance to singularity
- Statics
  - Static equilibrium relation of joint torques and task-space forces
  - Force ellipsoids

Laboratory

- Implementation of jacobian computation for planar manipulator
  - Finite difference method
  - Analytical method
- Generation of movement in null-space