ETH zürich

Lecture «Robot Dynamics»: Kinematic Control

151-0851-00 V

lecture:	HG F3	Tuesday 10:15 – 12:00, every week
exercise:	HG D7.1	Wednesday 8:15 – 10:00, according to schedule

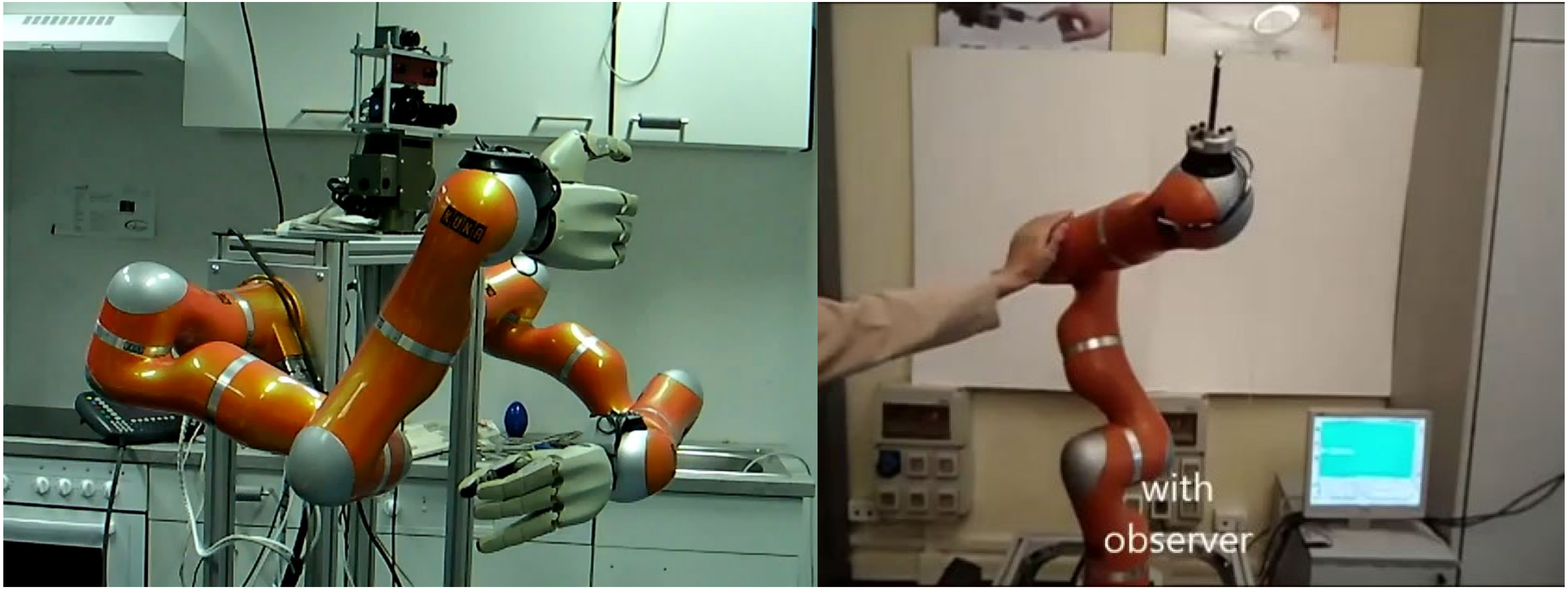
Marco Hutter, Roland Siegwart, and Thomas Stastny

17.09.2019	Intro and Outline	Course Introduction; Recapitulation Position, Linear Velocity			
24.09.2019	Kinematics 1	Rotation and Angular Velocity; Rigid Body Formulation, Transformation	25.09.2019	Exercise 1a	Kinematics Modeling the ABB arm
01.10.2019	Kinematics 2	Kinematics of Systems of Bodies; Jacobians	02.10.2019	Exercise 1a	Differential Kinematics of the ABB arm
08.10.2019	Kinematics 3	Kinematic Control Methods: Inverse Differential Kinematics, Inverse Kinematics; Rotation Error; Multi-task Control	09.10.2019	Exercise 1b	Kinematic Control of the ABB Arm
15.10.2019	Dynamics L1	Multi-body Dynamics	16.10.2019	Midterm 1	Programming kinematics with matlab
22.10.2019	Dynamics L2	Floating Base Dynamics	23.10.2019	Exercise 2a	Dynamic Modeling of the ABB Arm
29.10.2019	Dynamics L3	Dynamic Model Based Control Methods	30.10.2019	Exercise 2b	Dynamic Control Methods Applied to the ABB arm
05.11.2019	Legged Robot	Dynamic Modeling of Legged Robots & Control	06.11.2019	Midterm 2	Programming dynamics with matlab
12.11.2019	Case Studies 1	Legged Robotics Case Study	13.11.2019	Exercise 3	Legged robot
19.11.2019	Rotorcraft	Dynamic Modeling of Rotorcraft & Control	20.11.2019		
26.11.2019	Case Studies 2	Rotor Craft Case Study	27.11.2019	Exercise 4	Modeling and Control of Multicopter
03.12.2019	Fixed-wing	Dynamic Modeling of Fixed-wing & Control	04.12.2019		
10.12.2019	Case Studies 3	Fixed-wing Case Study (Solar-powered UAVs - AtlantikSolar, Vertical Take-off and Landing UAVs – Wingtra)	11.12.2019	Exercise 5	Fixed-wing Control and Simulation
17.12.2019	Summery and Outlook	Summery; Wrap-up; Exam			

Outline

- Kinematic control methods
  - Inverse kinematics
  - Singularities, redundancy
  - Multi-task control
  - Iterative inverse differential kinematics
  - Kinematic trajectory control

Null-space

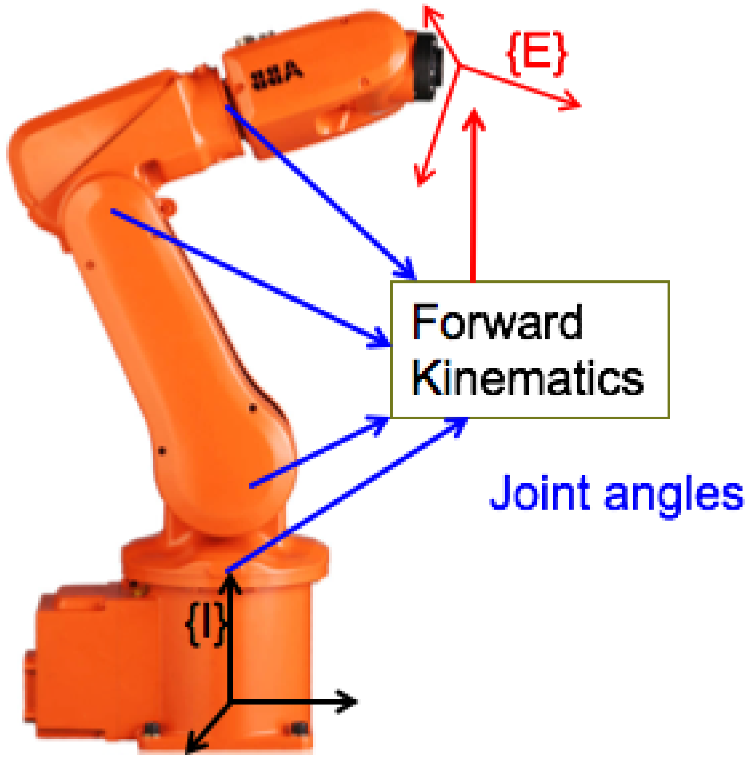


Forward kinematics

- Forward kinematics
  - Description of end-effector configuration (position & orientation) as a function of joint coordinates
  - Use the homogeneous transformation matrix
    - $$\mathbf{T}_{\mathcal{I}\mathcal{E}}(\mathbf{q}) = \begin{bmatrix} \mathbf{C}_{IE}(\mathbf{q}) & {}_{\mathcal{I}}\mathbf{r}_{IE}(\mathbf{q}) \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$$
    - Parametrized description

$$\mathbf{x}_e = \begin{pmatrix} \mathbf{r}_e(\mathbf{q}) \\ \phi_e(\mathbf{q}) \end{pmatrix} = f(\mathbf{q})$$

$$\chi_e = \begin{pmatrix} \chi_{eP} \\ \chi_{eR} \end{pmatrix}$$

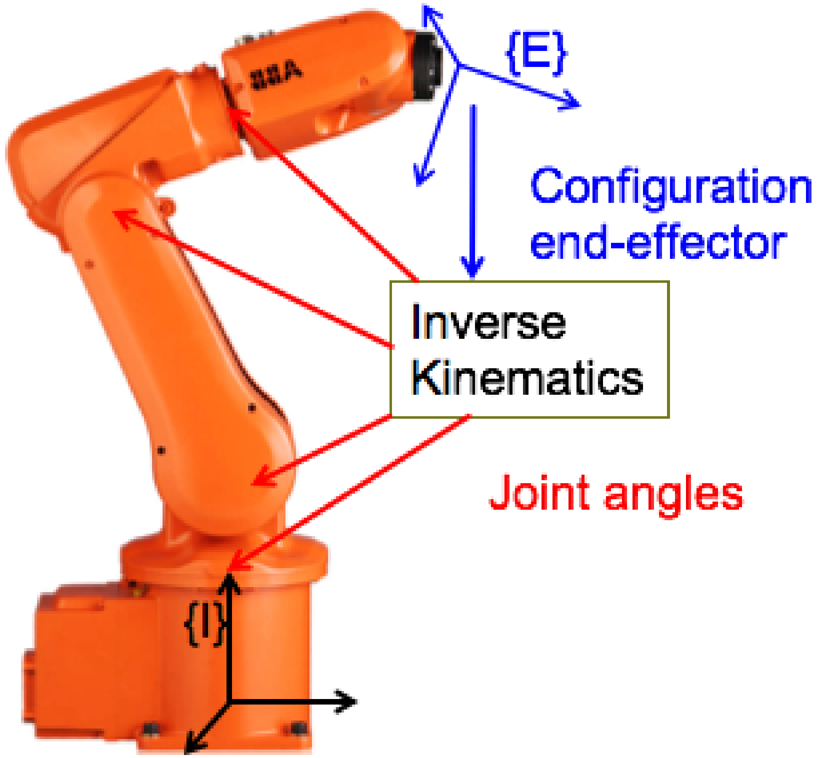


Inverse kinematics

- Inverse kinematics
  - Description of joint angles as a function of the end-effector configuration
  - Use the homogeneous transformation matrix
    - $\mathbf{T}_{\mathcal{IE}}(\mathbf{q}) = \begin{bmatrix} \mathbf{C}_{IE}(\mathbf{q}) & {}_{\mathcal{I}}\mathbf{r}_{IE}(\mathbf{q}) \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4}$
    - Parametrized description

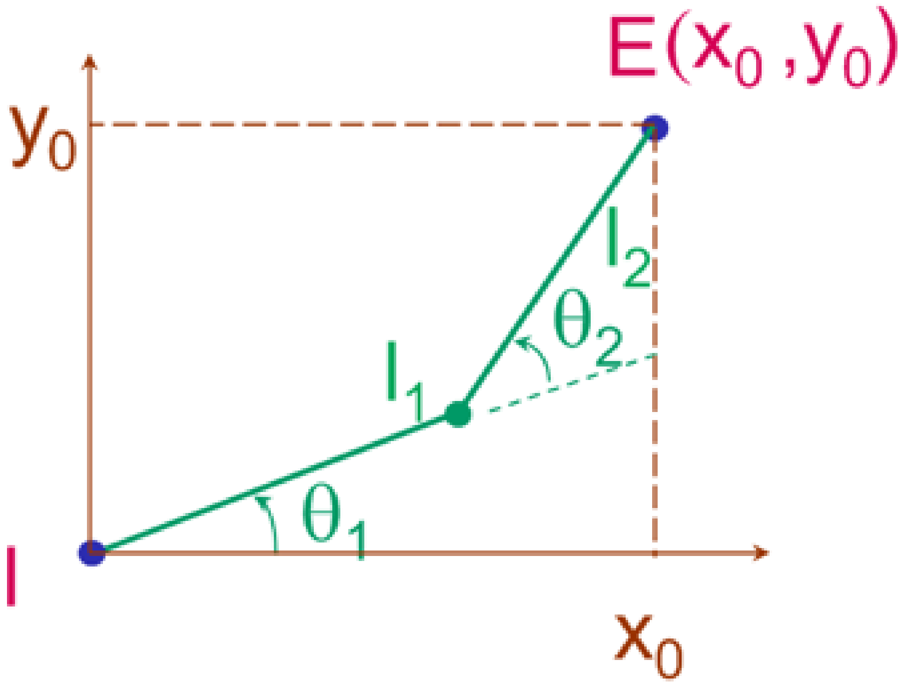
$$\mathbf{x}_e = \begin{pmatrix} \mathbf{r}_e(\mathbf{q}) \\ \phi_e(\mathbf{q}) \end{pmatrix} = f(\mathbf{q}) \qquad \mathbf{q} = \mathbf{f}^{-1}(\mathbf{x}_E)$$

$$\chi_e = \begin{pmatrix} \chi_{eP} \\ \chi_{eR} \end{pmatrix}$$



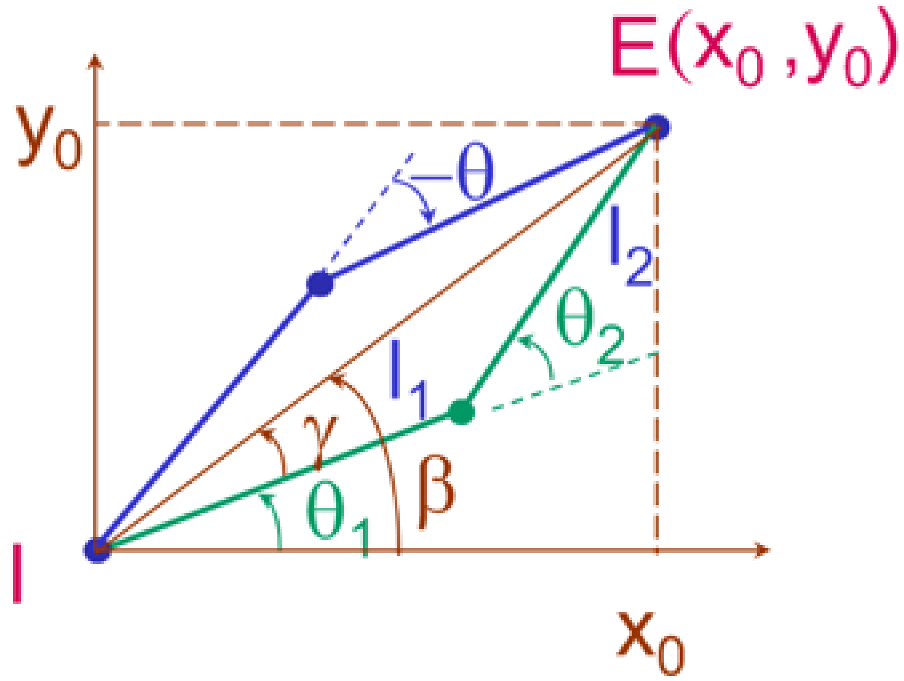
Closed form solutions

- Geometric or Algebra
  - Analytic solutions exist for a large class of mechanisms
    - 3 intersecting neighbouring axes (most industrial robots)



Closed form solutions

- Geometric or Algebraic
  - Analytic solutions exist for a large class of mechanisms
    - 3 intersecting neighbouring axes (most industrial robots)
- Geometric
  - Decompose spatial geometry of manipulator into several plane problems and apply geometric laws



Closed form solutions

- Geometric or Algebraic
  - Analytic solutions exist for a large class of mechanisms
    - 3 intersecting neighbouring axes (most industrial robots)
- Geometric
  - Decompose spatial geometry of manipulator into several plane problems and apply geometric laws
- Algebraic
  - Manipulate transformation equation to get the joint angles

In RD: focus on numerical methods

$$\mathbf{T}_{IE} = \mathbf{T}_{01}(\varphi_1)\mathbf{T}_{12}(\varphi_2)\mathbf{T}_{23}(\varphi_3)\mathbf{T}_{34}(\varphi_4)\mathbf{T}_{45}(\varphi_5)\mathbf{T}_{56}(\varphi_6)$$

$$\mathbf{T}_{01}(\varphi_1)^{-1}\mathbf{T}_{IE} = \mathbf{T}_{12}(\varphi_2)\mathbf{T}_{23}(\varphi_3)\mathbf{T}_{34}(\varphi_4)\mathbf{T}_{45}(\varphi_5)\mathbf{T}_{56}(\varphi_6)$$

$$\left(\mathbf{T}_{01}(\varphi_1)\mathbf{T}_{12}(\varphi_2)\right)^{-1}\mathbf{T}_{IE} = \mathbf{T}_{23}(\varphi_3)\mathbf{T}_{34}(\varphi_4)\mathbf{T}_{45}(\varphi_5)\mathbf{T}_{56}(\varphi_6)$$

$$\left(\mathbf{T}_{01}(\varphi_1)\mathbf{T}_{12}(\varphi_2)\mathbf{T}_{23}(\varphi_3)\right)^{-1}\mathbf{T}_{IE} = \mathbf{T}_{34}(\varphi_4)\mathbf{T}_{45}(\varphi_5)\mathbf{T}_{56}(\varphi_6)$$

Inverse Differential Kinematics

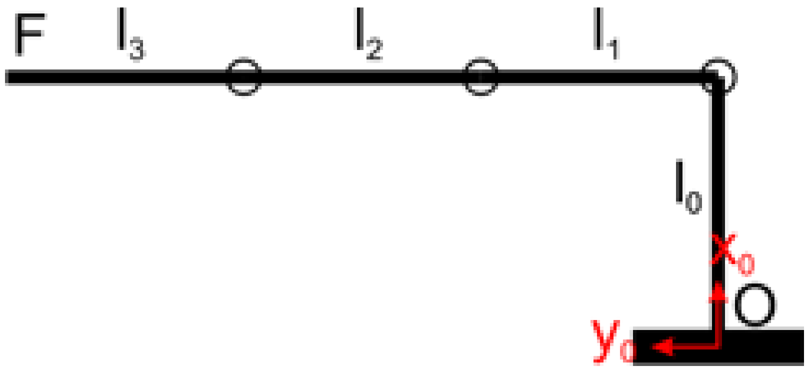
- We have seen how Jacobians map velocities from joint space to task-space
  - $\mathbf{w}_e = \mathbf{J}_{e0}\dot{\mathbf{q}}$
- In general, we are interested in the inverse problem
  - Simple method: use the pseudoinverse

    $$\dot{\mathbf{q}} = \mathbf{J}_{e0}^{+}\mathbf{w}_e^{*}$$

  - … however, the Jacobian might be singular!

Singularities

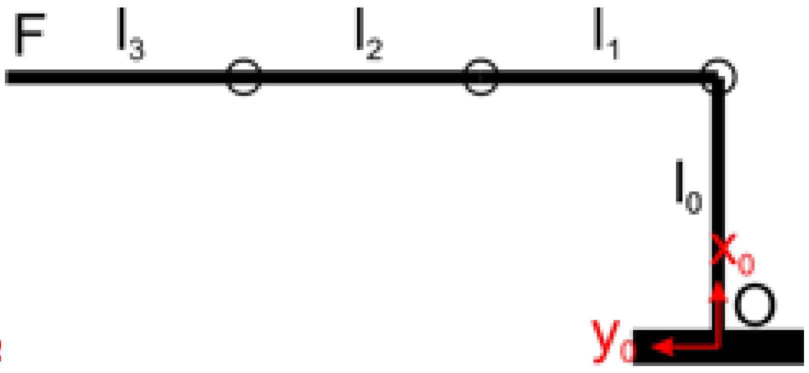
- A singularity is a joint-space configuration $\mathbf{q}_s$ such that $\mathbf{J}_{e0}(\mathbf{q}_s)$ is column-rank deficient
  - the Jacobian becomes badly conditioned
  - small desired velocities $\mathbf{w}_e^*$ produce high joint velocities $\dot{\mathbf{q}}$



- Singularities can be classified into:
  - boundary (e.g. a stretched out manipulator)
    - easy to avoid during motion planning
  - internal
    - harder to prevent, requires careful motion planning

Singularities

- A singularity is a joint-space configuration $\mathbf{q}_s$ such that $\mathbf{J}_{e0}(\mathbf{q}_s)$ is column-rank deficient
  - the Jacobian becomes badly conditioned
  - small desired velocities $\mathbf{w}_e^*$ produce high joint velocities $\dot{\mathbf{q}}$



- Use a damped version of the Moore-Penrose pseudo inverse

$$\dot{\mathbf{q}} = \mathbf{J}_{e0}^T(\mathbf{J}_{e0}\mathbf{J}_{e0}^T + \lambda^2\mathbf{I})^{-1}\mathbf{w}_e^* \qquad \min \quad \|\mathbf{w}_e^* - \mathbf{J}_{e0}\dot{\mathbf{q}}\|^2 + \lambda^2\|\dot{\mathbf{q}}\|^2$$

$$\lambda > 0, \lambda \in \mathbb{R}$$

Redundancy

- A kinematic structure is redundant if the dimension of the task-space is smaller than the dimension of the joint-space
  - E.g. the human arm has 7DoF (three in the shoulder, one in the elbow, and three in the wrist)
    - $\mathbf{q} \in \mathbb{R}^7$
    - $\mathbf{w} \in \mathbb{R}^6$
    - $\mathbf{J}_{e0} \in \mathbb{R}^{6\times 7}$

- Redundancy implies infinite solutions
  - $\dot{\mathbf{q}} = \mathbf{J}_{e0}^{+}\mathbf{w}_e^* + \mathbf{N}\dot{\mathbf{q}}_0$ $\quad\quad$ $\mathbf{N} = \mathcal{N}(\mathbf{J}_{e0})$ $\quad\quad$ $\mathbf{J}_{e0}(\mathbf{J}_{e0}^{+}\mathbf{w}_e^* + \mathbf{N}\dot{\mathbf{q}}_0) = \mathbf{w}_e^*$

    $$\mathbf{J}_{e0}\mathbf{N} = \mathbf{0}$$
  - One way to compute the nullspace projection matrix
    - $\mathbf{N} = \mathbf{I} - \mathbf{J}_{e0}^{+}\mathbf{J}_{e0}$

Multi-task control

- Manipulation (as well as locomotion!…) is a complex combination of high level tasks
  - track a desired position
  - ensure kinematic constraints
  - reach a desired end-effector orientation

- Break down the complexity into smaller tasks
  - Two methods
    - Multi-task with equal priority
    - Multi-task with Prioritization

$$task_i := \{\mathbf{J}_i, \mathbf{w}_i^*\}$$

Multi-task control
Equal priority

- Assume that *t* tasks have been defined
  - The generalised velocity is given by
  - $$\dot{q} = \underbrace{\begin{bmatrix} \mathbf{J}_1 \\ \vdots \\ \mathbf{J}_{n_t} \end{bmatrix}^+}_{\bar{\mathbf{J}}} \underbrace{\begin{pmatrix} \mathbf{w}_1^* \\ \vdots \\ \mathbf{w}_{n_t}^* \end{pmatrix}}_{\bar{\mathbf{w}}}$$

The pseudo inversion will try to solve all tasks at the same time in an optimal way

  - It is possible to weigh some tasks higher than others
    -

$$\bar{\mathbf{J}}^{+W} = \left(\bar{\mathbf{J}}^T \mathbf{W} \bar{\mathbf{J}}\right)^{-1} \bar{\mathbf{J}}^T \mathbf{W} \qquad \mathbf{W} = diag(w_1, \ldots, w_m)$$

Multi-task control
Prioritization

- Instead of solving all tasks at once, we can use consecutive nullspace projection to ensure a strict priority
- We already saw that $\dot{\mathbf{q}} = \mathbf{J}_1^+ \mathbf{w}_1^* + \mathbf{N}_1 \dot{\mathbf{q}}_0$
- The solution for task 2 should not violate the one found for task 1
  - $\mathbf{w}_2 = \mathbf{J}_2 \dot{\mathbf{q}} = \mathbf{J}_2 \left(\mathbf{J}_1^+ \mathbf{w}_1^* + \mathbf{N}_1 \dot{\mathbf{q}}_0 \right)$
  - This can be solved for $\dot{\mathbf{q}}_0$
  - Back substituting yields
  - The iterative solution for T tasks is then given by

$$\dot{\mathbf{q}}_0 = (\mathbf{J}_2 \mathbf{N}_1)^+ \left(\mathbf{w}_2^* - \mathbf{J}_2 \mathbf{J}_1^+ \mathbf{w}_1^* \right)$$

$$\dot{\mathbf{q}} = \mathbf{J}_1^+ \mathbf{w}_1^* + \mathbf{N}_1 (\mathbf{J}_2 \mathbf{N}_1)^+ \left(\mathbf{w}_2^* - \mathbf{J}_2 \mathbf{J}_1^+ \mathbf{w}_1^* \right)$$

$$\dot{\mathbf{q}} = \sum_{i=1}^{n_T} \mathbf{N}_i \dot{\mathbf{q}}_i, \quad \text{with} \quad \dot{\mathbf{q}}_i = (\mathbf{J}_i \mathbf{N}_i)^+ \left(\mathbf{w}_i^* - \mathbf{J} \sum_{k=1}^{i-1} \mathbf{N}_k \dot{\mathbf{q}}_k \right)$$

5

5

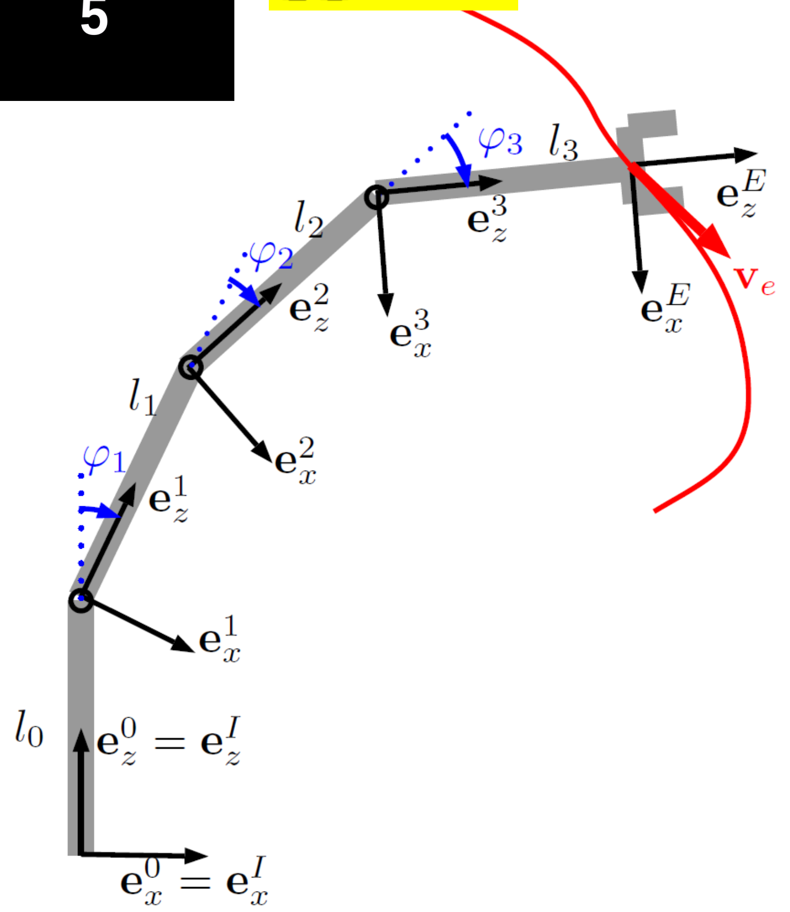
Quiz
3min
1min

Multi-task control
Example - single task

- 3DoF planar robot arm with unitary link lengths
  - Find the generalised velocities, given
    - $\mathbf{q}_t = (\pi/6, \pi/3, \pi/3)^T \qquad {}_0\dot{\mathbf{r}}^*_{E,t} = (1,1)^T$
    - $$\mathbf{r}_E = \begin{pmatrix} \sin(\varphi_1) + \sin(\varphi_1+\varphi_2) + \sin(\varphi_1+\varphi_2+\varphi_3) \\ 0 \\ 1 + \cos(\varphi_1) + \cos(\varphi_1+\varphi_2) + \cos(\varphi_1+\varphi_2+\varphi_3) \end{pmatrix} = \begin{pmatrix} s_1 + s_{12} + s_{123} \\ 0 \\ 1 + c_1 + c_{12} + c_{123} \end{pmatrix}$$

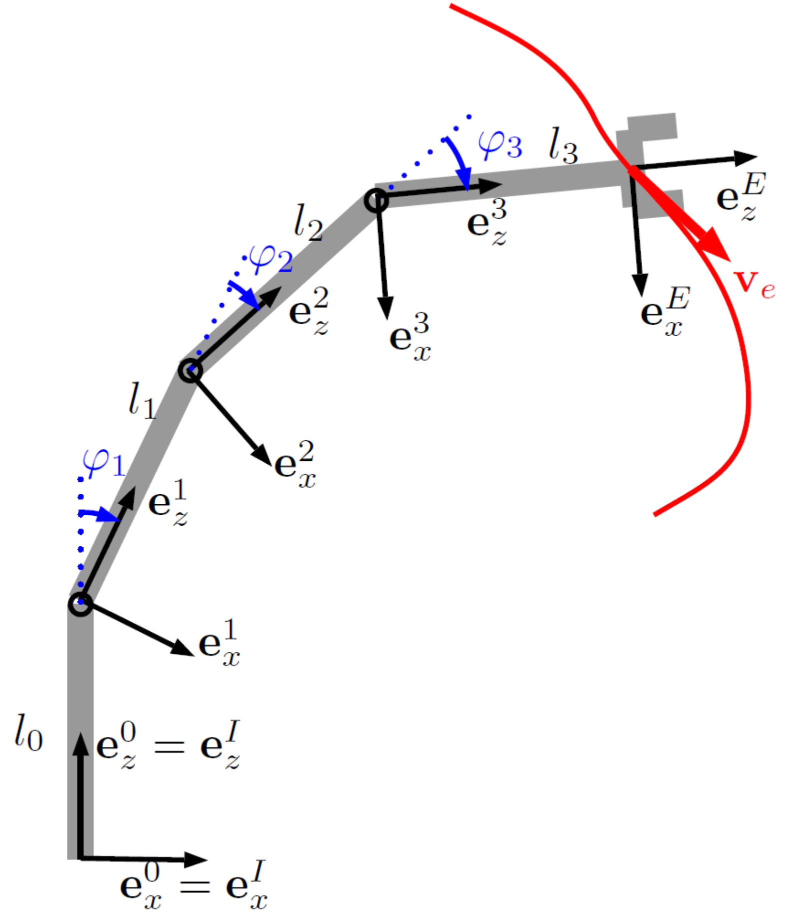
$$\mathbf{J}_E = \begin{bmatrix} +c_1 + c_{12} + c_{123} & +c_{12} + c_{123} & +c_{123} \\ 0 & 0 & 0 \\ -s_1 - s_{12} - s_{123} & -s_{12} - s_{123} & -s_{123} \end{bmatrix}$$

$$\mathbf{q}_t = (\pi/6, \pi/3, \pi/3)^T \quad \mathbf{J}_E = \frac{1}{2}\begin{bmatrix} +\sqrt{3} + 0 - \sqrt{3} & 0 - \sqrt{3} & -\sqrt{3} \\ 0 & 0 & 0 \\ -1-2-1 & -2-1 & -1 \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 0 & -\sqrt{3} & -\sqrt{3} \\ 0 & 0 & 0 \\ -4 & -3 & -1 \end{bmatrix}$$



Multi-task control
Example - single task

- 3DoF planar robot arm with unitary link lengths
  - Find the generalised velocities, given
    - $\mathbf{q}_t = (\pi/6, \pi/3, \pi/3)^T \qquad {}_0\dot{\mathbf{r}}^*_{E,t} = (1,1)^T$



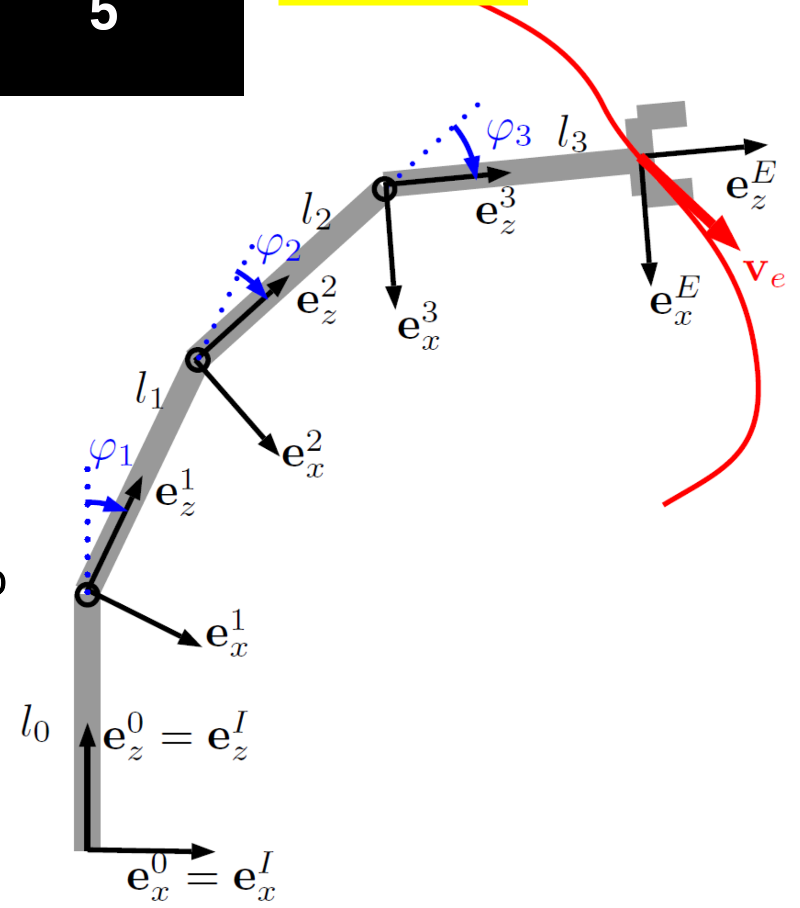
5

5

Quiz
3min
1min

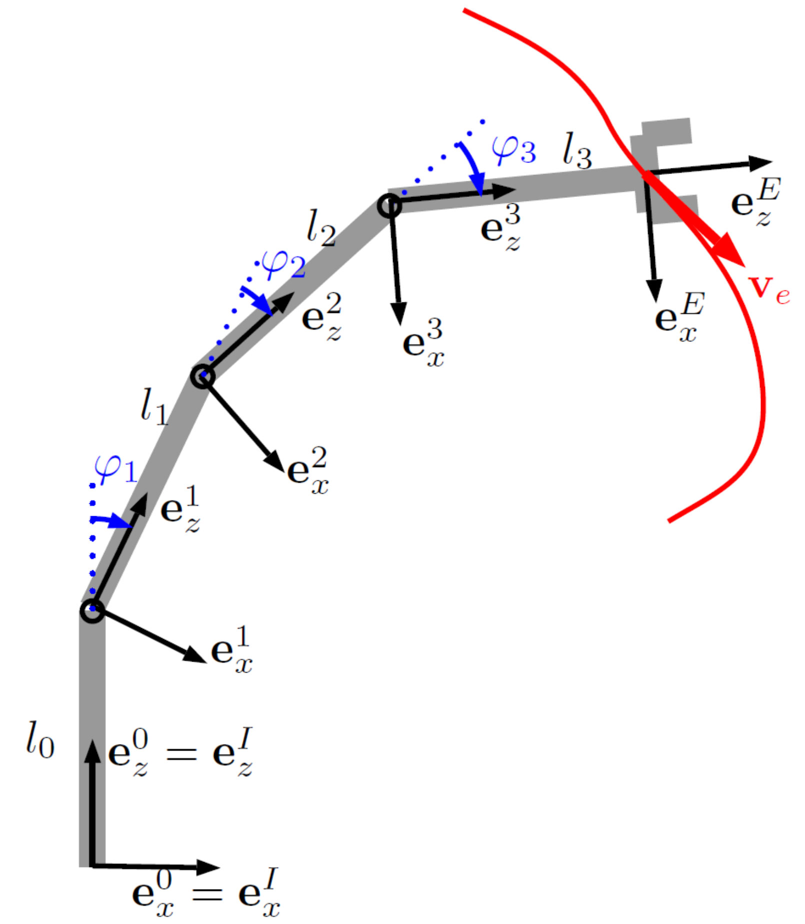
Multi-task control
Example - stacked task

- 3DoF planar robot arm with unitary link lengths
  - Find the generalised velocities, given
    - $\mathbf{q}_t = (\pi/6, \pi/3, \pi/3)^T \qquad {}_0\dot{\mathbf{r}}^*_{E,t} = (1,1)^T$
  - Additionally, we want to fulfill a second task with the same priority as the first, namely that the first and second joint velocities are zero
  -

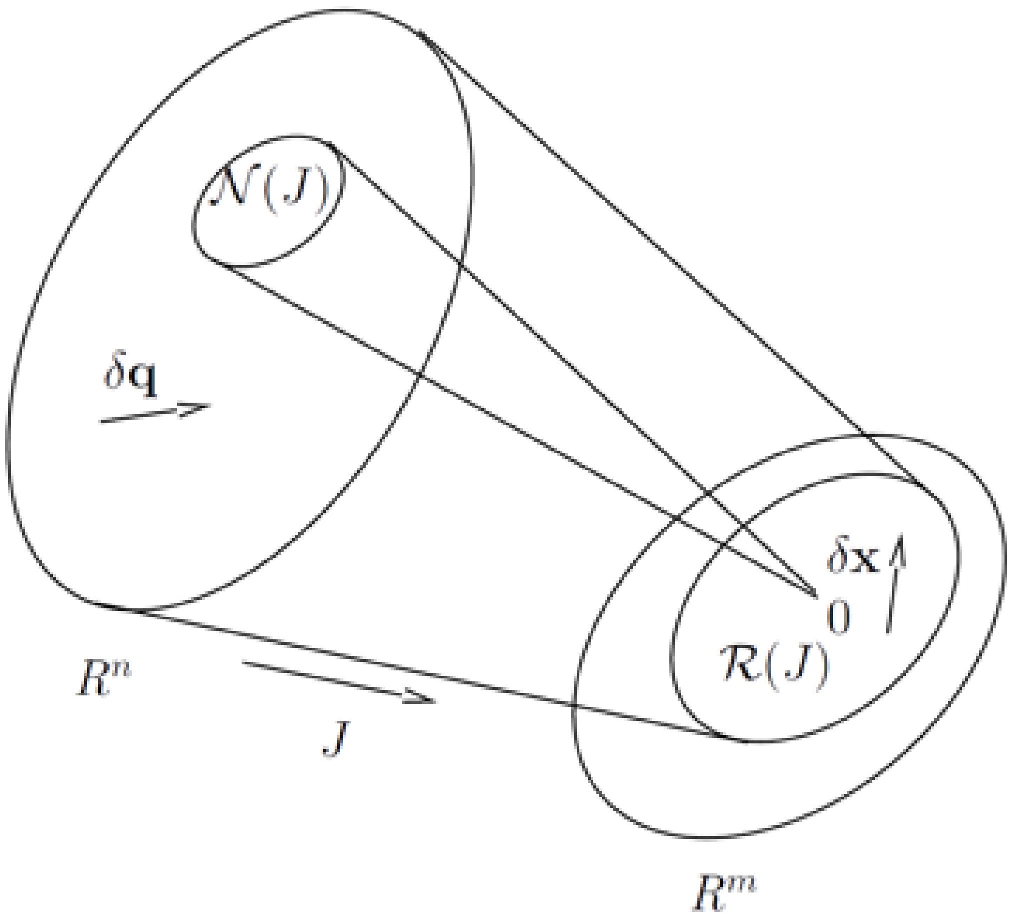


Multi-task control

Example - stacked task



Mapping associated with the Jacobian



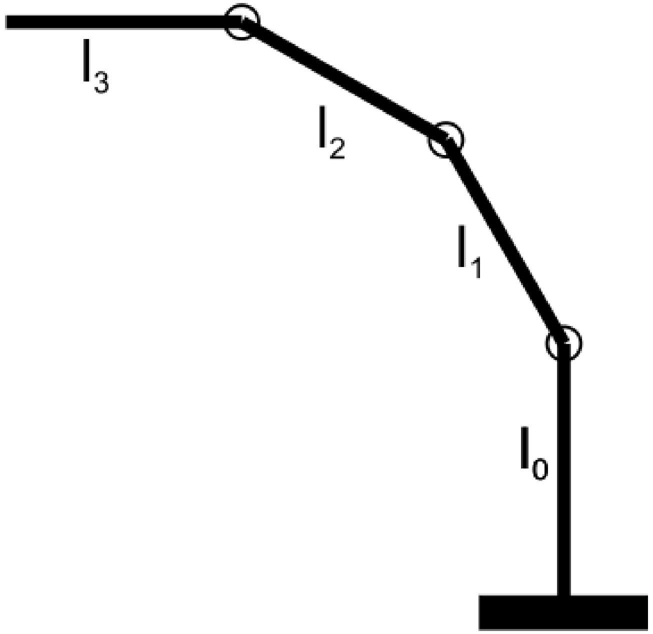
Numerical solutions
Inverse differential kinematics

- Jacobians map joint-space velocities to end-effector velocities
  - $\dot{\chi}_e = \mathbf{J}_{eA}(\mathbf{q})\dot{\mathbf{q}}$ $\quad\quad$ $\Delta\chi_e = \mathbf{J}_{eA}(\mathbf{q}) \cdot \Delta\mathbf{q}$
- We can use this to iteratively solve the inverse kinematics problem
  - target configuration χ_e^* , initial joint space guess $\mathbf{q}^0$

1. $\mathbf{q} \leftarrow \mathbf{q}^0$ ▷ start configuration
2. while $\|\chi_e^* - \chi_e(\mathbf{q})\| \geq$ tol do ▷ while the solution is not reached
3. $\mathbf{J}_{eA} \leftarrow \mathbf{J}_{eA}(\mathbf{q}) = \frac{\partial \chi_e}{\partial \mathbf{q}}(\mathbf{q})$ ▷ evaluate Jacobian
4. $\mathbf{J}_{eA}^+ \leftarrow (\mathbf{J}_{eA}(\mathbf{q}))^+$ ▷ compute the pseudo inverse
5. $\Delta\chi_e \leftarrow \chi_e^* - \chi_e(\mathbf{q})$ ▷ find the end-effector configuration error vector
6. $\mathbf{q} \leftarrow \mathbf{q} + \mathbf{J}_{eA}^+\Delta\chi_e$ ▷ updated the generalized coordinates

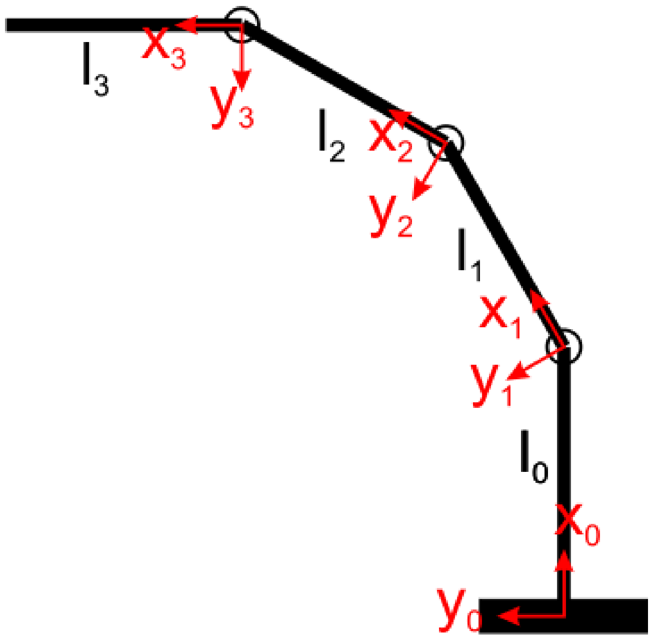
Inverse kinematics
Three-link arm example

- Determine end-effector Jacobian



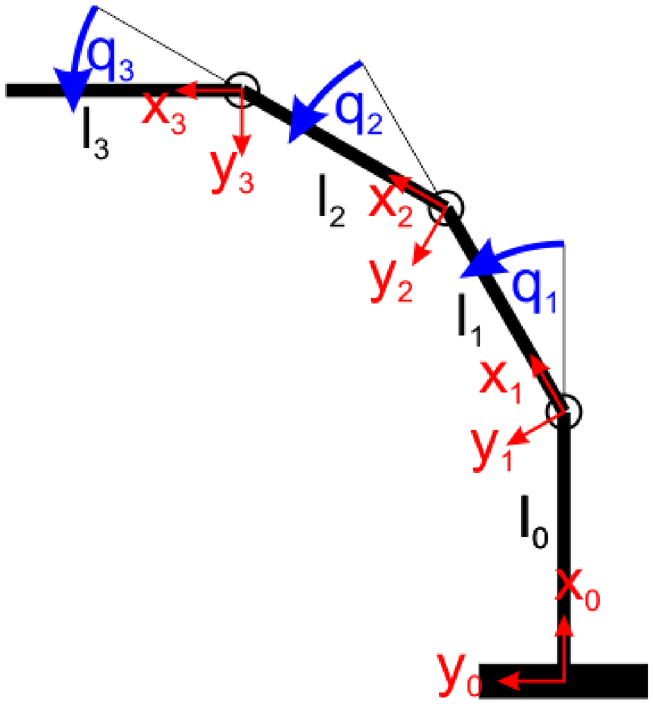
Inverse kinematics
Three-link arm example

- Determine end-effector Jacobian
  1. Introduce coordinate frames



Inverse kinematics
Three-link arm example

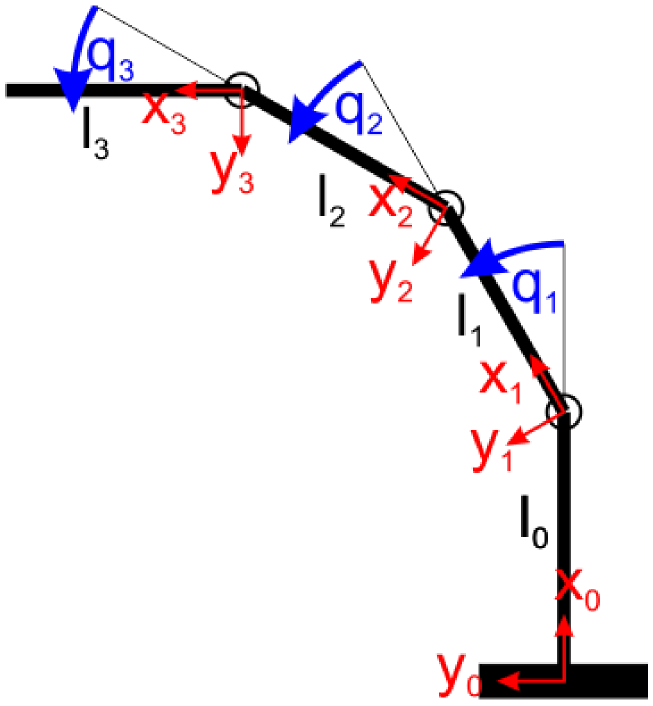
- Determine end-effector Jacobian
  1. Introduce coordinate frames
  2. Introduce generalized coordinates



Inverse kinematics
Three-link arm example

- Determine end-effector Jacobian
  1. Introduce coordinate frames
  2. Introduce generalized coordinates
  3. Determine end-effector position

$$
{}_0\mathbf{r}_{0E}(\mathbf{q}) = \begin{bmatrix} l_0 + l_1\cos(q_1) + l_2\cos(q_1+q_2) + l_3\cos(q_1+q_2+q_3) \\ l_1\sin(q_1) + l_2\sin(q_1+q_2) + l_3\sin(q_1+q_2+q_3) \\ 0 \end{bmatrix}
$$



Inverse kinematics
Three-link arm example

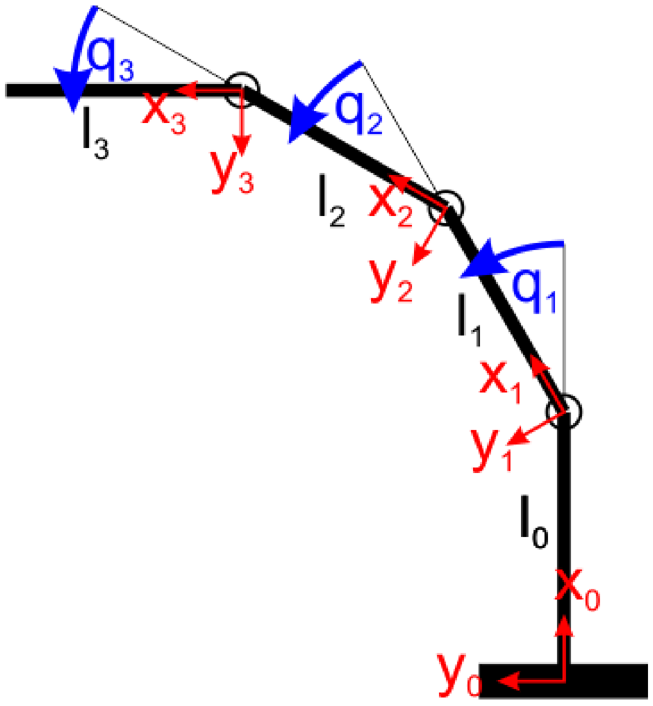
- Determine end-effector Jacobian
  1. Introduce coordinate frames
  2. Introduce generalized coordinates
  3. Determine end-effector position

$$ {}_0\mathbf{r}_{0E}(\mathbf{q}) = \begin{bmatrix} l_0 + l_1\cos(q_1) + l_2\cos(q_1+q_2) + l_3\cos(q_1+q_2+q_3) \\ l_1\sin(q_1) + l_2\sin(q_1+q_2) + l_3\sin(q_1+q_2+q_3) \\ 0 \end{bmatrix} $$

  4. Compute the Jacobian

$$ {}_0\mathbf{J}_{eP} = \frac{\partial}{\partial \mathbf{q}} {}_0\mathbf{r}_{0E}(\mathbf{q}) $$

$$ = \begin{bmatrix} -l_1\sin(q_1) - l_2\sin(q_1+q_2) - l_3\sin(q_1+q_2+q_3) & -l_2\sin(q_1+q_2) - l_3\sin(q_1+q_2+q_3) & -l_3\sin(q_1+q_2+q_3) \\ l_1\cos(q_1) + l_2\cos(q_1+q_2) + l_3\cos(q_1+q_2+q_3) & l_2\cos(q_1+q_2) + l_3\cos(q_1+q_2+q_3) & l_3\cos(q_1+q_2+q_3) \\ 0 & 0 & 0 \end{bmatrix} $$



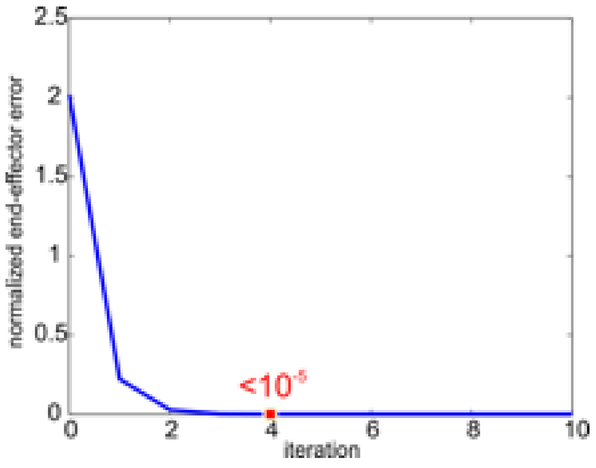
Inverse kinematics
Three-link arm example

- Iterative inverse kinematics to find desired configuration
  - $\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{J}_{eP}^{+}(\mathbf{r}_{goal} - \mathbf{r}^i)$

Inverse kinematics
Three-link arm example

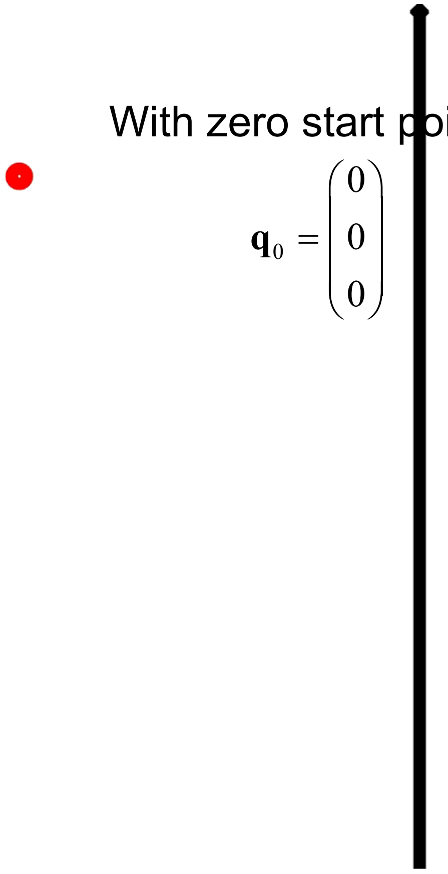
- Iterative inverse kinematics to find desired configuration
  - $\mathbf{q}^{i+1} = \mathbf{q}^{i} + \mathbf{J}_{eP}^{+}(\mathbf{r}_{goal} - \mathbf{r}^{i})$
  - start value

$$\mathbf{q}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$



With zero start point

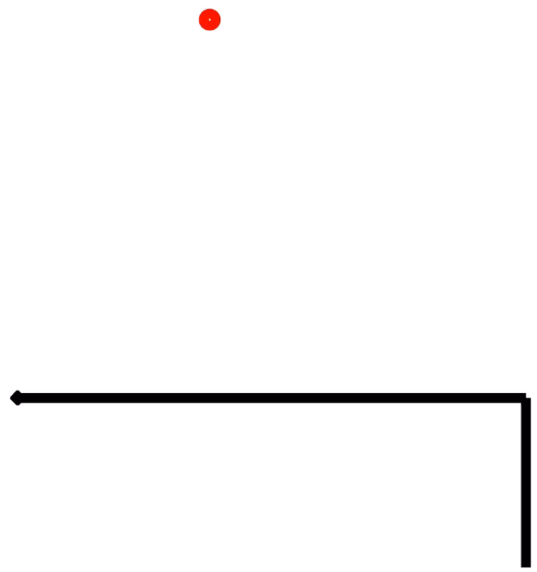
$$\mathbf{q}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$



Inverse kinematics
Three-link arm example

- Iterative inverse kinematics to find desired configuration
  - $\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{J}_{eP}^{+}(\mathbf{r}_{goal} - \mathbf{r}^i)$
  - start value

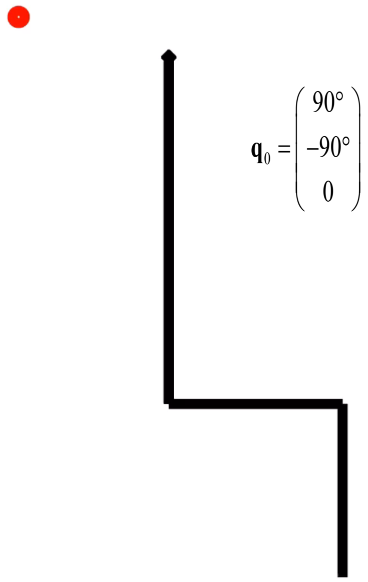
$$\mathbf{q}^0 = \begin{bmatrix} \pi/2 \\ 0 \\ 0 \end{bmatrix}$$



Inverse kinematics
Three-link arm example

- Iterative inverse kinematics to find desired configuration
  - $\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{J}_{eP}^{+}(\mathbf{r}_{goal} - \mathbf{r}^i)$
  - start value

$$\mathbf{q}^0 = \begin{bmatrix} \pi/2 \\ -\pi/2 \\ 0 \end{bmatrix}$$



- Same goal position, multiple solutions
  - joint-space bigger than task-space, redundant system

Inverse kinematics
Iterative methods

- Let's have a closer look at the joint update rule
  - $\mathbf{q}^{i+1} = \mathbf{q}^{i} + \mathbf{J}_{eA}^{+} \Delta \chi$
- Two main issues
  - Scaling
    - if the current error is too large, the error linearization implemented by the Jacobian is not accurate enough
    - use a scaling factor $0 < k < 1$ $\mathbf{q}^{i+1} = \mathbf{q}^{i} + k \mathbf{J}_{eA}^{+} \Delta \chi$
    - unfortunately, this will lead to slower convergence

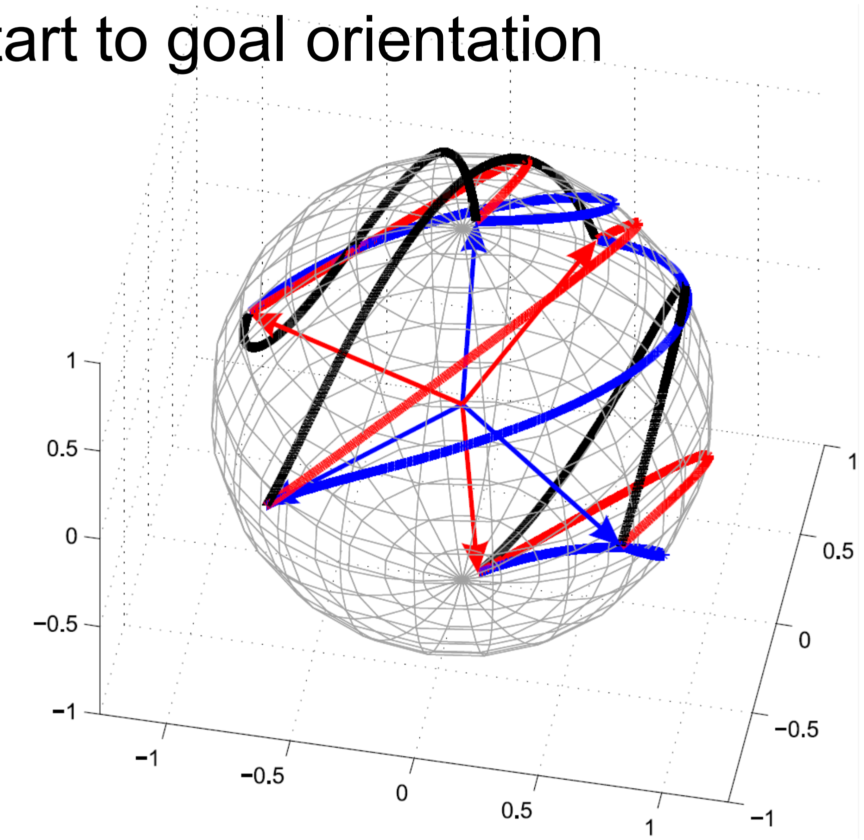
Inverse kinematics
Iterative methods

- Let's have a closer look at the joint update rule
  - $\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{J}_{e\mathcal{A}}^{+}\Delta\chi$
- Two main issues
  - Singular configurations
    - When the Jacobian is rank-deficient, the inversion becomes a badly conditioned problem
    - Use the damped pseudoinverse (Levenberg-Marquardt)
      - $\mathbf{q}^{i+1} = \mathbf{q}^i + \mathbf{J}_{e\mathcal{A}}^{T}(\mathbf{J}_{e\mathcal{A}}\mathbf{J}_{e\mathcal{A}}^{T} + \lambda^2\mathbf{I})^{-1}\Delta\chi$
    - Use the transpose of the Jacobian
      - $\mathbf{q}^{i+1} = \mathbf{q}^i + \alpha\mathbf{J}_{e\mathcal{A}}^{T}\Delta\chi$
    - For a detailed explanation, check "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods", **Samuel Buss,** 2009

Inverse differential kinematics
Orientation error

- 3D rotations are defined in the Special Orthogonal group SO(3)
- The parametrization affects convergence from start to goal orientation

$$\mathbf{q}^{i+1} = \mathbf{q}^{i} + \mathbf{J}_{eA}^{+}\Delta\chi$$

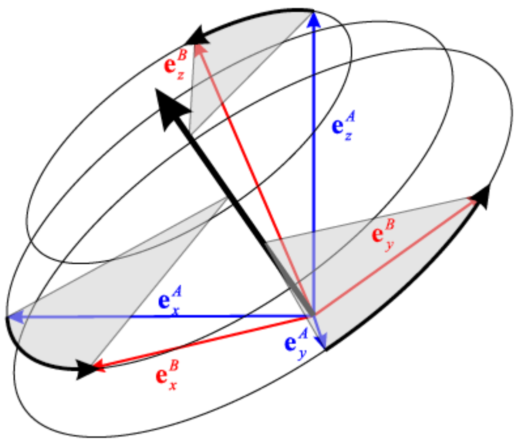


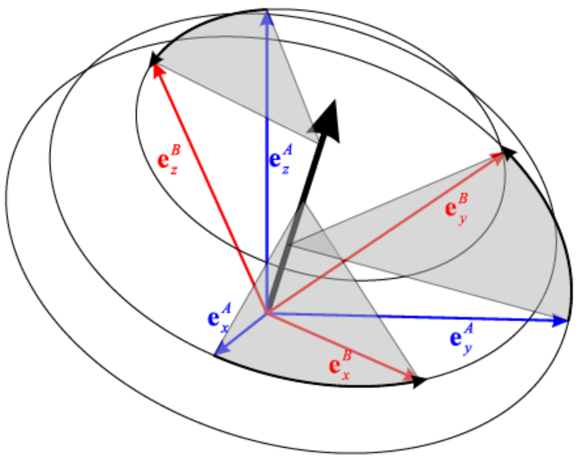
Inverse differential kinematics
Orientation error

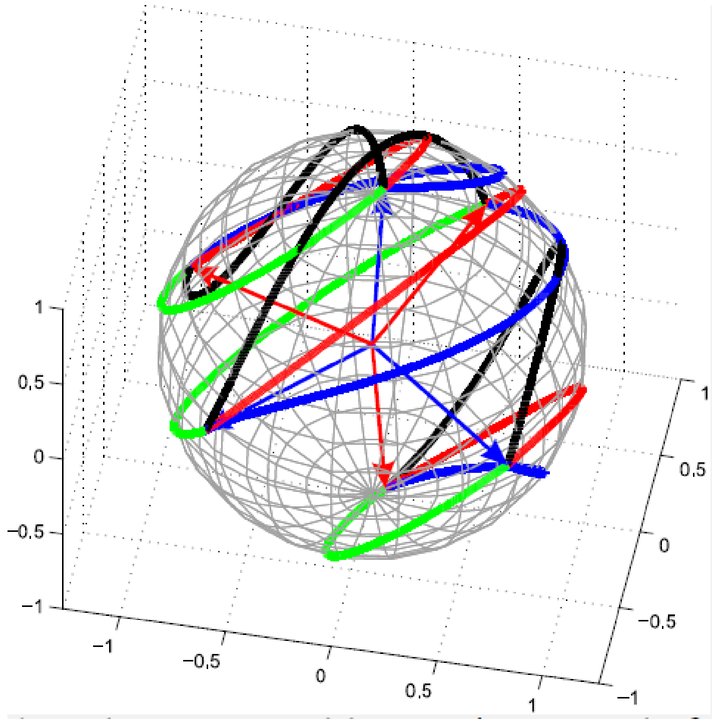
- 3D rotations are defined in the Special Orthogonal group SO(3)
- The parametrization affects convergence from start to goal orientation
  - Rotate along shortest path in SO(3): use rotational vectors which parametrize rotation from start to goal
    - $$\Delta\chi_{rotvec} = \Delta\varphi \qquad \Longrightarrow \quad \mathbf{C}_{\mathcal{GS}}(\Delta\varphi) = \mathbf{C}_{\mathcal{GI}}(\varphi^*)\mathbf{C}_{\mathcal{SI}}^T(\varphi^t)$$
  - The update law for rotations will then be
    - $$\mathbf{q} \leftarrow \mathbf{q} + k_{PR}\mathbf{J}_{e0_R}^{+}\Delta\varphi$$

This is NOT the difference between rotation vectors, but the rotation vector extracted from the relative rotation between start and goal

Rotation with rotation vector and angle







Trajectory control
Position

- Consider a planned desired motion of the end effector
  - $\mathbf{r}_e^*(t)$
    $\dot{\mathbf{r}}_e^*(t)$
- Let’s see how to kinematically control the end-effector position
  - Feedback term
    - $\Delta\mathbf{r}_e^t = \mathbf{r}_e^*(t) - \mathbf{r}_e(\mathbf{q}^t)$
  - We can design a nonlinear stabilizing controller law
    - $\dot{\mathbf{q}}^* = \mathbf{J}_{e0_P}^+(\mathbf{q}^t) \cdot \left(\dot{\mathbf{r}}_e^*(t) + k_{pP}\Delta\mathbf{r}_e^t\right)$

Trajectory control
Orientation

- Derivation more involved
- Final control law similar to the position case

$$\dot{\mathbf{q}} = \mathbf{J}_{e0_R}^{+}\left(\omega(t)_e^* + k_{PR}\Delta\varphi\right)$$

Note that we are not using the analytical Jacobian since we are dealing with angular velocities and rotational vectors

ETH zürich

Floating Base Kinematics

151-0851-00 V

lecture:	CAB G11	Tuesday 10:15 – 12:00, every week
exercise:	HG E1.2	Wednesday 8:15 – 10:00, according to schedule (about every 2nd week)
office hour:	LEE H303	Friday 12.15 – 13.00

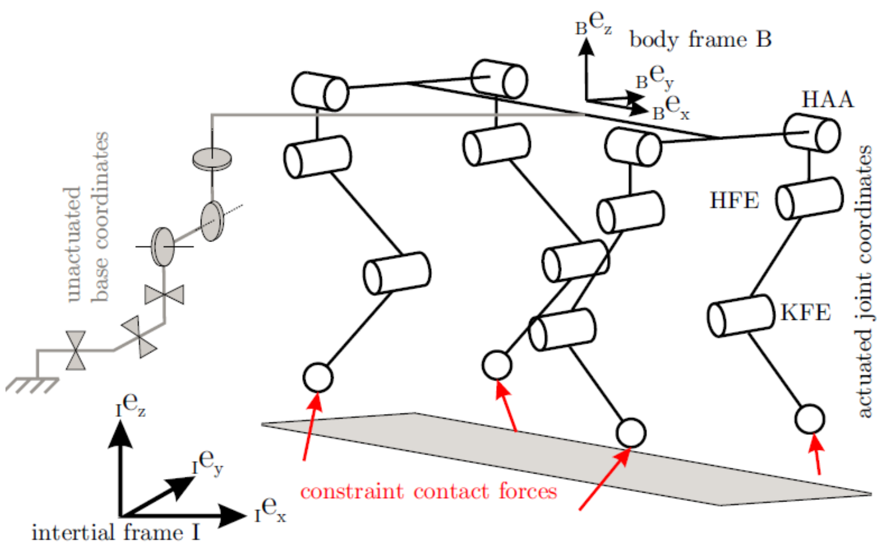
Marco Hutter, Roland Siegwart, and Thomas Stastny

Floating Base Systems
Kinematics

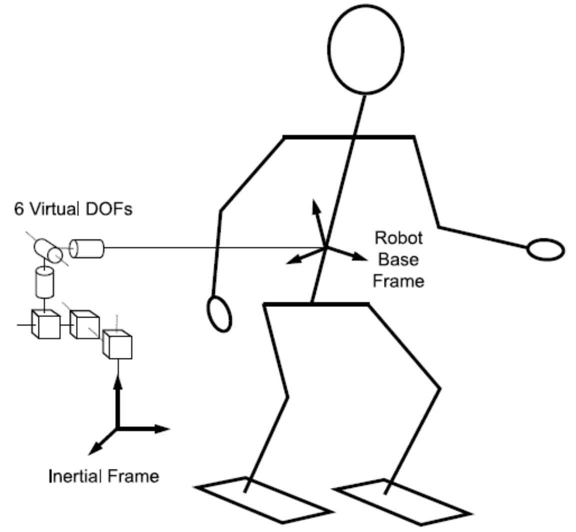
- Generalized coordinates

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_b \\ \mathbf{q}_j \end{pmatrix} \quad \text{with} \quad \mathbf{q}_b = \begin{pmatrix} \mathbf{q}_{b_P} \\ \mathbf{q}_{b_R} \end{pmatrix} \in \mathbb{R}^3 \times SO(3)$$

- Generalized velocities and accelerations?
  - Time derivatives $\dot{\mathbf{q}}, \ddot{\mathbf{q}}$ depend on parameterization
  - Often $\mathbf{u} = \begin{pmatrix} {}_I\mathbf{v}_B \\ {}_{\boxed{B}}\boldsymbol{\omega}_{IB} \\ \dot{\varphi}_1 \\ \vdots \\ \dot{\varphi}_{n_j} \end{pmatrix} \in \mathbb{R}^{6+n_j} = \mathbb{R}^{n_u} \qquad \dot{\mathbf{u}} = \begin{pmatrix} {}_I\mathbf{a}_B \\ {}_{\boxed{B}}\boldsymbol{\psi}_{IB} \\ \ddot{\varphi}_1 \\ \vdots \\ \ddot{\varphi}_{n_j} \end{pmatrix} \in \mathbb{R}^{6+n_j}$
  - Linear mapping $\mathbf{u} = \mathbf{E}_{fb} \cdot \dot{\mathbf{q}}$, with $\mathbf{E}_{fb} = \begin{bmatrix} \mathbb{I}_{3\times3} & 0 & 0 \\ 0 & \mathbf{E}_{\chi_R} & 0 \\ 0 & 0 & \mathbb{I}_{n_j \times n_j} \end{bmatrix}$



(a) Quadruped



(b) Humanoid

Floating Base Systems
Differential kinematics

- Position of an arbitrary point on the robot

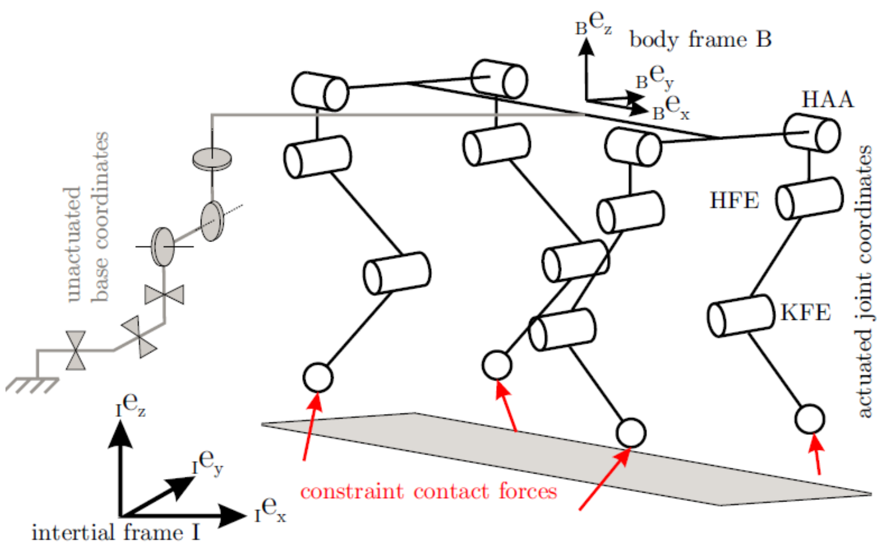
$$
{}_{\mathcal{I}}\mathbf{r}_{IQ}(\mathbf{q}) = \boxed{{}_{\mathcal{I}}\mathbf{r}_{IB}(\mathbf{q})} + \boxed{\mathbf{C}_{\mathcal{IB}}(\mathbf{q})} \cdot \boxed{{}_{\mathcal{B}}\mathbf{r}_{BQ}(\mathbf{q})}
$$

$$
{}_{\mathcal{I}}\mathbf{r}_{IB}(\mathbf{q}_b) \qquad \mathbf{C}_{\mathcal{IB}}(\mathbf{q}_b) \qquad {}_{\mathcal{B}}\mathbf{r}_{BQ}(\mathbf{q}_j)
$$

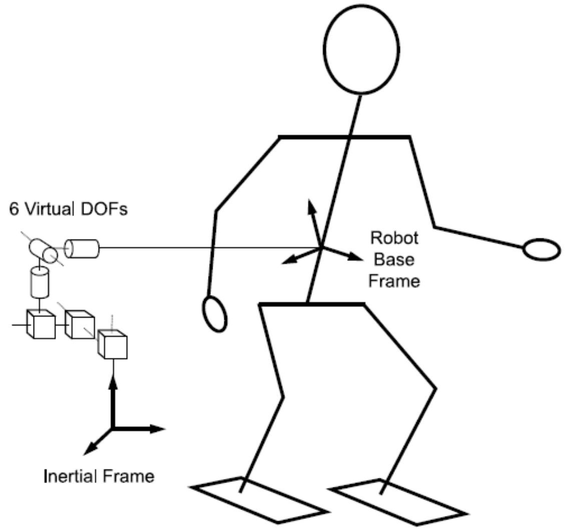
- Velocity of this point

$$
\begin{aligned}
{}_{\mathcal{I}}\mathbf{v}_Q &= {}_{\mathcal{I}}\mathbf{v}_B + \dot{\mathbf{C}}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{r}_{BQ} + \mathbf{C}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\dot{\mathbf{r}}_{BQ} \\
&= {}_{\mathcal{I}}\mathbf{v}_B + \mathbf{C}_{\mathcal{IB}} \cdot [{}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}]_\times \cdot {}_{\mathcal{B}}\mathbf{r}_{BQ} + \mathbf{C}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\dot{\mathbf{r}}_{BQ} \\
&= {}_{\mathcal{I}}\mathbf{v}_B - \mathbf{C}_{\mathcal{IB}} \cdot [{}_{\mathcal{B}}\mathbf{r}_{BQ}]_\times \cdot {}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}} + \mathbf{C}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\dot{\mathbf{r}}_{BQ} \\
&= {}_{\mathcal{I}}\mathbf{v}_B - \mathbf{C}_{\mathcal{IB}} \cdot [{}_{\mathcal{B}}\mathbf{r}_{BQ}]_\times \cdot {}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}} + \mathbf{C}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{J}_{P_{q_j}}(\mathbf{q}_j) \cdot \dot{\mathbf{q}}_j \\
&= \underbrace{\boxed{\begin{bmatrix} \mathbb{I}_{3\times 3} & -\mathbf{C}_{\mathcal{IB}} \cdot [{}_{\mathcal{B}}\mathbf{r}_{BQ}]_\times & \mathbf{C}_{\mathcal{IB}} \cdot {}_{\mathcal{B}}\mathbf{J}_{P_{q_j}}(\mathbf{q}_j) \end{bmatrix}}}_{{}_{\mathcal{I}}\mathbf{J}_Q(\mathbf{q})} \cdot \mathbf{u} \quad \text{with} \quad \mathbf{u} = \begin{pmatrix} {}_{I}\mathbf{v}_B \\ {}_{B}\boldsymbol{\omega}_{IB} \\ \dot{\varphi}_1 \\ \vdots \\ \dot{\varphi}_{n_j} \end{pmatrix}
\end{aligned}
$$

$$
\begin{aligned}
[{}_{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}]_\times &= \mathbf{C}_{\mathcal{BI}} [{}_{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}}]_\times \mathbf{C}_{\mathcal{BI}}^T \\
&= \mathbf{C}_{\mathcal{IB}}^T \dot{\mathbf{C}}_{\mathcal{IB}} \mathbf{C}_{\mathcal{IB}}^T \mathbf{C}_{\mathcal{IB}} = \mathbf{C}_{\mathcal{IB}}^T \dot{\mathbf{C}}_{\mathcal{IB}}
\end{aligned}
$$



(a) Quadruped



(b) Humanoid

Contact Constraints

- A contact point C_i is not allowed to move:

$$_{\mathcal{I}}\mathbf{r}_{IC_i} = const, \quad _{\mathcal{I}}\dot{\mathbf{r}}_{IC_i} = {_{\mathcal{I}}\ddot{\mathbf{r}}_{IC_i}} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

- Constraint as a function of generalized coordinates:

$$_{\mathcal{I}}\mathbf{J}_{C_i}\mathbf{u} = \mathbf{0}, \qquad _{\mathcal{I}}\mathbf{J}_{C_i}\dot{\mathbf{u}} + {_{\mathcal{I}}\dot{\mathbf{J}}_{C_i}}\mathbf{u} = \mathbf{0}$$

- Stack of constraints

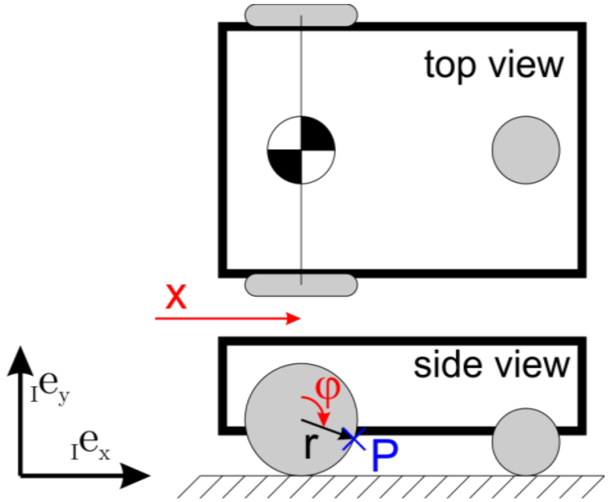
$$\mathbf{J}_c = \begin{bmatrix} \mathbf{J}_{C_1} \\ \vdots \\ \mathbf{J}_{C_{n_c}} \end{bmatrix} \in \mathbb{R}^{3n_c \times n_n}$$

Contact Constraint
Wheeled vehicle simple example

- Contact constraints
  - Point on wheel $${}_{\mathcal{I}}\mathbf{r}_{IP} = \begin{pmatrix} x + r\sin(\varphi) \\ r + r\cos(\varphi) \\ 0 \end{pmatrix}$$
  - Jacobian $${}_{\mathcal{I}}\mathbf{J}_P = \begin{bmatrix} 1 & r\cos(\varphi) \\ 0 & -r\sin(\varphi) \\ 0 & 0 \end{bmatrix}$$
  - Contact constraints

$${}_{\mathcal{I}}\dot{\mathbf{r}}_{IP}\big|_{\varphi=\pi} = {}_{\mathcal{I}}\mathbf{J}_P\big|_{\varphi=\pi}\dot{\mathbf{q}} = \begin{bmatrix} 1 & -r \\ 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{pmatrix} \dot{x} \\ \dot{\varphi} \end{pmatrix} = \mathbf{0}$$

=> Rolling condition $\dot{x} - r\dot{\varphi} = 0$



$$\mathbf{q} = \begin{pmatrix} x \\ \varphi \end{pmatrix}$$

Un-actuated base

Actuated joints

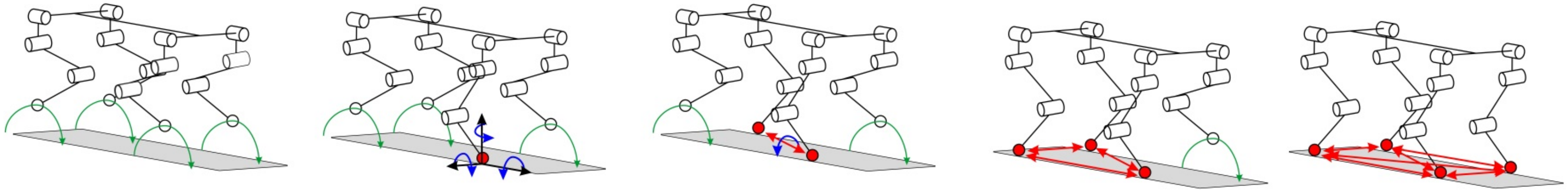
Properties of Contact Jacobian

- Contact Jacobian tells us, how a system can move.
  - Separate stacked Jacobian $\mathbf{J}_c = \begin{bmatrix} \boxed{\mathbf{J}_{c,b}} & \mathbf{J}_{c,j} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{r}_c}{\partial \mathbf{q}_b} & \frac{\partial \mathbf{r}_c}{\partial \mathbf{q}_j} \end{bmatrix} \in \mathbb{R}^{n_c \times (n_b + n_j)}$

    relation between base motion and constraints
  - Base is fully controllable if $rank(\mathbf{J}_{c,b}) = 6$
  - Nr of kinematic constraints for joint actuators: $rank(\mathbf{J}_c)$ - $rank(\mathbf{J}_{c,b})$
- Generalized coordinates DON'T correspond to the degrees of freedom
  - Contact constraints!
- Minimal coordinates (= correspond to degrees of freedom)
  - Require to switch the set of coordinates depending on contact state (=> never used)

Quadrupedal Robot with Point Feet

- Floating base system with 12 actuated joint and 6 base coordinates (18DoF)



Total constraints	0	3	6	9	12
Internal constraints	0	0	1	3	6
Uncontrollable DoFs	6	3	1	0	0