Advanced Robotics

Forward Kinematics and Inverse Kinematics

Steve Tonneau School of Informatics
University of Edinburgh

Reading for this week

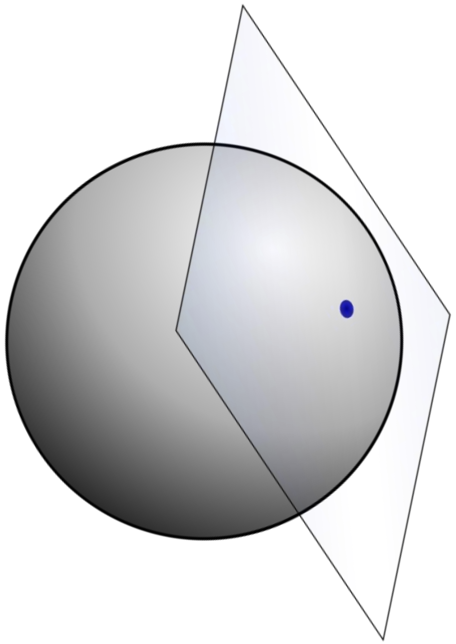
- Siciliano, B., et al., Robotics: Modelling, Planning and Control.

  Chapter 3.2 , 3.3, 3.5

- (If you are really motivated) Roy Featherstone, Rigid body dynamics algorithms

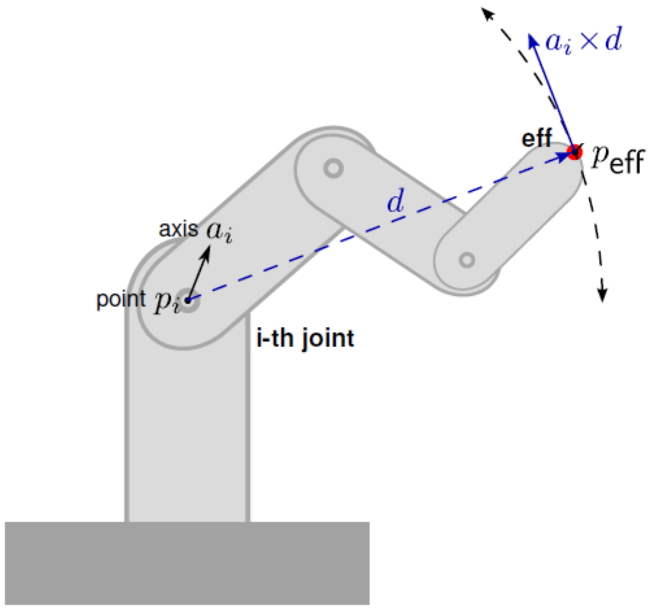
  Chapter 2.1, 2.2

An informal introduction



An informal introduction

How does an infinetesimal change in configuration affects end-effector linear velocity ?



$$\delta p_{\text{eff}} = \delta q_i [a_i \times (p_{\text{eff}} - p_i)]$$

$$J_{\text{pos}}(q) = \begin{pmatrix} [a_1 \times (p_{\text{eff}} - p_1)] & [a_2 \times (p_{\text{eff}} - p_2)] & \cdots & [a_n \times (p_{\text{eff}} - p_n)] \end{pmatrix} \in \mathbb{R}^{3 \times n}$$

Our objectives for today

- Forward **kinematics** consists, given configuration and velocity in configuration space, in computing the velocity of a rigid body in the cartesian space:

$$FK : \mathbf{q}, \mathbf{v}_q \longrightarrow \nu$$

- Inverse **kinematics** consists, given a desired velocity ν^* in the cartesian space (and the current configuration), in computing a velocity in the configuration space result in a velocity as close as possible to ν^*:

$$IK : \nu^*, \mathbf{q} \longrightarrow \mathbf{v}_q$$

But before, minimal info on spatial velocity

An element of the group…. | Can be represented as …

$$r \in SO(3) \qquad \simeq \qquad \mathbf{R} \in \mathbb{R}^{3\times 3}$$ Rotation matrix

rotation

$$\mathbf{q} \in \mathbb{H} \simeq \mathbb{R}^4, \|\mathbf{q}\| = \mathbf{1}$$ quaternion

$$\omega \in so(3) \simeq \mathbb{R}^3$$ A velocity ?

$$m \in SE(3) \qquad \simeq \qquad \mathbb{R}^3 \times SO(3) \qquad \simeq \qquad \mathbb{R}^3 \times \mathbb{R}^{3\times 3} \simeq \mathbb{R}^{4\times 4}$$ Homogeneous matrix

displacement

translation rotation

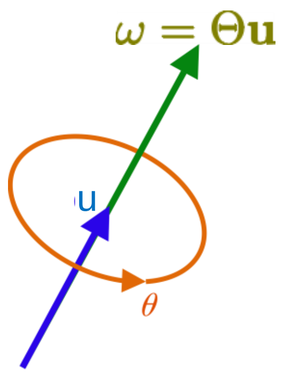
$$\mathbb{R}^3 \times \mathbb{H} \simeq \mathbb{R}^7$$

$$\boxed{\mathcal{V} \in se(3) = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \simeq \mathbb{R}^6 \quad \text{A spatial velocity ?}}$$

Spatial velocity vector

- **v** linear velocity
- ω angular velocity

$$\mathcal{V} \in se(3) = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \simeq \mathbb{R}^6$$

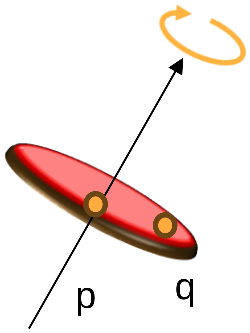


Spatial velocity vector

- **v** linear velocity
  - Not completely intuitive

$$\mathcal{V} \in se(3) = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \simeq \mathbb{R}^6$$

$\mathbf{v}_q = \mathbf{v}_p + \omega \times \overrightarrow{pq}$: velocity of the point of the rigid body currently coinciding with **q** seen from p

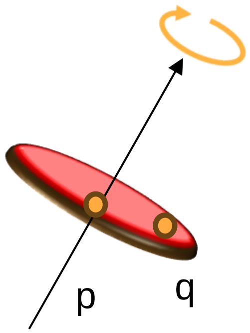


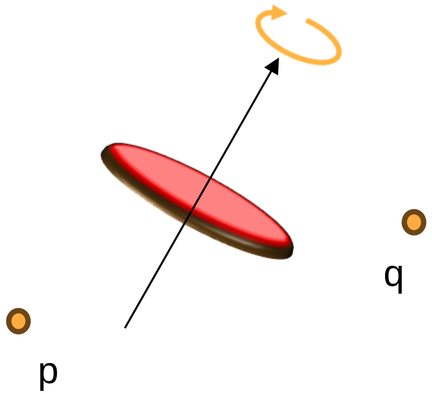
Spatial velocity vector

- **v** linear velocity
  - Not completely intuitive

$$\mathcal{V} \in se(3) = \begin{bmatrix} \mathbf{v} \\ \omega \end{bmatrix} \simeq \mathbb{R}^6$$

$\mathbf{v}_q = \mathbf{v}_p + \omega \times \overrightarrow{pq}$: velocity of the point of the rigid body currently coinciding with **q** seen from p



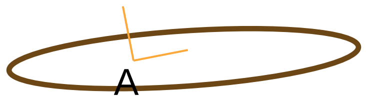


Also true. Defined beyond a rigid body: vector field

Spatial velocity vector

- $\mathbf{v}$ linear velocity
- ω angular velocity

$${}^{A}\mathcal{V} \in se(3) = \begin{bmatrix} {}^{A}\mathbf{v}_{A} \\ {}^{A}\omega \end{bmatrix} \simeq \mathbb{R}^6$$



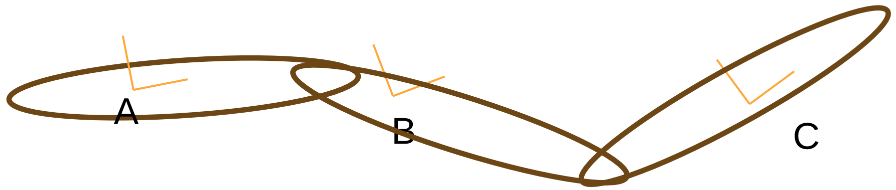
Spatial velocity vector

- $\mathbf{v}$ linear velocity
- ω angular velocity

$$ {}^{A}\mathcal{V} \in se(3) = \begin{bmatrix} {}^{A}\mathbf{v}_{A} \\ {}^{A}\omega \end{bmatrix} \simeq \mathbb{R}^6 $$

- ${}^{A}\mathcal{V}_{BC}$: spatial velocity of frame C wrt to frame B, expressed in frame A

$$ {}^{A}\mathcal{V}_{AC} = {}^{A}\mathcal{V}_{AB} + {}^{A}\mathcal{V}_{BC} $$



Spatial velocity vector

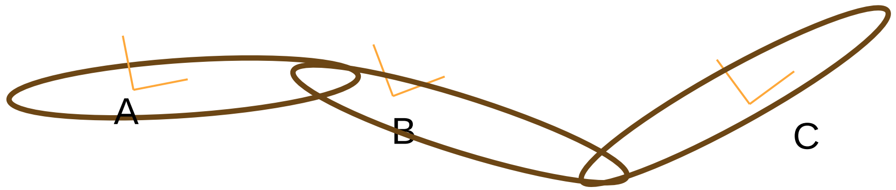
- $\mathbf{v}$ linear velocity
- ω angular velocity

$$ {}^{A}\mathcal{V} \in se(3) = \begin{bmatrix} {}^{A}\mathbf{v} \\ {}^{A}\omega \end{bmatrix} \simeq \mathbb{R}^6 $$

- ${}^{A}\mathcal{V}_{BC}$: spatial velocity of frame C wrt to frame B, expressed in frame A

$$ {}^{A}\mathcal{V}_{AC} = {}^{A}\mathcal{V}_{AB} + {}^{A}\mathcal{V}_{BC} $$

$$ {}^{A}\mathcal{V}_{AC} = {}^{A}\mathcal{V}_{AB} + {}^{A}\mathbf{X}_{B}\,{}^{B}\mathcal{V}_{BC} $$



Spatial velocity vector

- $\mathbf{v}$ linear velocity
- ω angular velocity

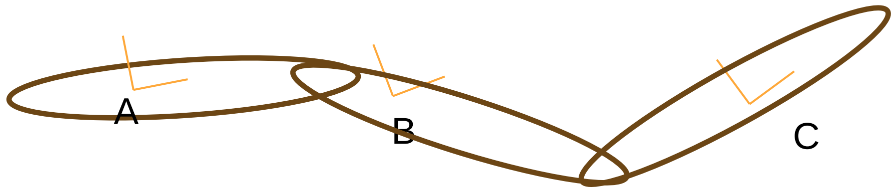
$${}^{A}\mathcal{V} \in se(3) = \begin{bmatrix} {}^{A}\mathbf{v}_{A} \\ {}^{A}\omega \end{bmatrix} \simeq \mathbb{R}^6$$

- ${}^{A}\mathcal{V}_{BC}$: spatial velocity of frame C wrt to frame B, expressed in frame A

$${}^{A}\mathcal{V}_{AC} = {}^{A}\mathcal{V}_{AB} + {}^{A}\mathcal{V}_{BC}$$

$${}^{A}\mathcal{V}_{AC} = {}^{A}\mathcal{V}_{AB} + {}^{A}\mathbf{X}_{B}\,{}^{B}\mathcal{V}_{BC}$$

with ${}^{A}X_{B}$ the « adjoint » or « action » matrix



$${}^{A}\mathbf{X}_{B} = \begin{bmatrix} {}^{A}\mathbf{R}_{B} & {}^{A}\overrightarrow{\mathbf{AB}}_{\times}\,{}^{A}\mathbf{R}_{B} \\ 0 & {}^{A}\mathbf{R}_{B} \end{bmatrix}$$

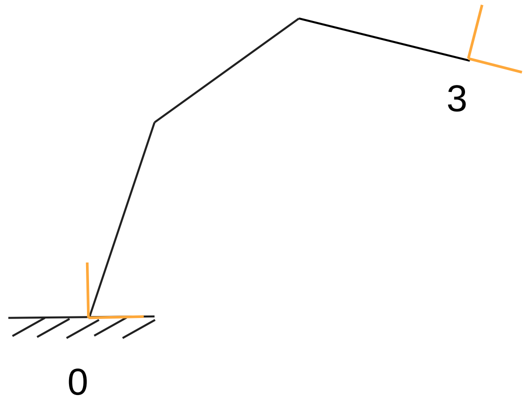
Let’s just accept this for now

Forward kinematics

- The spatial velocity vector between the root frame {0} of a kinematic tree and the end effector frame {3} can depend both on the position and velocity in configuration space.

$$ {}^{0}\nu_{03}(\mathbf{q},\ v_q) \quad \text{and} \quad {}^{3}\nu_{03}(\mathbf{q},\ v_q) $$

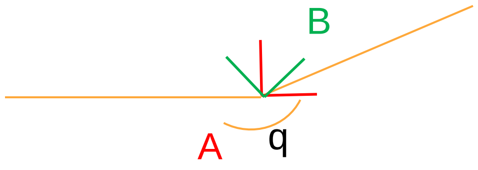
- However for most classical joints it only depends on $\mathbf{v_q}$



Example: revolute joint around x axis

$$
{}^{A}M_{B}(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q) & \sin(q) & 0 \\ 0 & -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

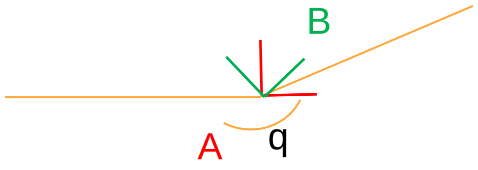
$$
v_q = \dot{q}
$$



Example: revolute joint around x axis

$$^{A}M_{B}(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(q) & \sin(q) & 0 \\ 0 & -\sin(q) & \cos(q) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{A}\nu_{AB}(\dot{q}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \dot{q}$$

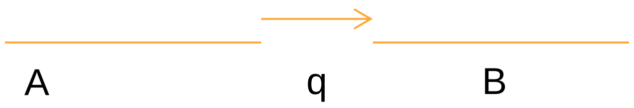
$$v_q = \dot{q}$$



Example: prismatic z joint

$$^{A}M_{B}(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & q \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad {}^{A}\nu_{AB}(\dot{q}) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \dot{q}$$

$$v_q = \dot{q}$$



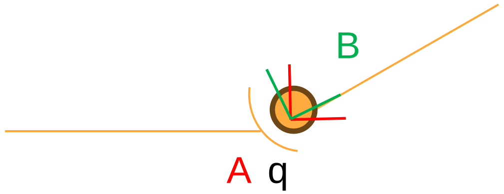
Example: spherical joint

$$\underline{q} \in \mathbb{R}^4, \quad \|\underline{q}\| = 1$$

$$^{A}M_B(q) = \begin{bmatrix} & \mathcal{R}(\underline{q}) & & \begin{matrix}0\\0\\0\end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$v_q = \omega$$

It's a choice



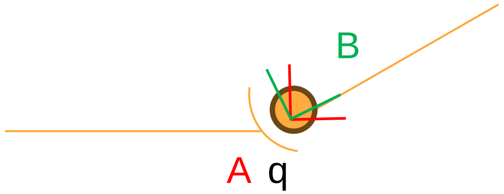
Example: spherical joint

$$\underline{q} \in \mathbb{R}^4, \quad \|\underline{q}\| = 1$$

$$^A M_B(q) = \begin{bmatrix} & \mathcal{R}(\underline{q}) & & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad ^A\nu_{AB}(v_q) = \begin{bmatrix} 0_{3\times 3} \\ I_{3\times 3} \end{bmatrix} v_q$$

$$v_q = \omega$$

It's a choice



In general, forward kinematics for one joint

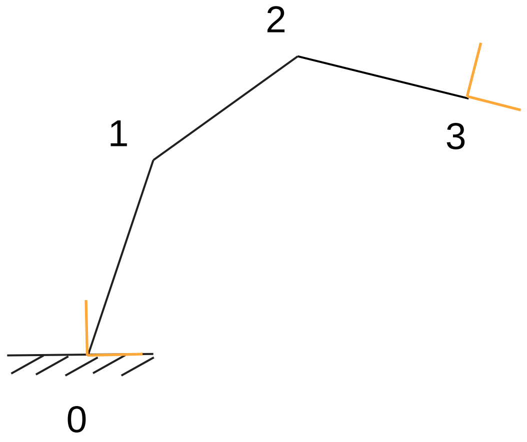
$$ {}^{B}\nu_{AB}(q, v_q) \;=\; {}^{B}J(q)v_q $$

$$ \frac{\partial f}{\partial t} \qquad\qquad \frac{\partial f}{\partial q} \qquad\qquad \frac{\partial q}{\partial t} $$

J is a joint Jacobian (or at least behaves as such):

Forward kinematics for a chain of joints:

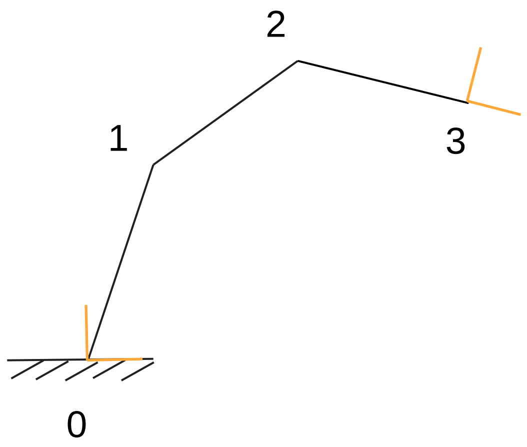
$$^{0}\nu_{03}(q, v_q) = {}^{0}\nu_{01} + {}^{0}\nu_{12} + {}^{0}\nu_{23}$$



Forward kinematics for a chain of joints:

$$^{0}\nu_{03}(q, v_q) = {}^{0}\nu_{01} + {}^{0}\nu_{12} + {}^{0}\nu_{23}$$

$$^{0}\nu_{03}(q, v_q) = {}^{0}\mathbf{X}_1{}^{1}\nu_{01} + {}^{0}\mathbf{X}_2{}^{2}\nu_{12} + {}^{0}\mathbf{X}_3{}^{3}\nu_{23}$$

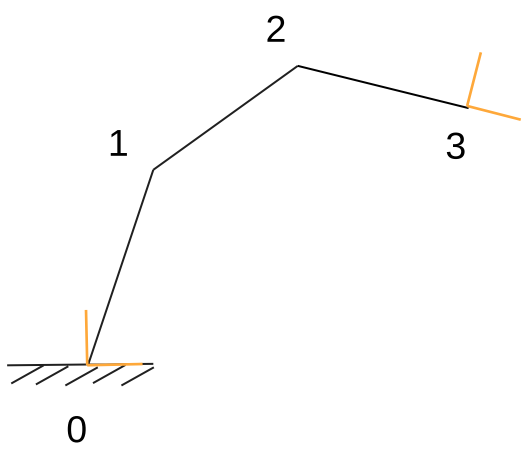


Forward kinematics for a chain of joints:

$$^{0}\nu_{03}(q, v_q) = {}^{0}\nu_{01} + {}^{0}\nu_{12} + {}^{0}\nu_{23}$$

$$^{0}\nu_{03}(q, v_q) = {}^{0}\mathbf{X}_1 {}^{1}\nu_{01} + {}^{0}\mathbf{X}_2 {}^{2}\nu_{12} + {}^{0}\mathbf{X}_3 {}^{3}\nu_{23}$$

$$^{0}\nu_{03}(q, v_q) = {}^{0}\mathbf{X}_1 {}^{1}J_1(q_1) v_{q_1} + {}^{0}\mathbf{X}_2 {}^{2}J_2(q_2) v_{q_2} + {}^{0}\mathbf{X}_3 {}^{3}J_3(q_3) v_{q_3}$$



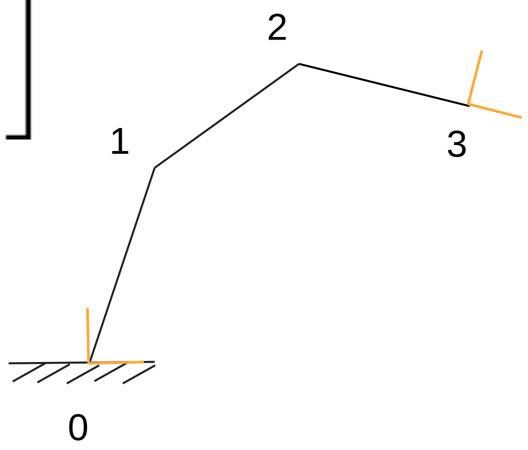
Forward kinematics for a chain of joints:

$${}^{0}\nu_{03}(q, v_q) = {}^{0}\nu_{01} + {}^{0}\nu_{12} + {}^{0}\nu_{23}$$

$${}^{0}\nu_{03}(q, v_q) = {}^{0}\mathbf{X}_1 {}^{1}\nu_{01} + {}^{0}\mathbf{X}_2 {}^{2}\nu_{12} + {}^{0}\mathbf{X}_3 {}^{3}\nu_{23}$$

$${}^{0}\nu_{03}(q, v_q) = {}^{0}\mathbf{X}_1 {}^{1}J_1(q_1) v_{q_1} + {}^{0}\mathbf{X}_2 {}^{2}J_2(q_2) v_{q_2} + {}^{0}\mathbf{X}_3 {}^{3}J_3(q_3) v_{q_3}$$

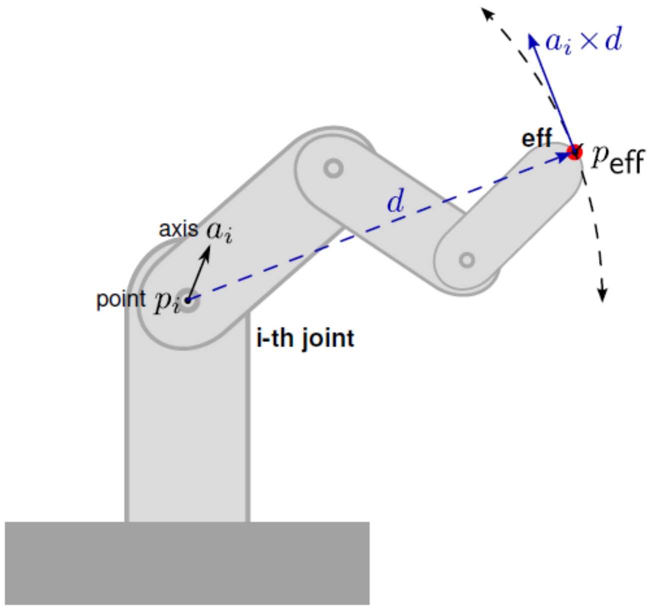
$${}^{0}\nu_{03}(q, v_q) = \begin{bmatrix} {}^{0}\mathbf{X}_1 {}^{1}J_1(q_1) & {}^{0}\mathbf{X}_2 {}^{2}J_2(q_2) & {}^{0}\mathbf{X}_3 {}^{3}J_3(q_3) \end{bmatrix} \begin{bmatrix} v_{q_1} \\ v_{q_2} \\ v_{q_3} \end{bmatrix}$$



$$\boxed{\nu(\mathbf{q}, v_{\mathbf{q}}) = \mathbf{J}(\mathbf{q}) v_{\mathbf{q}}}$$

An informal introduction

How does an infinetesimal change in configuration affects end-effector linear velocity ?

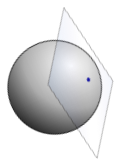


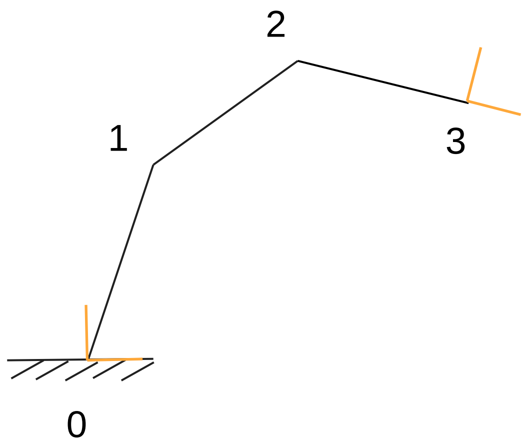
$$\delta p_{\text{eff}} = \delta q_i [a_i \times (p_{\text{eff}} - p_i)]$$

$$J_{\text{pos}}(q) = \begin{pmatrix} [a_1 \times (p_{\text{eff}} - p_1)] & [a_2 \times (p_{\text{eff}} - p_2)] & \cdots & [a_n \times (p_{\text{eff}} - p_n)] \end{pmatrix} \in \mathbb{R}^{3 \times n}$$

Forward kinematics for a chain of joints:

- For any given q, J(q) is straightforward to compute
- If **J** is known, there is a linear mapping from a velocity in the configuration to a velocity in the cartesian space
- J is only valid **locally**





$$\boxed{\nu(\mathbf{q}, v_{\mathbf{q}}) = \mathbf{J}(\mathbf{q}) v_{\mathbf{q}}}$$

Inverse kinematics is an inversion problem

$$\nu(\mathbf{q}, v_{\mathbf{q}}) = \mathbf{J}(\mathbf{q})v_{\mathbf{q}}$$

$$\Longrightarrow$$

$$v_{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1}\nu(\mathbf{q}, v_{\mathbf{q}})$$

Inverse kinematics is an inversion problem

$$\nu(\mathbf{q}, v_{\mathbf{q}}) = \mathbf{J}(\mathbf{q}) v_{\mathbf{q}}$$

$$\Longrightarrow$$

$$v_{\mathbf{q}} = \mathbf{J}(\mathbf{q})^{-1} \nu(\mathbf{q}, v_{\mathbf{q}})$$

J is most likely not invertible, so what?

We can formulate IK as optimisation problem

- Also known in literature as differential inverse kinematics of closed loop inverse kinematics (click)

- Given q and a target end-effector velocity ν^*, find a joint velocity v_q that results in a end-effector velocity ν as close as possible to ν^*

$$\min_{v_q} ||\nu(q, v_q) - \nu^*||^2$$

How is this similar to inverse geometry ?

- IG:

$$\min_q \ \mathrm{dist}({}^0M_e(q), {}^0M_*)$$

$$\min_{v_q} \ \mathrm{dist}({}^0M_e(q_0 \oplus v_q), {}^0M_*)$$

- IK:

$$\min_{v_q} ||\nu(q, v_q) - \nu^*||^2$$

$$\min_{v_q} ||J(q)v_q - \nu^*||^2$$

How is this similar to inverse geometry ?

- IG:

$$\min_q \ \text{dist}({}^0M_e(q), {}^0M_*)$$

$$\min_{v_q} \ \text{dist}({}^0M_e(q_0 \oplus v_q), {}^0M_*)$$

Non-linear

- IK:

$$\min_{v_q} ||\nu(q, v_q) - \nu^*||^2$$

Linear

$$\min_{v_q} ||J(q)v_q - \nu^*||^2$$

Solution to the unconstrained IK problem:

$$v_q{}^* = J^\dagger \nu^*$$

With $J^\dagger$ Moore Penrose pseudo-inverse

However, we could consider additional constraints to our problem: joint limits, velocity limits, etc:

IK with constraints

- Velocity bounds (element-wise)

$$\min_{v_q} ||J(q)v_q - \nu^*||^2$$
$$\text{s.t.} \quad v_q^- \leq v_q \leq v_q^+$$

- Joint bounds (using euler integration over a time step)

$$\mathbf{q}^- \leq \mathbf{q} + \Delta t v_q \leq \mathbf{q}^+$$

- Can also add other cost functions…

- ${v_q}^* = J^\dagger \nu^*$ no longer optimal solution

However, easy to solve using a Quadratic Program solver (e.g. quadprog)

IK vs IG

- Inverse kinematics (also called differential IK) is a linear, convex problem, very easy to solve
- Inverse geometry (also called IK) is a non-linear problem, very hard to solve
- When trying to solve IG iteratively, we can use the pseudo-inverse of the jacobian to locally update a configuration towards one that is closer to the goal. This is similar to performing one step of gradient descent (See example after)

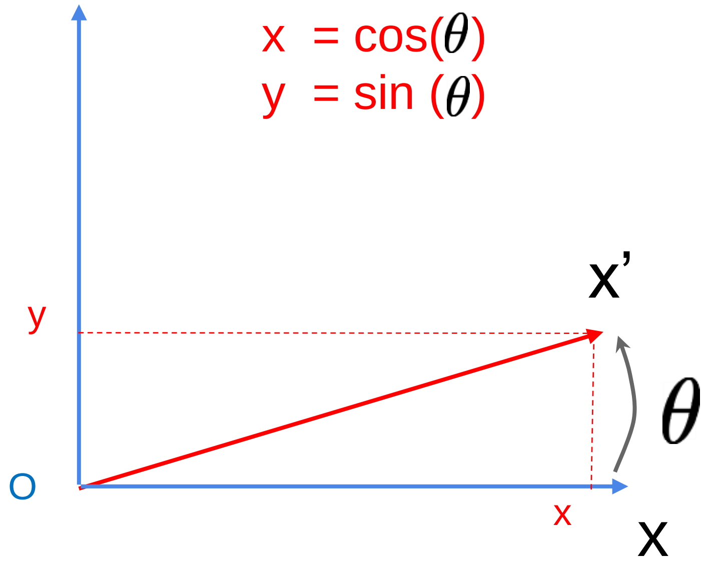
A really simple articulated robot

- 1 Dof, unit norm link (clock)
  - FG:

$$\phi(q) = \phi(\theta) = [cos(\theta), sin(\theta)]^T$$

  - IG:

$$\phi^{-1}([x, y]^T) = atan2(y, x)$$

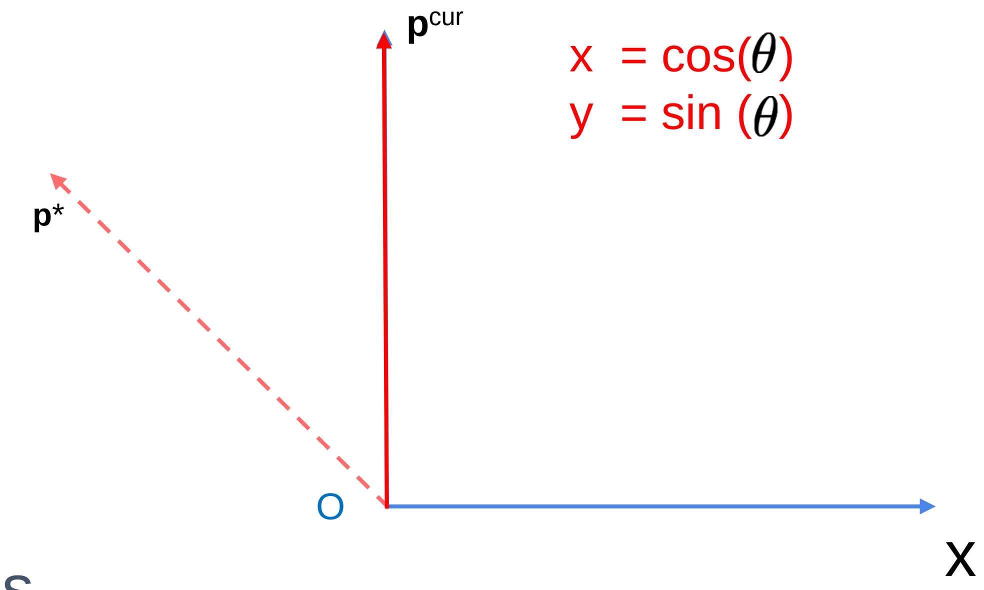


- This problem has an analytical solution. **When solution exists, analytical IG defined if num (dof) <= dim(task)** (necessary condition)

- What if we did not have an analytical solution?

A toy problem

- Current θ= pi/2 ⇔ $\mathbf{p}^{cur}$ = [0, 1]
- Target: $\mathbf{p}^*$ = [-$\sqrt{2}$,$\sqrt{2}$]



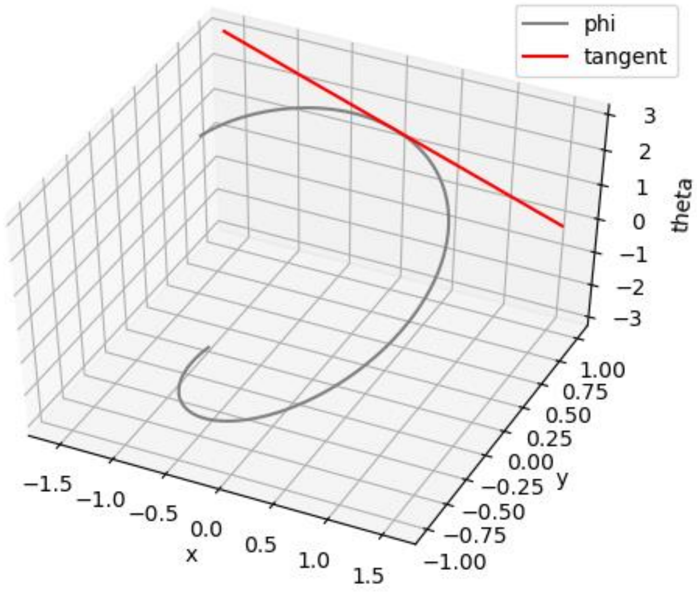
$x = \cos(\theta)$
$y = \sin(\theta)$

- Idea: 'nudge' θ to see what happens

  Finite differences?

A toy problem

- Current θ= pi/2 ⇔ $\mathbf{p}^{cur}$ = [0, 1]
- Target: **p*** = [-$\sqrt{2}$,$\sqrt{2}$]
- The partial derivatives might give us information



$$\delta p = \frac{\partial}{\partial_q}\phi(q)\delta q$$

Local derivative indicate how an infinitesimal change in configuration affects **p**

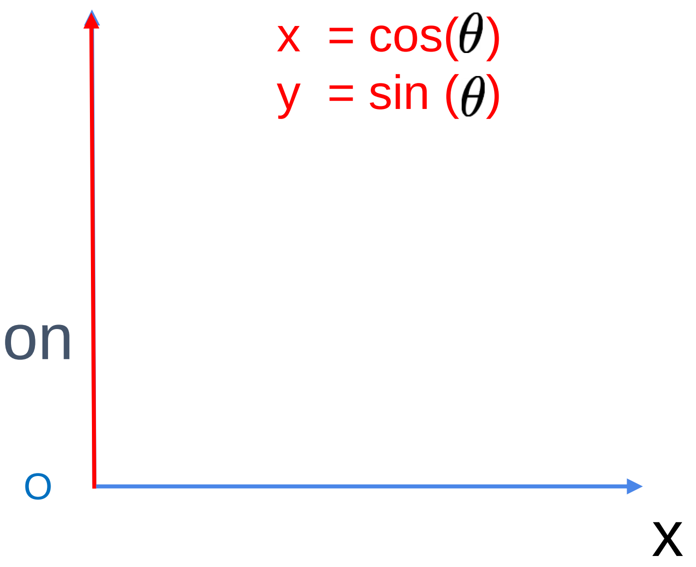
A toy problem

- Current θ= pi/2 ⇔ $\mathbf{p}^{cur}$ = [0, 1]

- Target: $\mathbf{p}^*$ = [-$\sqrt{2}$,$\sqrt{2}$]

- The partial derivatives might give us information

$x = \cos(\theta)$
$y = \sin(\theta)$



$$\frac{\partial}{\partial_\theta}\phi(\theta) = \begin{bmatrix} \frac{\partial_x}{\partial_\theta} \\ \frac{\partial_y}{\partial_\theta} \end{bmatrix}$$

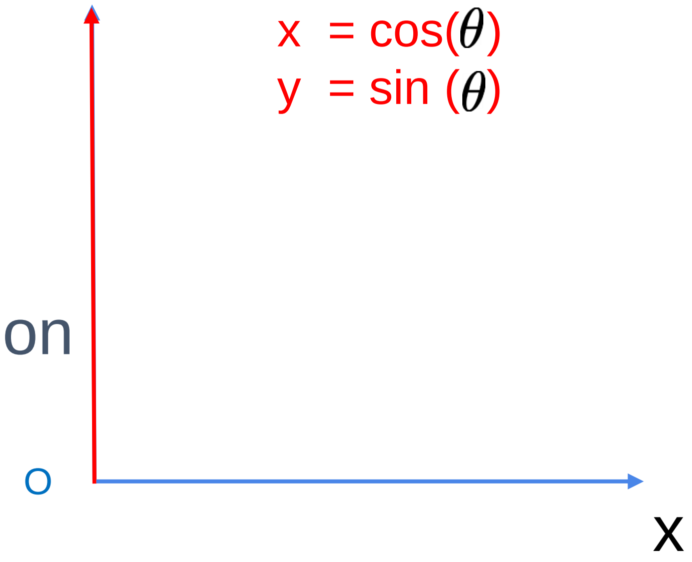
$$\frac{\partial_x}{\partial_\theta} = -sin(\theta)$$

$$\frac{\partial_y}{\partial_\theta} = cos(\theta)$$

Toy problem

- Current θ= pi/2 ⇔ $\mathbf{p}^{cur}$ = [0, 1]
- Target: **p*** = [-$\sqrt{2}$,$\sqrt{2}$]
- The partial derivatives might give us information

x = cos(θ)
y = sin (θ)



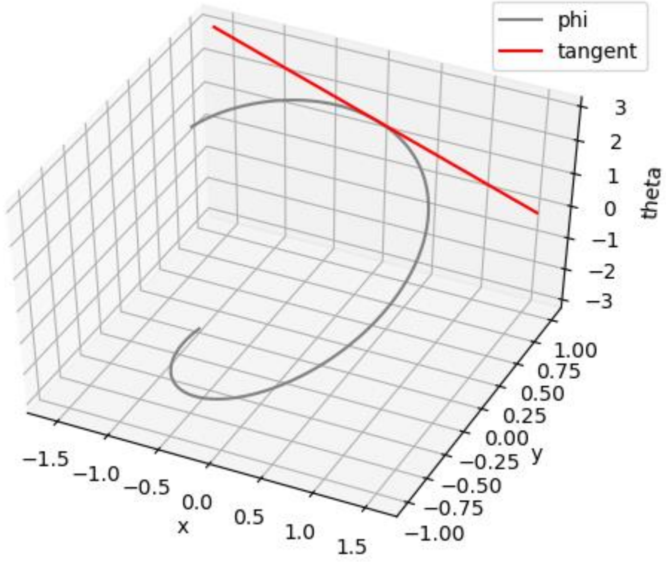
$$\frac{\partial}{\partial_\theta}\phi(\theta) = \begin{bmatrix} \frac{\partial_x}{\partial_\theta} \\ \frac{\partial_y}{\partial_\theta} \end{bmatrix}$$

$$\frac{\partial_x}{\partial_\theta} = -sin(\theta) \qquad \Leftrightarrow \qquad \frac{\partial}{\partial_\theta}\phi(\frac{\pi}{2}) = [-1, 0]^T$$

$$\frac{\partial_y}{\partial_\theta} = cos(\theta)$$

A gradient descent approach

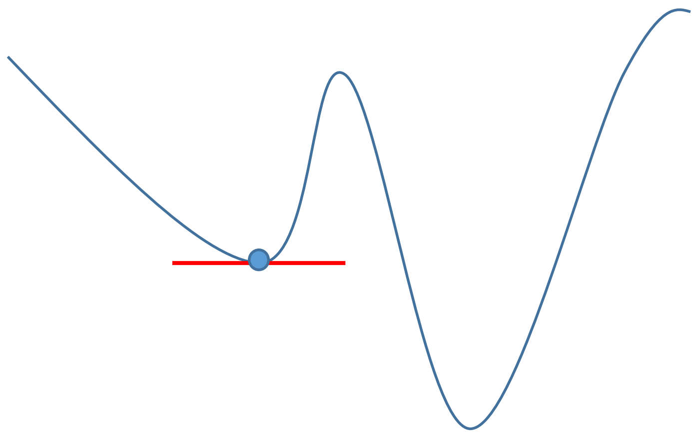
$$\frac{\partial}{\delta_\theta}\phi(\frac{\pi}{2}) = [-1, 0]^T$$



- ❑ Slope of the tangent at pi/2
- ❑ Thus, local linear approximation of Phi
  - ❑ Locally, if $\theta \nearrow$, x $\searrow$ (not true if we increase θ too much)
  - ❑ Nothing to say about y?
- ❑ To reach [- $\sqrt{2}$,$\sqrt{2}$], we have an idea of a baby step to make:
  - ❑ Increase θ a little. If target is reached, we won, otherwise…
  - ❑ …start again: compute the new partial derivatives, slope etc

Gradient descent => find the minimum of a function

- Here distance between current position / target can be used as such function
- Trying to find the **global minima** will not always work…



Exercise: compute partial derivatives for an arm with 2 unit length axes (2 dofs, ie 2 angle values)