Universitat de Girona

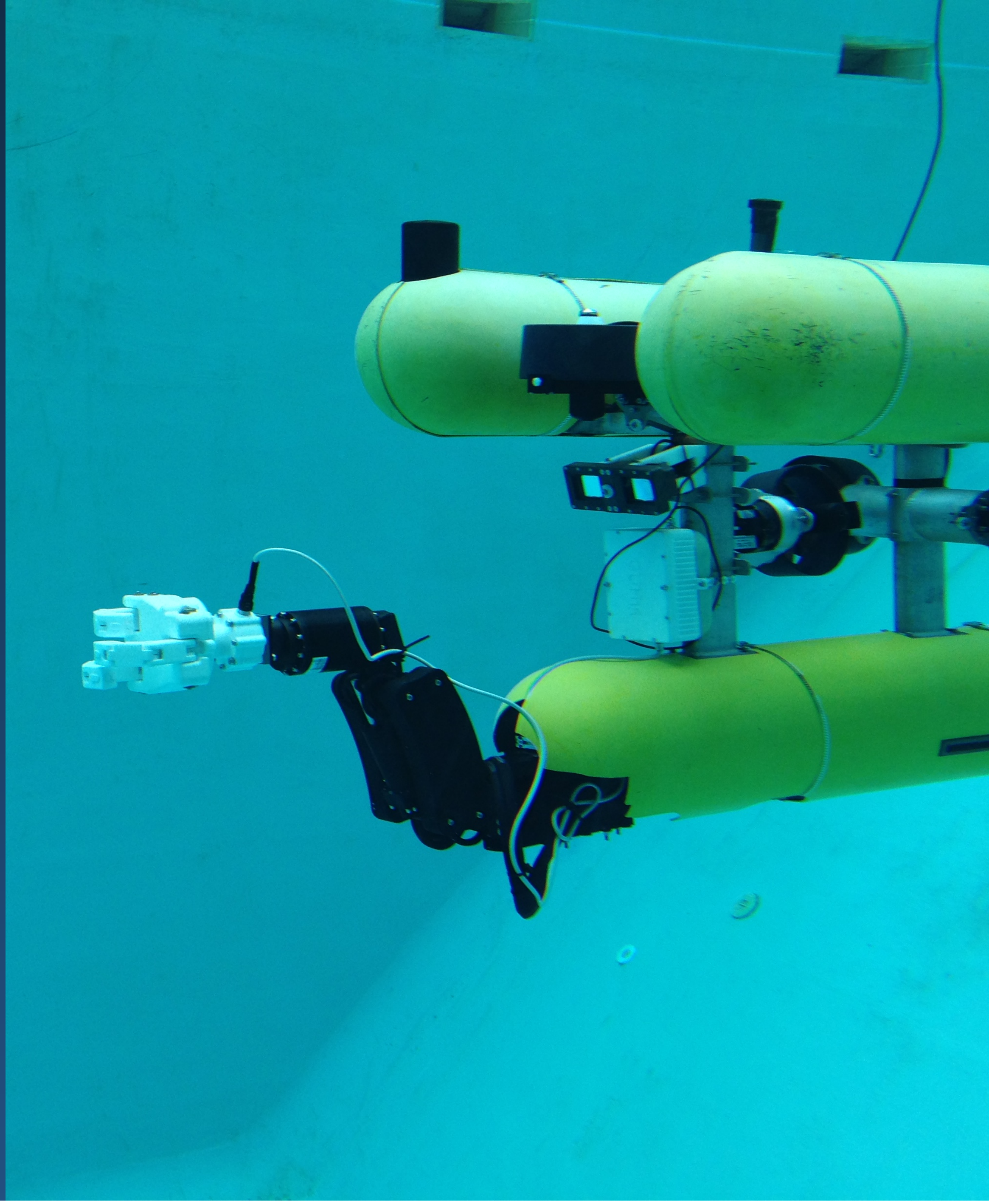
IFRoS MIRS

HANDS-ON INTERVENTION:

Vehicle-Manipulator Systems

Patryk Cieślak

patryk.cieslak@udg.edu



Lecture 2: Resolved-rate motion control

Contents

1. Robot control

Robot control

Kinematic

- Position and velocity control based on a kinematic model.
- The output of the controller is the desired velocity.
- The velocity has to be followed by the low level controllers (often PID).
- Due to lack of knowledge about system dynamics the requested velocities/accelerations may not be possible to achieve y the system.
- Requires high gain velocity controllers which results in a stiff system, susceptible to oscillations and overshoot.

Dynamic

- Position and velocity control based on a dynamic model.
- The output of the controller is the desired torque.
- The torque can be generated in open loop or in using current/torque feedback.
- Knowledge of dynamics allows for generating adequate inputs, based on the current state of the system, which results in smooth, efficient motion, fast motion.
- Since the feedback of the low level controllers is on the current/torque level the system is less susceptible to oscillations.

Force

- Control which takes into account forces and torques exerted on the environment.
- Force control can be based both on kinematic and dynamic models.
- The environment is modelled as soft, i.e., as a spring-damped system.
- Torques can be measured on each of the joints or in the end-effector.
- Allows for compliant operation which is necessary in cooperation with humans as well as manipulation in uncertain environment, i.e., it ensures safety and protection from damage.

Impedance | **Admittance** | **Hybrid position/force**

Kinematics

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

Jacobian

*Jacobian is continuously computed for the current configuration to **locally linearise** the system **kinematics**.*

$$\dot{x}_E = \left[v_E^T \ \omega_E^T\right]^T$$ Cartesian end-effector twist (linear and angular velocities)

$$\mathbf{q} = \left[\eta^T \ q^T\right]^T$$ System configuration vector (positions of all system DOF)

$$\zeta = \left[\nu^T \ \dot{q}^T\right]^T$$ Quasi-velocities (velocities of all system DOF)

$$\eta = \left[\eta_1 \ \eta_2\right]^T = \left[x \ y \ z \mid \phi \ \theta \ \psi\right]^T$$ Pose - Cartesian position & orientation (RPY angles)

Problem

Find a **vector of quasi-velocities** which will drive the system's end-effector at the **desired (Cartesian) velocity**.

Possible solution: open-loop control

Vector of quasi-velocities
to be applied to the system

Jacobian inverse (only square matrices)

Analytic solution

$$\zeta = J^{-1}(\mathbf{q})\dot{x}_E$$

Actual realisation
(digital control system)

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + J^{-1}(\mathbf{q}(t_k))\dot{x}_E(t_k)\Delta t$$

Due to the limited sampling time of digital control systems and numerical errors, this solution results in **drift of the solution** in the operational space. The planned trajectory is not possible to track and the error is growing with time.

Solution: resolved-rate motion control

$$\zeta = J^{-1}(\mathbf{q})\dot{x}_E$$

Feedforward action (tracking)

Gain (diagonal)

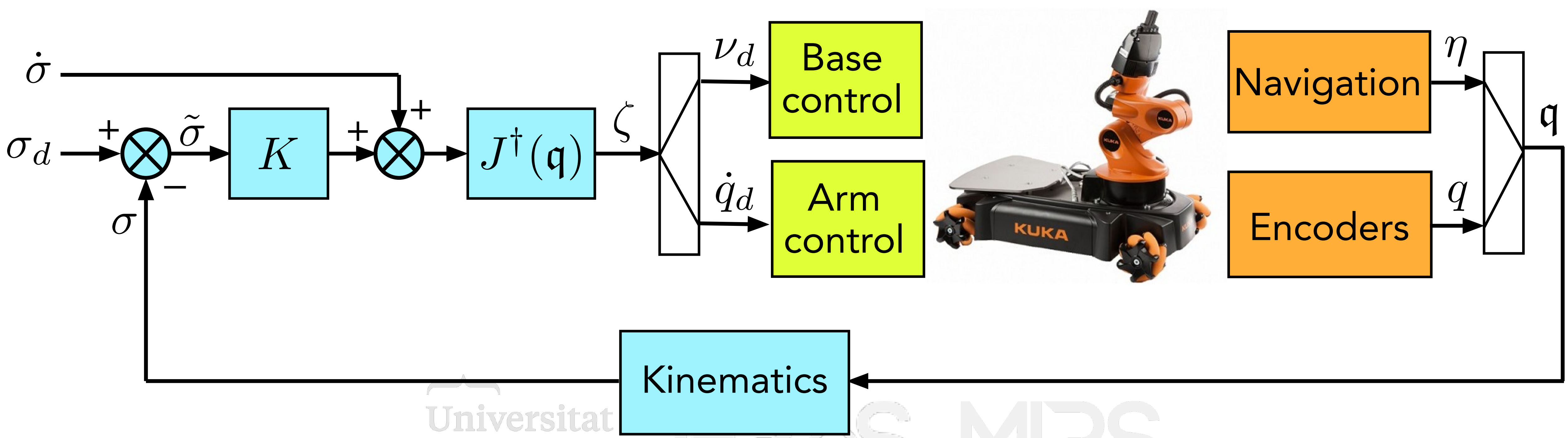
Goal (desired end-effector pose)

$$\dot{x}_E = \dot{\sigma}_E + K\tilde{\sigma}_E \qquad \tilde{\sigma}_E = \sigma_{E,d} - \sigma_E$$

Cartesian twist needed to move the end-effector in the direction of the goal

Feedback action (regulation)

Current value (current end-effector pose)



Solution: resolved-rate motion control

Control law $$\zeta = J^{-1}(\mathbf{q})(\dot{\sigma}_E + K\tilde{\sigma}_E)$$

$$e = \tilde{\sigma}_E = \sigma_{E,d} - \sigma_E$$

$$\dot{e} = \dot{\sigma}_E - J(\mathbf{q})\zeta$$

$$\dot{e} = \dot{\sigma}_E - J(\mathbf{q})J^{-1}(\mathbf{q})(\dot{\sigma}_E + Ke)$$

System error dynamics $$\boxed{\dot{e} = -Ke}$$

Lyapunov's second method of stability

For $\dot{x} = f(x)$, with an equilibrium at $x = 0$, consider a function $V : \mathbb{R}^n \rightarrow \mathbb{R}$, such that:

1. $V(x) = 0$ if and only if $x = 0$.
2. $V(x) > 0$ if and only if $x \neq 0$.
3. $\dot{V}(x) = \frac{d}{dt}V(x) = \sum_{i=1}^{n} \frac{\partial V}{\partial x_i} f_i(x) = \nabla V \cdot f(x) \leq 0$ for all values of $x \neq 0$.

Then $V(x)$ if called the **Lyapunov function** and the system is **stable in the Lyapunov sense**.

Moreover, if $\boxed{\dot{V}(x) < 0}$ for all $x \neq 0$, then the system is **asymptotically stable**.

Proof for resolved-rate motion control

$V(e) = \frac{1}{2}e^2$ Lyapunov function

$\dot{V}(e) = e\dot{e} = e(-Ke) = -Ke^2$

From error dynamics

System is asymptotically stable if $K > 0$.

Analytical Jacobian

$p = p(q)$ Position

$\phi = \phi(q)$ Orientation (e.g. RPY angles)

$$\dot{p} = \frac{\partial p}{\partial q}\dot{q} = J_p(q)\dot{q}$$

$$\dot{\phi} = \frac{\partial \phi}{\partial q}\dot{q} = J_\phi(q)\dot{q}$$

$$\dot{x} = \begin{bmatrix} \dot{p} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} J_p(q) \\ J_\phi(q) \end{bmatrix} \dot{q} = J_A(q)\dot{q}$$

Differential quantities in the operational space

Geometrical Jacobian

$$T_n^0(q) = \begin{bmatrix} R_n^0(q) & O_n^0(q) \\ 0 & 1 \end{bmatrix}$$ Robot kinematics

Linear velocity

$$\begin{bmatrix} v_n^0 \\ \omega_n^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q} = J_n^0(q)\dot{q}$$

Angular velocity

Quantities of clear physical meaning

Denavit-Hartenberg

$$T_n^{n-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\sin(\alpha_i) & 0 \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

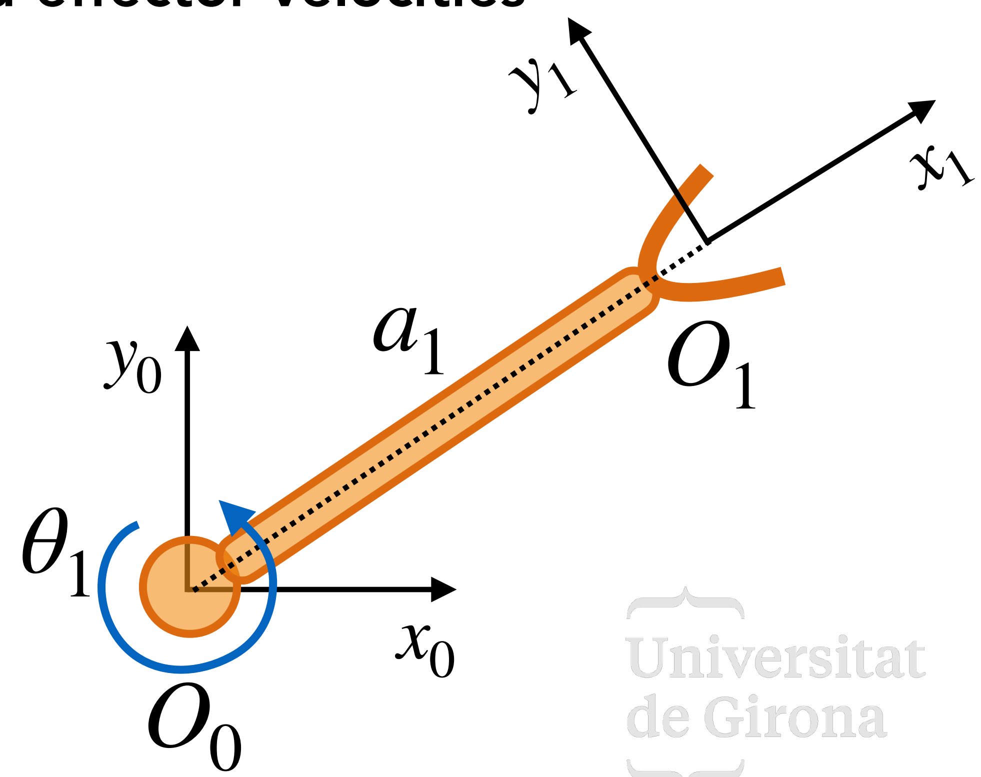
$$T_n^0(q) = T_1^0\, T_2^1\, T_3^2 \dots T_n^{n-1} = \begin{bmatrix} R_n^0(q) & O_n^0(q) \\ 0 & 1 \end{bmatrix}$$

Configuration vector (joint positions including all **prismatic and revolute** joints)

$$R = \begin{bmatrix} x_x & y_x & z_x \\ x_y & y_y & z_y \\ x_z & y_z & z_z \end{bmatrix}$$

Joint axis

End-effector velocities



Revolute joint

$$v_1 = \omega_1 \times (O_1 - O_0)$$

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} = \dot{\theta}_1 z_0$$

Prismatic joint

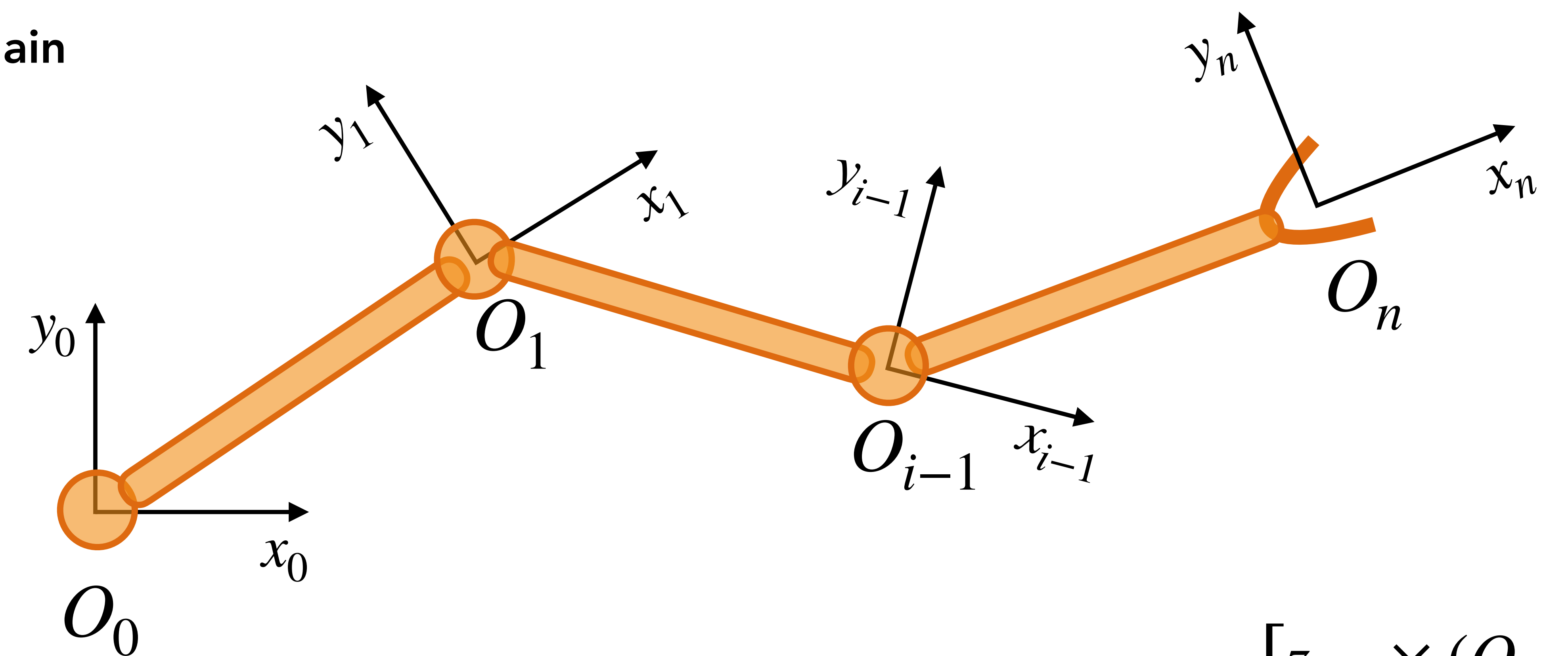
$$v_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{d}_1 \end{bmatrix} = \dot{d}_1 z_0$$

$$\omega_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Jacobian (1 revolute joint)

$$\begin{bmatrix} v_1^0 \\ \omega_1^0 \end{bmatrix} = \begin{bmatrix} J_v \\ J_\omega \end{bmatrix} \dot{q} = J_1^0(q)\dot{q} \qquad J_1^0(q) = \begin{bmatrix} z_0 \times (O_1 - O_0) \\ z_0 \end{bmatrix} \qquad \dot{q} = \dot{\theta}_1$$

Kinematic chain



For each column of $J_n^0 = \begin{bmatrix} J_1 & J_2 & J_3 & \dots & J_n \end{bmatrix}$

Revolute joint $J_i = \begin{bmatrix} z_{i-1} \times (O_n - O_{i-1}) \\ z_{i-1} \end{bmatrix}$

OR

Prismatic joint $J_i = \begin{bmatrix} z_{i-1} \\ 0 \end{bmatrix}$

The algorithm

Input: DH parameters $\underbrace{d, \theta}_{q}, a, \alpha, \rho \in \mathbb{R}^n$

Output: Jacobian matrix $J(q) = \left[J_1, J_2, \ldots, J_n\right]^T \in \mathbb{R}^{6 \times n}$

1 Compute: $T_i^{i-1} = DH(d_i, \theta_i, a_i, \alpha_i), \quad i = 1 \ldots n$

2 Compute: $T_i = T_i^0 = \prod_{k=1}^{i} T_k^{k-1}, \quad i = 1 \ldots n$

3 Initialise: $z_0 = [0 \quad 0 \quad 1]^T, O_0 = [0 \quad 0 \quad 0]^T$

4 ***for*** $i \in 1 \ldots n$

5 $$J_i = \begin{bmatrix} \rho_i z_{i-1} \times (O_n - O_{i-1}) + (1 - \rho_i) z_{i-1} \\ \rho_i z_{i-1} \end{bmatrix}$$

6 ***end for***

7 ***return*** J

Vector specifying type of joint

$$\rho_i = \begin{cases} 0, & \text{joint } i \text{ prismatic} \\ 1, & \text{joint } i \text{ revolute} \end{cases}$$

Z-axis and frame origin

$$T_i = \begin{bmatrix} R_i & O_i \\ 0 & 1 \end{bmatrix}$$

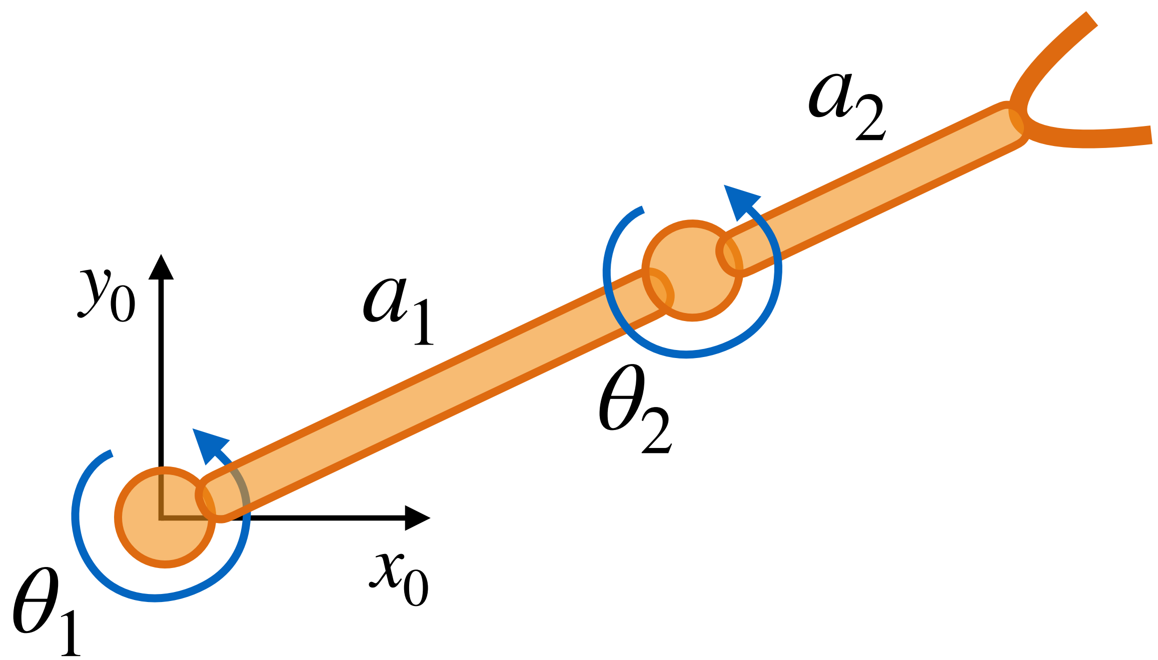
z_i

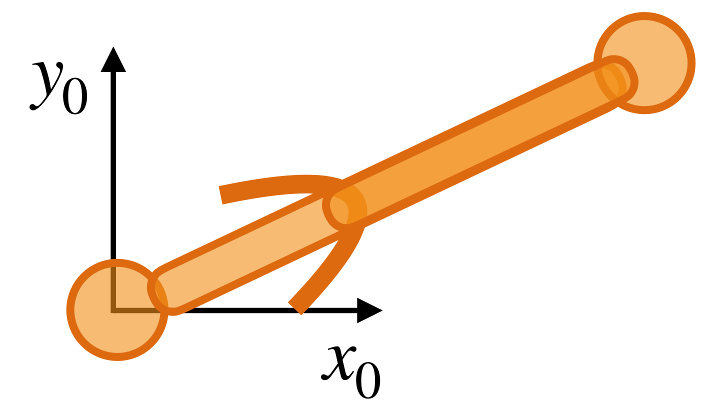
4.1. Solving the control problem: Singularities

Kinematic singularities are those configurations of the robot joints at which the Jacobian matrix becomes rank-deficient. It is important to find them and take into account when solving the control problems.

- Singularities represent configurations at which mobility of the structure is reduces, i.e., it is not possible to impose arbitrary motion on the end-effector.
- When the structure is at a singularity, infinite solutions to the inverse kinematics problem may exist.
- In the neighbourhood of a singularity, small velocities in the operational space may cause large velocities in the joint space.
- ***Boundary singularities*** - manipulator stretched or retracted; easy to avoid by avoiding boundaries of the reachable workspace.
- ***Internal singularities*** - caused by alignment of two or more axes of motion; can occur in any place inside the reachable workspace.

Boundary singularity





$$J = \begin{bmatrix} -a_1\sin(\theta_1) - a_2\sin(\theta_1+\theta_2) & -a_2\sin(\theta_1+\theta_2) \\ a_1\cos(\theta_1) + a_2\cos(\theta_1+\theta_2) & a_2\cos(\theta_1+\theta_2) \end{bmatrix}$$

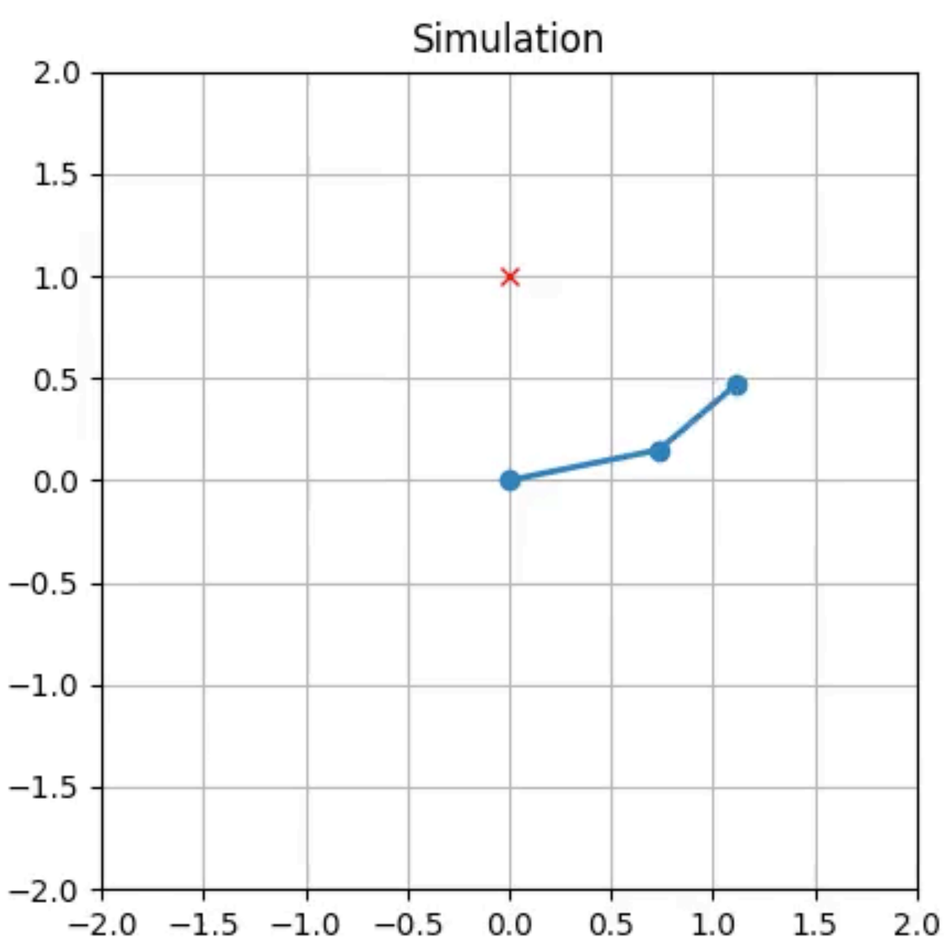
$$\det(J) = a_1 a_2 \sin(\theta_2)$$

- For $a_1, a_2 \neq 0$ the determinant equals zero if $\theta_2 = 0$ or $\theta_2 = \pi$.
- The position of the first joint is irrelevant.

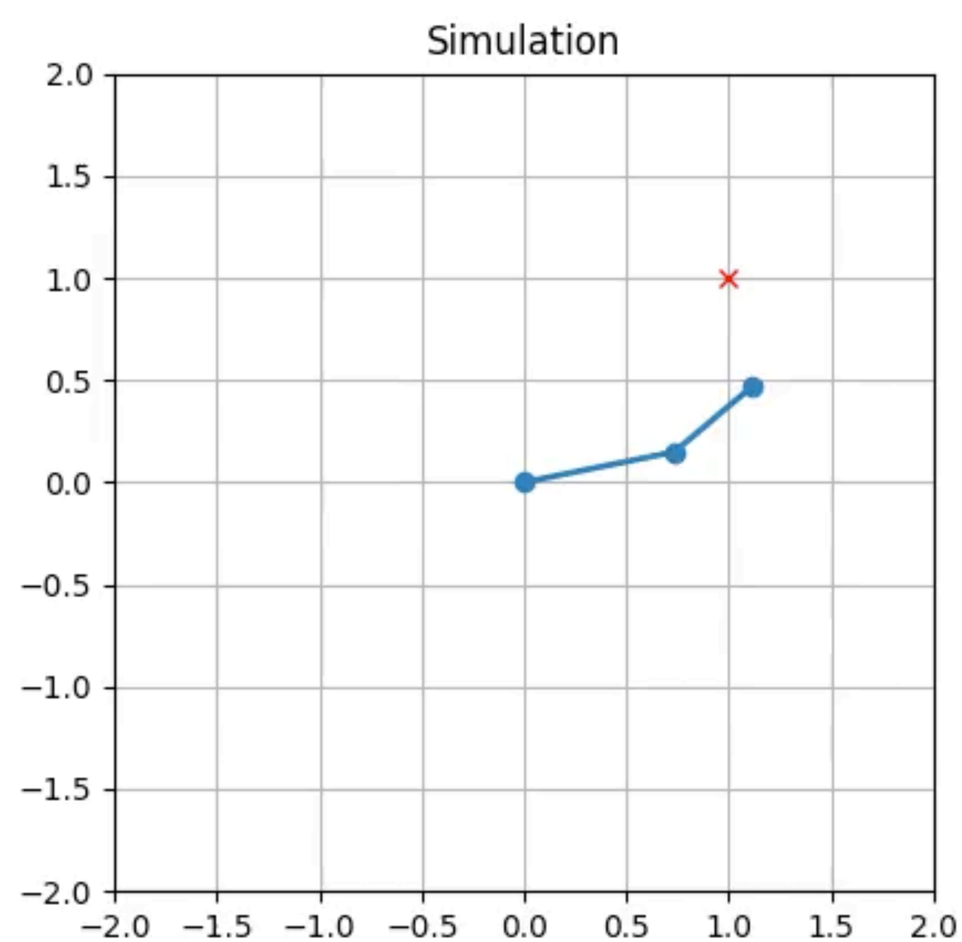
Inverse and pseudoinverse (Moore-Penrose)

$$\zeta = J^{-1}(\mathbf{q})\dot{x}_E$$ Square and full-rank Jacobian

$$\zeta = J^{\dagger}(\mathbf{q})\dot{x}_E$$ Square/non-square and full-rank Jacobian



Desired positions inside workspace

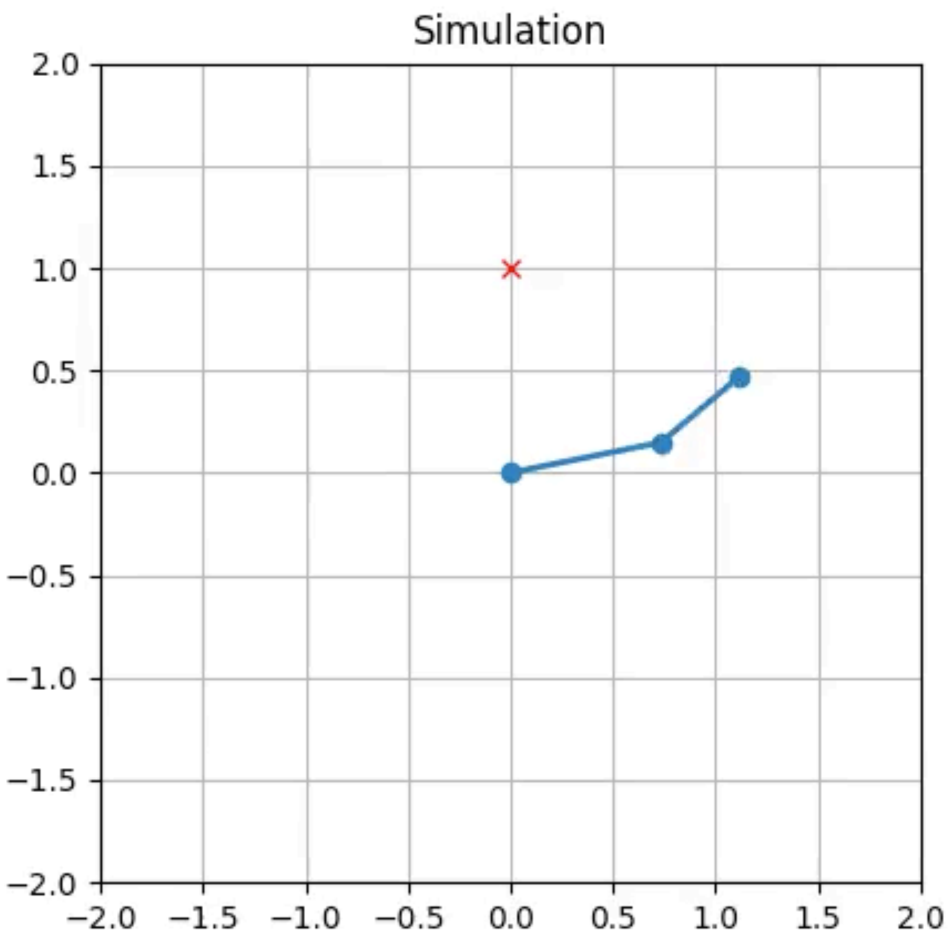


Desired position outside workspace

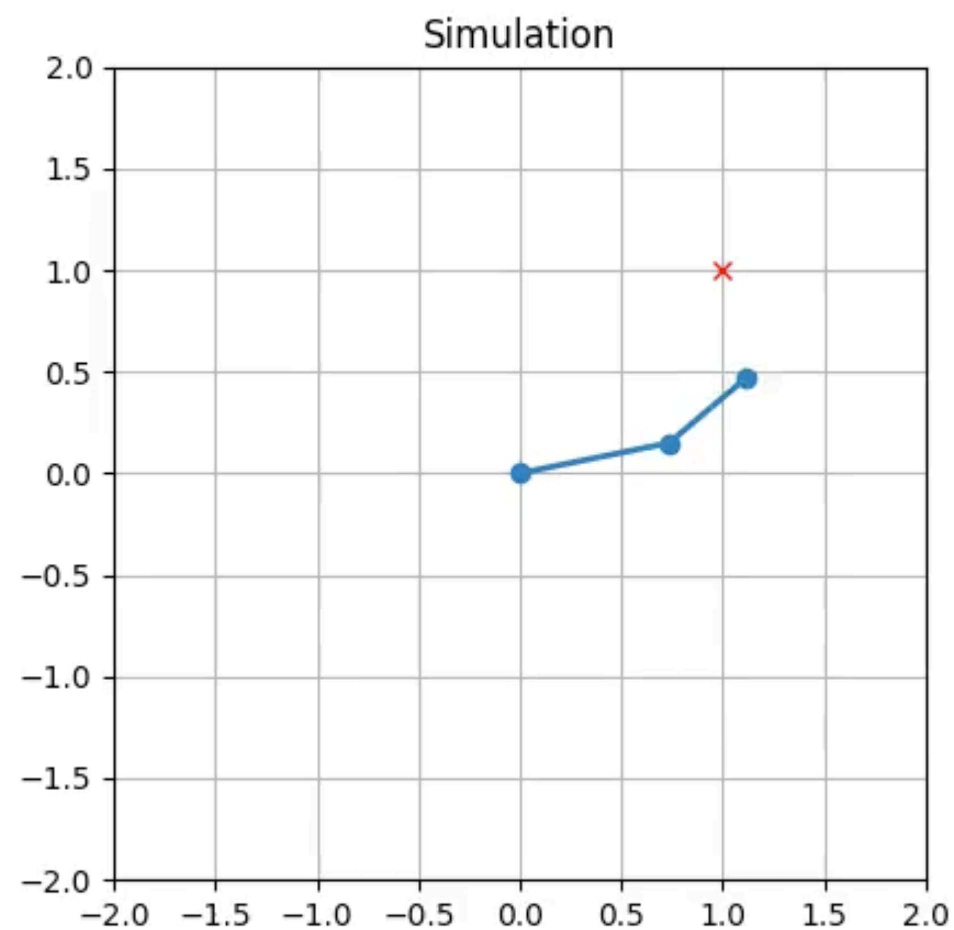
Jacobian transpose (no linearisation)

$$\zeta = J^T(\mathbf{q})\dot{x}_E$$

No inverse, no problem :)



Desired positions inside workspace



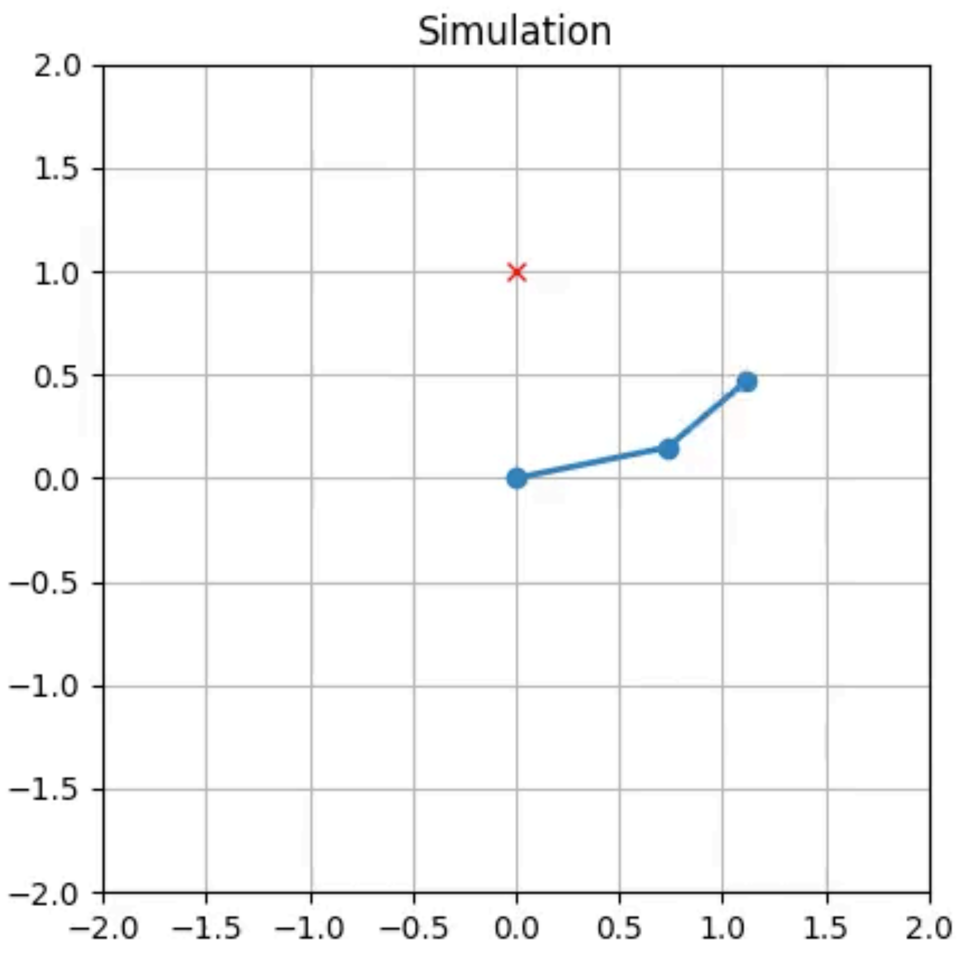
Desired position outside workspace

Damped least-squares (DLS)

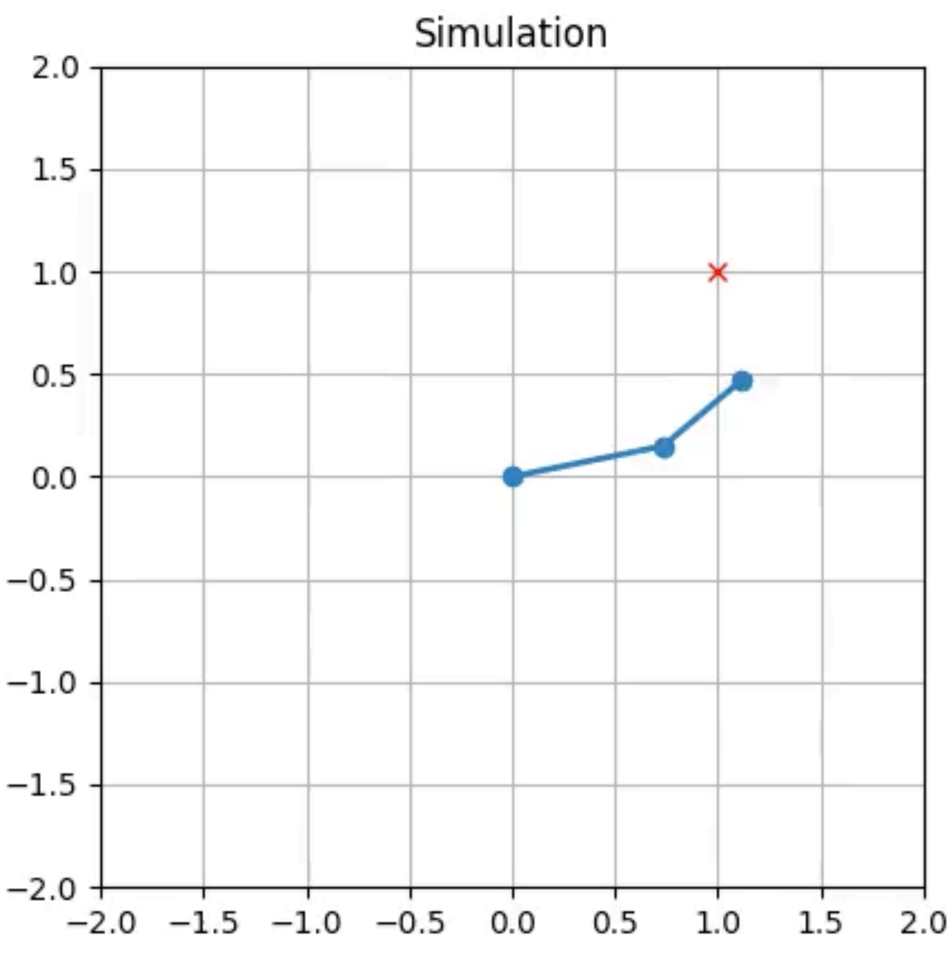
$$\zeta = J^T(\mathbf{q})\left(J(\mathbf{q})J^T(\mathbf{q}) + \lambda^2 I\right)^{-1} \dot{x}_E$$

Square/non-square Jacobian, full rank or rank deficient

Damping factor



Desired positions inside workspace



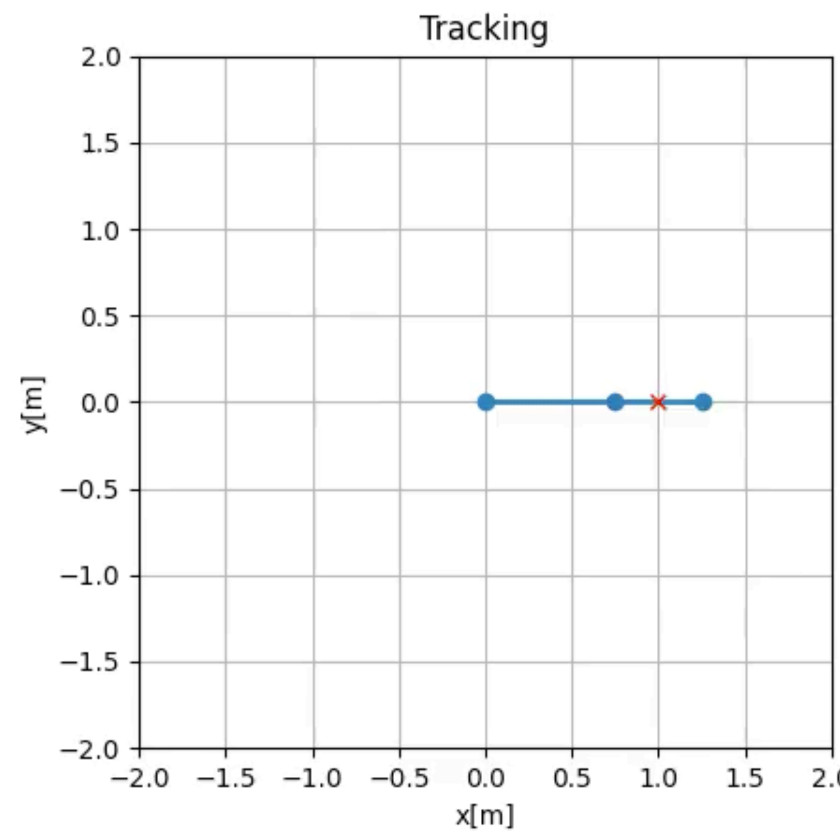
Desired position outside workspace

Tracking a moving object

$$\zeta = J^{-1}(\mathbf{q})(\dot{\sigma}_E + K\tilde{\sigma}_E)$$

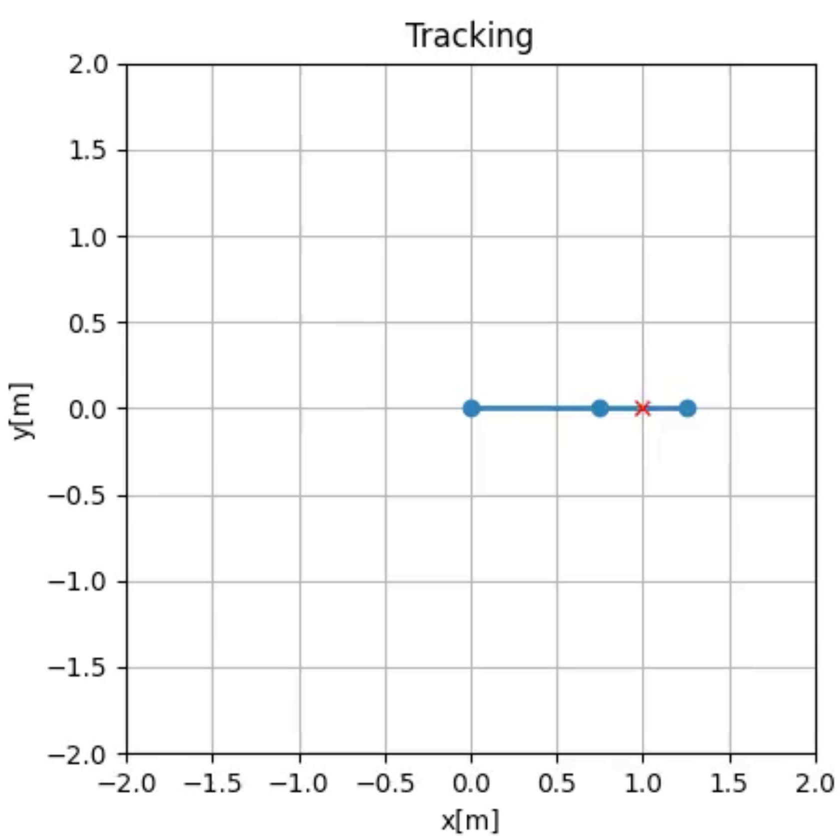
Feedforward

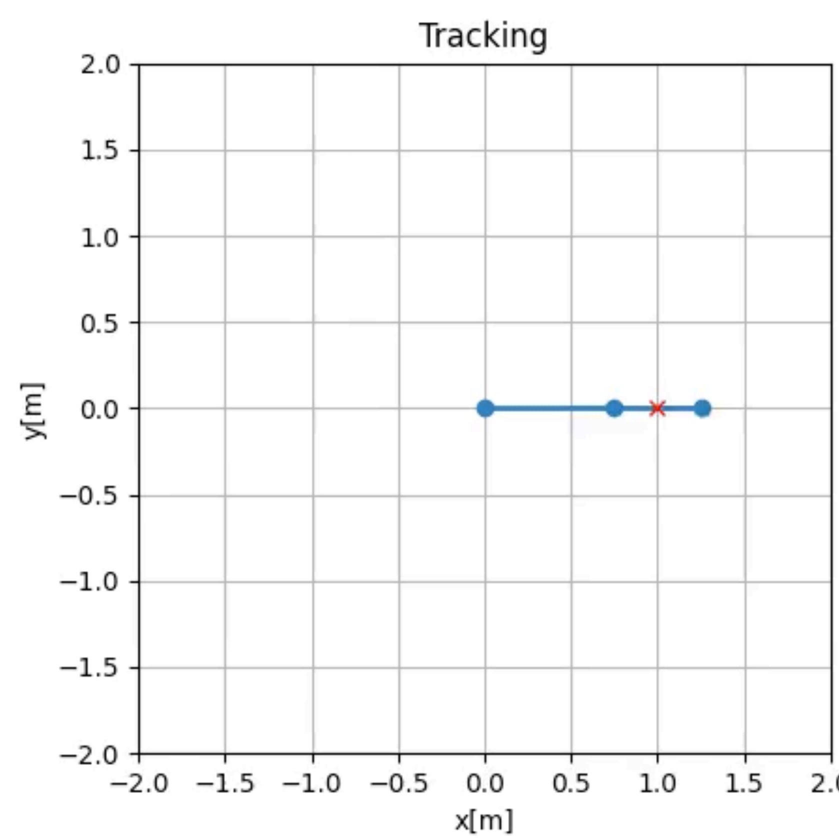
Feedback



K=10

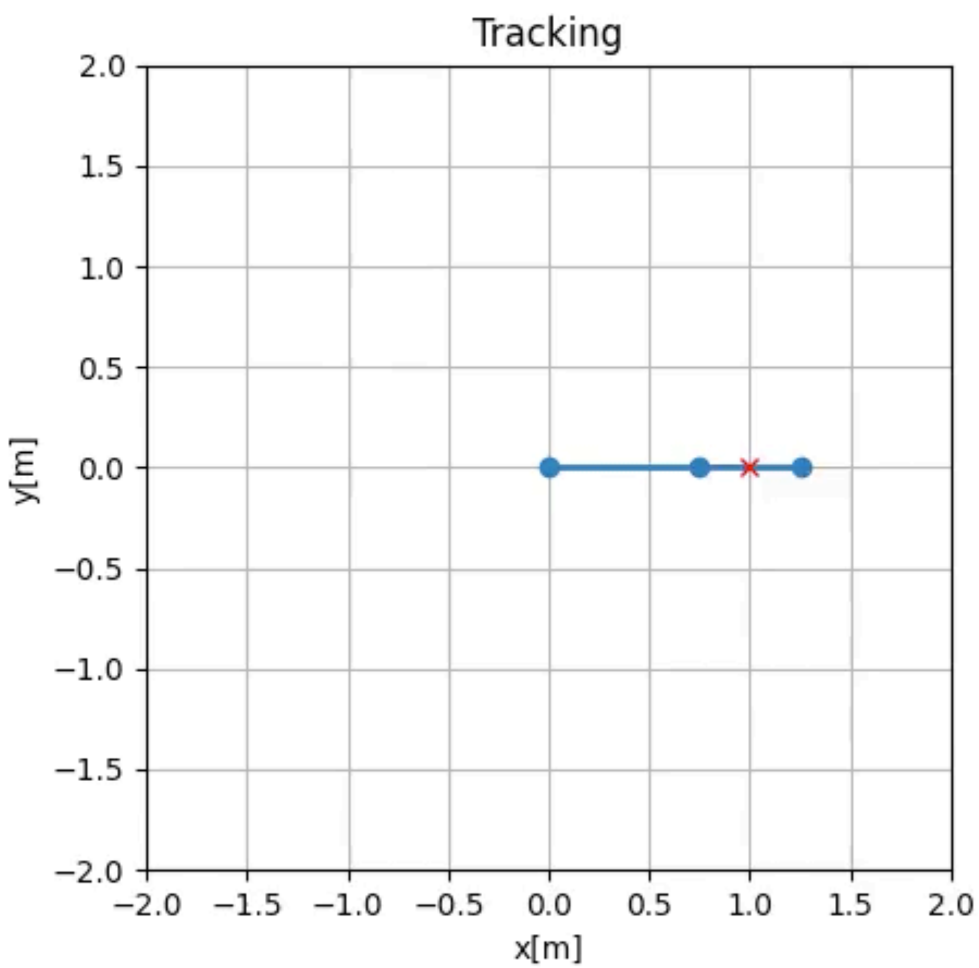
K=1



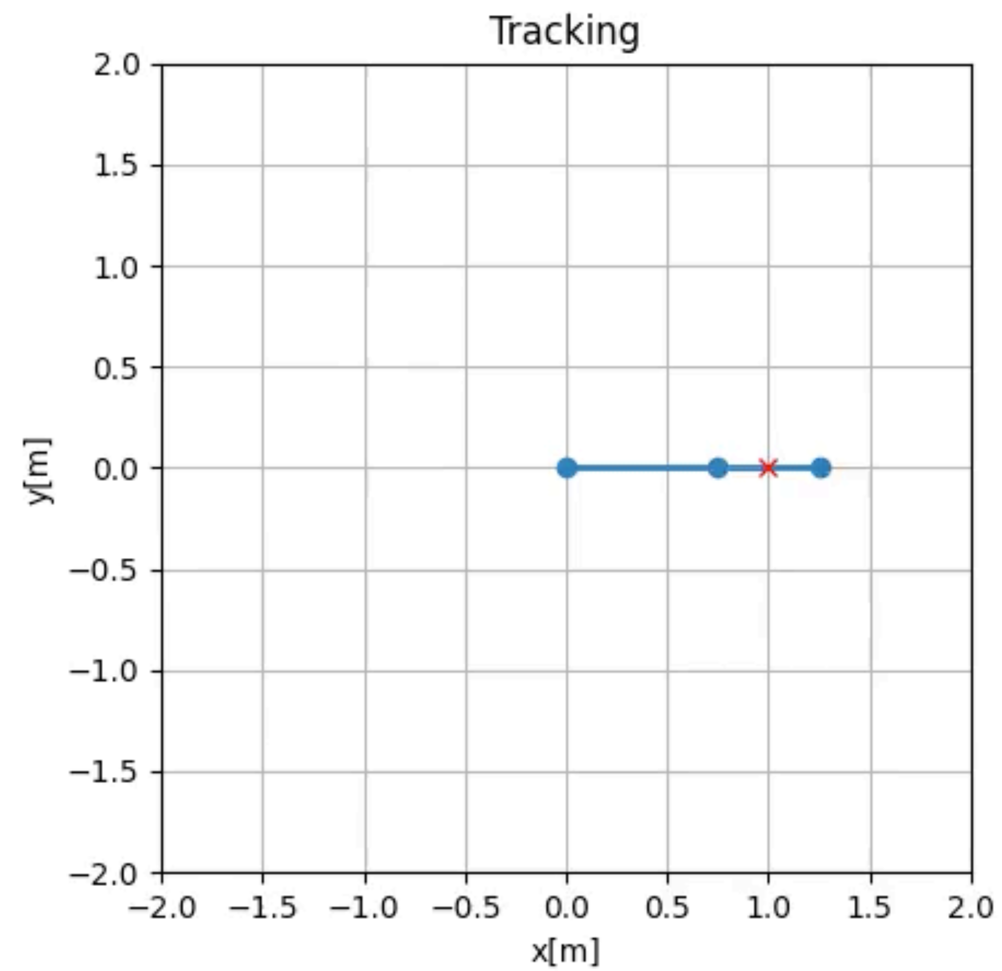


K=100

Tracking a moving object



K=1 + feedforward



K=10 + feedforward

Error analysis