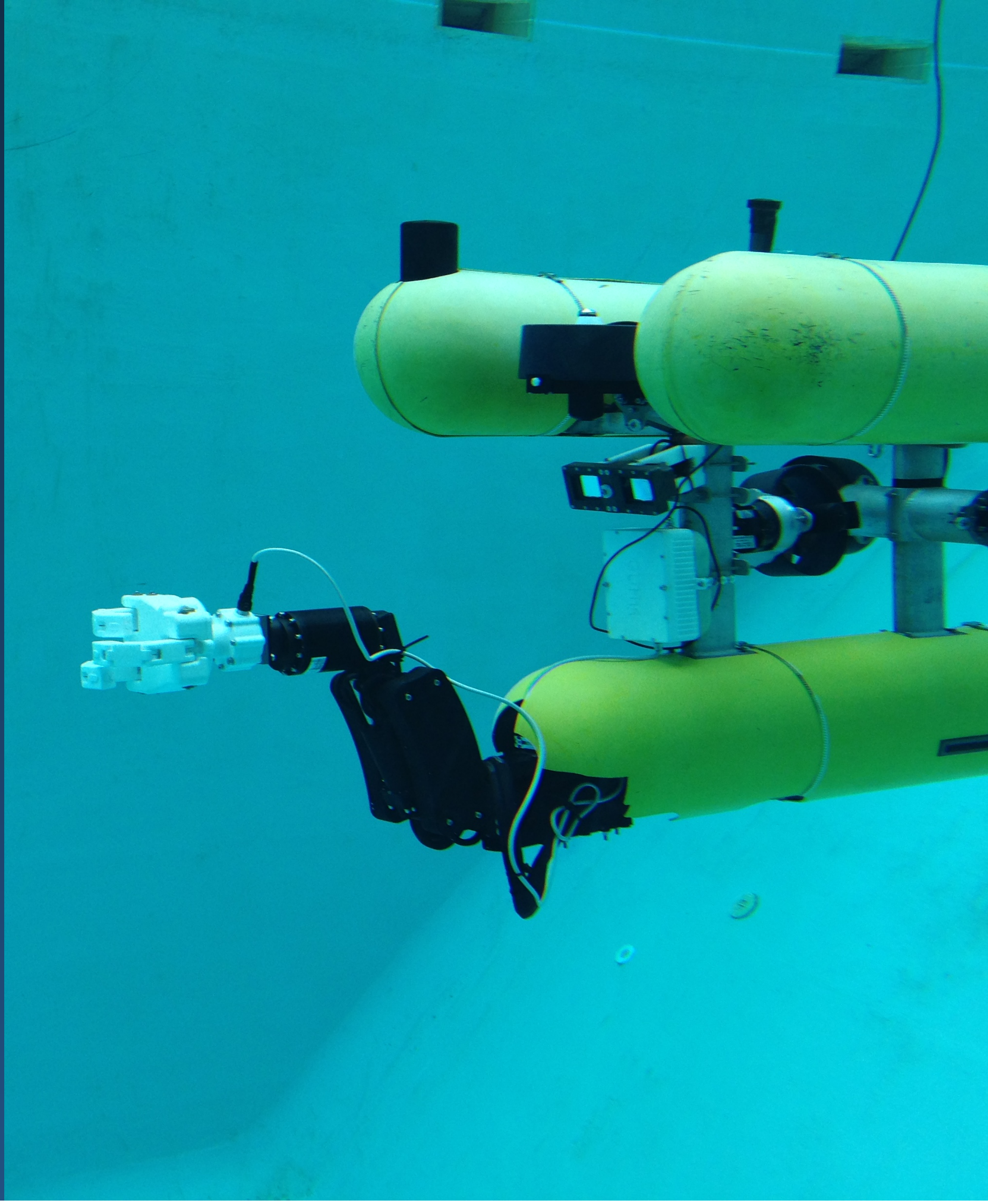
Universitat de Girona IFRoS MIRS

HANDS-ON INTERVENTION:

Vehicle-Manipulator Systems



Patryk Cieślak

patryk.cieslak@udg.edu

Lecture 3: Task-Priority kinematic control (part 1)

Contents

Redundancy

Task dimension

Task variable $\sigma_i = \sigma_i(\mathbf{q}) \in \mathbb{R}^{m_i}$

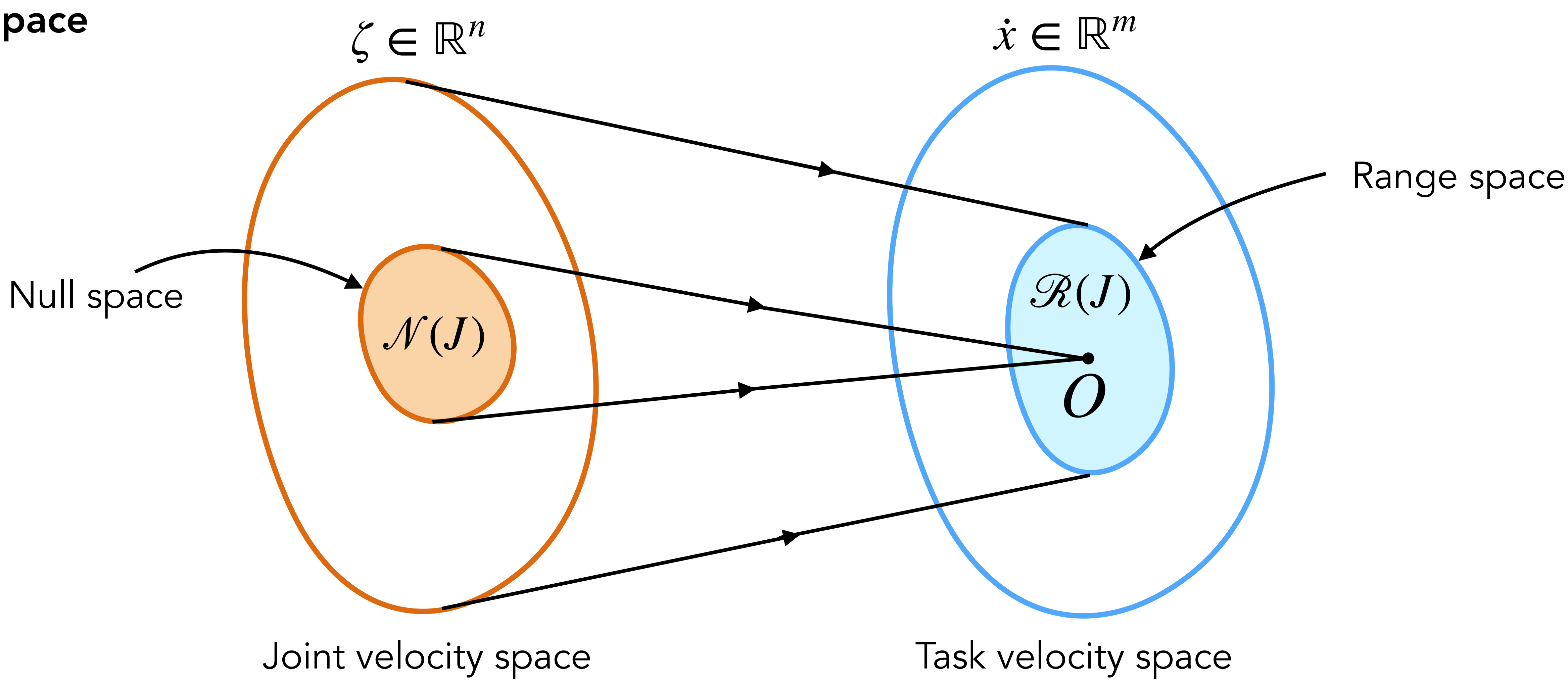
Task definition $\dot{x}_i = \dot{\sigma}_i + K_i\tilde{\sigma}_i$ $\tilde{\sigma}_i = \sigma_{i,d} - \sigma_i$ $J_i = J_i(\mathbf{q}) \in \mathbb{R}^{m \times n}$

Configuration vector $\mathbf{q} \in \mathbb{R}^n$ DOF

If $n > m_i$ robot is kinematically redundant with respect to task i

- The robot possesses more degrees of freedom that are necessary to perform the task
- The robot can achieve more dexterous motions
- In case of failure of one of the joints it can be possible to still complete the task
- A null space of the Jacobian can be used to realise additional tasks
- The Jacobian matrix for the task is not square (classic inverse cannot be used!)

Null space



- The **range space** of J is the subspace $\mathscr{R}(J)$ of $\mathbb{R}^m$, i.e., a set of task velocities that can be generated by the joint velocity space, in a given robot configuration
- The **null space** of J is the subspace $\mathscr{N}(J)$ of $\mathbb{R}^n$, i.e., a set of joint velocities that does not produce any velocity in the task space, in a given robot configuration

$$\boxed{\dim(\mathscr{R}(J)) + \dim(\mathscr{N}(J)) = n}$$

Solution for redundant systems

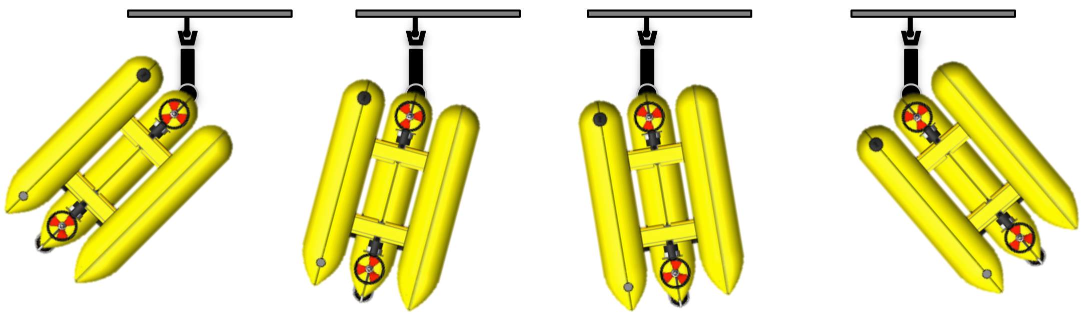
Arbitrary velocity vector

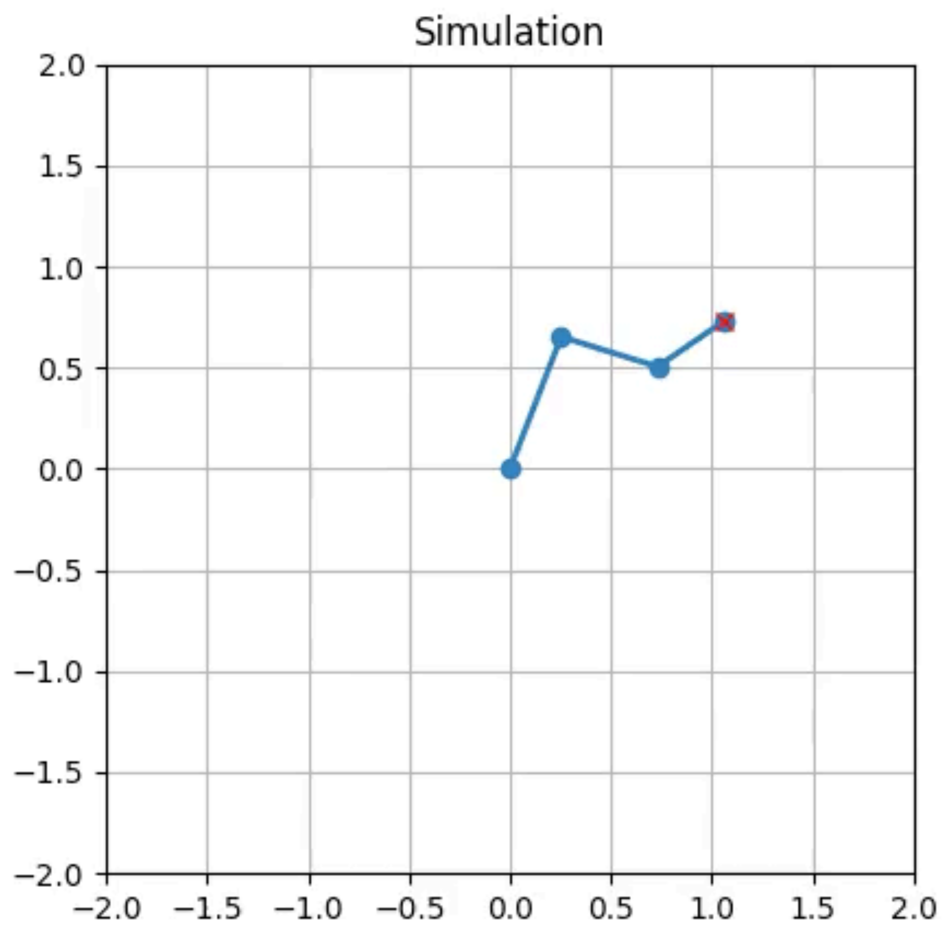
$$\zeta = J^{\dagger}(\mathbf{q})\dot{x}_E + \left(I - J^{\dagger}(\mathbf{q})J(\mathbf{q})\right) y$$

$$\boxed{P = \left(I - J^{\dagger}(\mathbf{q})J(\mathbf{q})\right)}$$ Null space projector

Non-zero if the system is redundant
with respect to the task

Null space motions (or internal motions)





How to take profit of the system's redundancy?

- Use the null space motions to perform secondary tasks without affecting the primary one
- Define a list of prioritised tasks
- Build an algorithm that can combine any number of prioritised tasks

Task hierarchy

		e.g.
Priority 1 (primary task)	$\dot{x}_1 = J_1(\mathbf{q})\zeta_1$	End-effector pose
Priority 2 (secondary task)	$\dot{x}_2 = J_2(\mathbf{q})\zeta_2$	Camera FOV
$\vdots$		
Priority i (least important task)	$\dot{x}_i = J_i(\mathbf{q})\zeta_i$	Preferred arm configuration

Combining tasks

Single task $$\zeta = J_1^{\dagger}\dot{x}_1 + \overset{P_1}{\boxed{\left(I - J_1^{\dagger}J_1\right)}}\, y$$

Arbitrary velocity (space for secondary task)

Two tasks $$\zeta = J_1^{\dagger}\dot{x}_1 + \bar{J}_2^{\dagger}\left(\dot{x}_2 - J_2(J_1^{\dagger}\dot{x}_1)\right) + \left(I - \bar{J}_2^{\dagger}\bar{J}_2\right) z \qquad \bar{J}_2 = J_2(I - J_1^{\dagger}J_1) = J_2P_1$$

The algorithm

Input: List of tasks $\{J_i(\mathbf{q}), \dot{x}_i(\mathbf{q})\}, \quad i \in 1 \ldots k$

Output: Quasi-velocities $\zeta_k \in \mathbb{R}^n$

Initialise: $\zeta_0 = 0^n, \quad P_0 = I^{n \times n}$

for $i \in 1 \ldots k$

$$\bar{J}_i(\mathbf{q}) = J_i(\mathbf{q}) P_{i-1}$$

$$\zeta_i = \zeta_{i-1} + \bar{J}_i^{\dagger}(\mathbf{q})(\dot{x}_i(\mathbf{q}) - J_i(\mathbf{q})\zeta_{i-1})$$

$$P_i = P_{i-1} - \bar{J}_i^{\dagger}(\mathbf{q})\bar{J}_i(\mathbf{q})$$

end for

return ζ_k

Task **Jacobians** and **desired velocities** have to be updated based on **current system state** before running the algorithm!

k number of tasks

n number of robot DOF

DLS can be used instead of pseudoinverse.

Pseudoinverse has to be used in the update of the null space projector! Using DLS will result in violation of task priorities.

Tasks

Equality

The task variable is regulated to reach the desired value

- End-effector position
- End-effector orientation
- Joint position
- Field of view
- Base orientation

Inequality (set)

The task variable is kept inside or outside of a specific set of values

- Joint limits
- Obstacle avoidance
- Manipulability

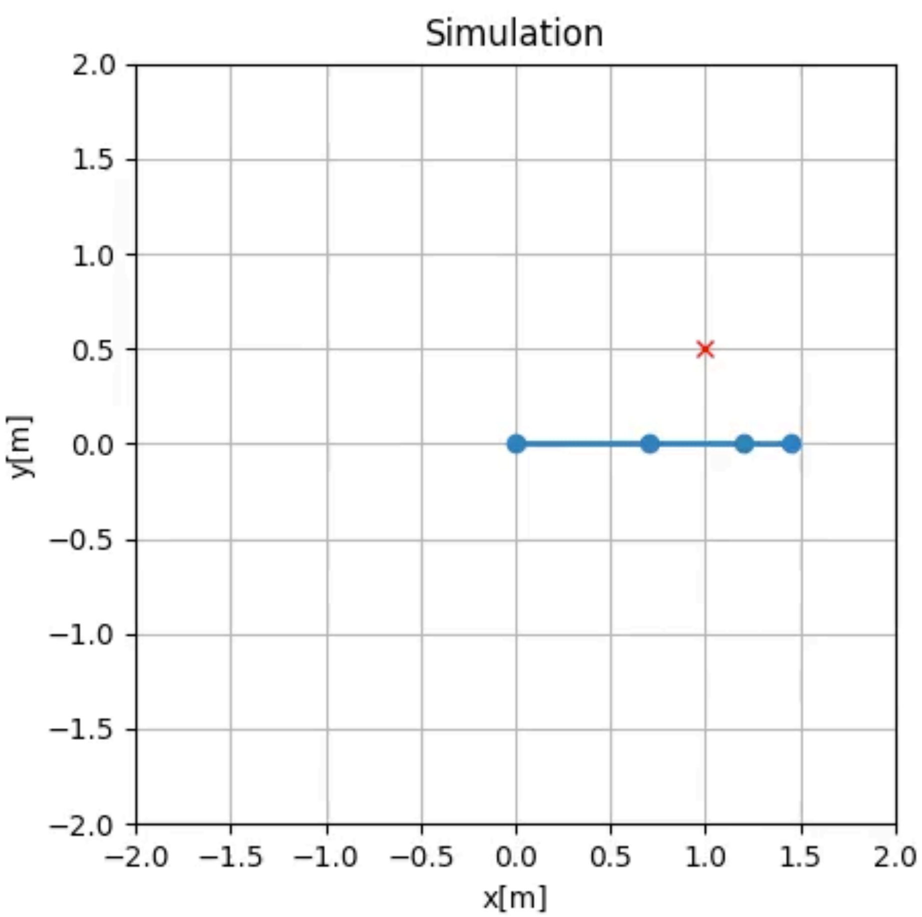
Require special treatment - *on-line switching*. This increases the complexity of the control algorithm and can result in discontinuities in the quasi-velocities.

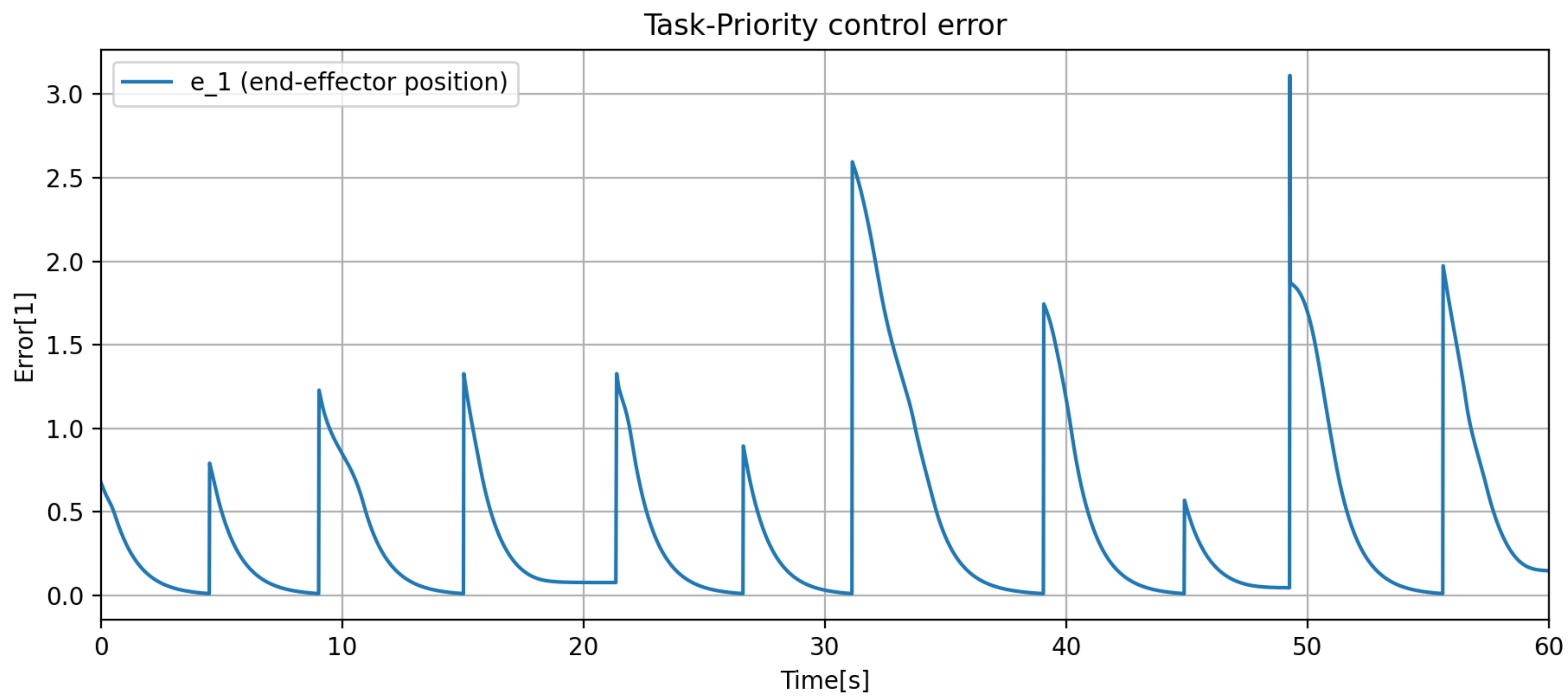
Task definition

$$\sigma_p = \sigma_p(\mathbf{q}) \in \mathbb{R}^{3\times 1}$$

$$\sigma_p = \eta_1 = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \qquad \tilde{\sigma}_p = \eta_{1,d} - \eta_1 \qquad J_p = J_p(\mathbf{q}) = J_v(\mathbf{q}) \in \mathbb{R}^{3\times n}$$

Linear velocity part of the end-effector Jacobian (3 top rows)





Quaternion

$$Q = \{w, \epsilon\}$$

Vector part of the quaternion $\epsilon = [x, y, z]^T \in \mathbb{R}^3$

$$w \in \mathbb{R}$$

$$w = \cos\frac{\vartheta}{2}$$

$$\epsilon = \sin\frac{\vartheta}{2}\hat{r}$$

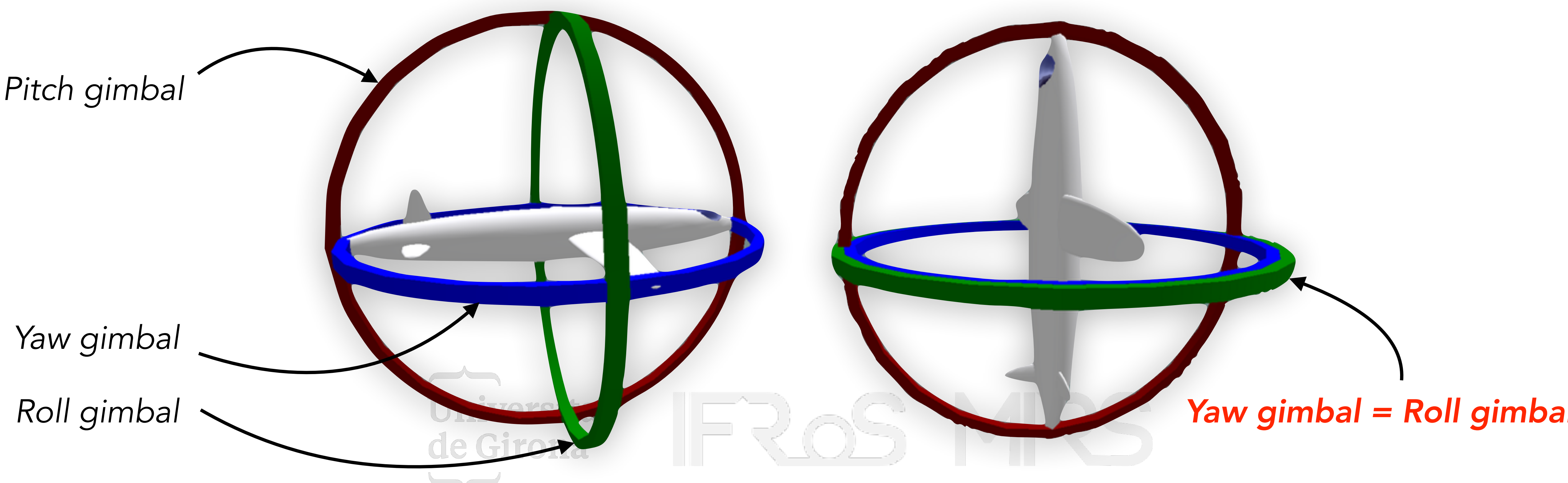
Rotation axis

$$w^2 + x^2 + y^2 + z^2 = 1$$

Does not suffer from gimbal lock like the RPY representation!

Gimbal lock

The **problem** can occur when using the **roll-pitch-yaw** (RPY) **representation** when describing **orientation in 3D space**. It manifests by the **alignment of rotation axes** (virtual gimbals).



Quaternion properties

Equivalence $\{w, \epsilon\} \equiv \{-w, -\epsilon\}$

Inverse $Q^{-1} = \{w, -\epsilon\}$ quaternion extracted from $R^{-1} = R^T$

Product $Q_1 * Q_2 = \{w_1 w_2 - \epsilon_1^T \epsilon_2, \;\; w_1 \epsilon_2 + w_2 \epsilon_1 + \epsilon_1 \times \epsilon_2\}$ corresponding to $R_1 R_2$

Quaternion from rotation matrix

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$w = \frac{1}{2}\sqrt{r_{11} + r_{22} + r_{33} + 1}$$

$$\epsilon = \frac{1}{2}\begin{bmatrix} \operatorname{sgn}(r_{32} - r_{23})\sqrt{r_{11} - r_{22} - r_{33} + 1} \\ \operatorname{sgn}(r_{13} - r_{31})\sqrt{r_{22} - r_{33} - r_{11} + 1} \\ \operatorname{sgn}(r_{21} - r_{12})\sqrt{r_{33} - r_{11} - r_{22} + 1} \end{bmatrix}$$

$$\operatorname{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

Attitude error representation

Rotation matrix from the **current end-effector frame** to the world frame $\quad {}^0R_B \in \mathbb{R}^{3\times3} \rightarrow Q = \{w, \epsilon\}$

Rotation matrix from the frame expressing the **desired end-effector orientation** to the world frame $\quad {}^0R_d \in \mathbb{R}^{3\times3} \rightarrow Q_d = \{w_d, \epsilon_d\}$

Rotation matrix that aligns the frames $\quad \tilde{R} = {}^0R_B^T\,{}^0R_d = {}^BR_0\,{}^0R_d$

In quaternion sense $\quad \tilde{Q} = Q^{-1} * Q_d$

$$\tilde{Q} = \{\tilde{w}, \tilde{\epsilon}\} = \{ww_d + \epsilon^T\epsilon_d,\ \ w\epsilon_d - w_d\epsilon - \epsilon \times \epsilon_d\}$$

Since for aligned frames $\tilde{R} = I^{3\times3} \rightarrow \tilde{Q} = \{1, 0\}$ it is sufficient to represent the attitude error as $\tilde{\epsilon}$.

Task definition

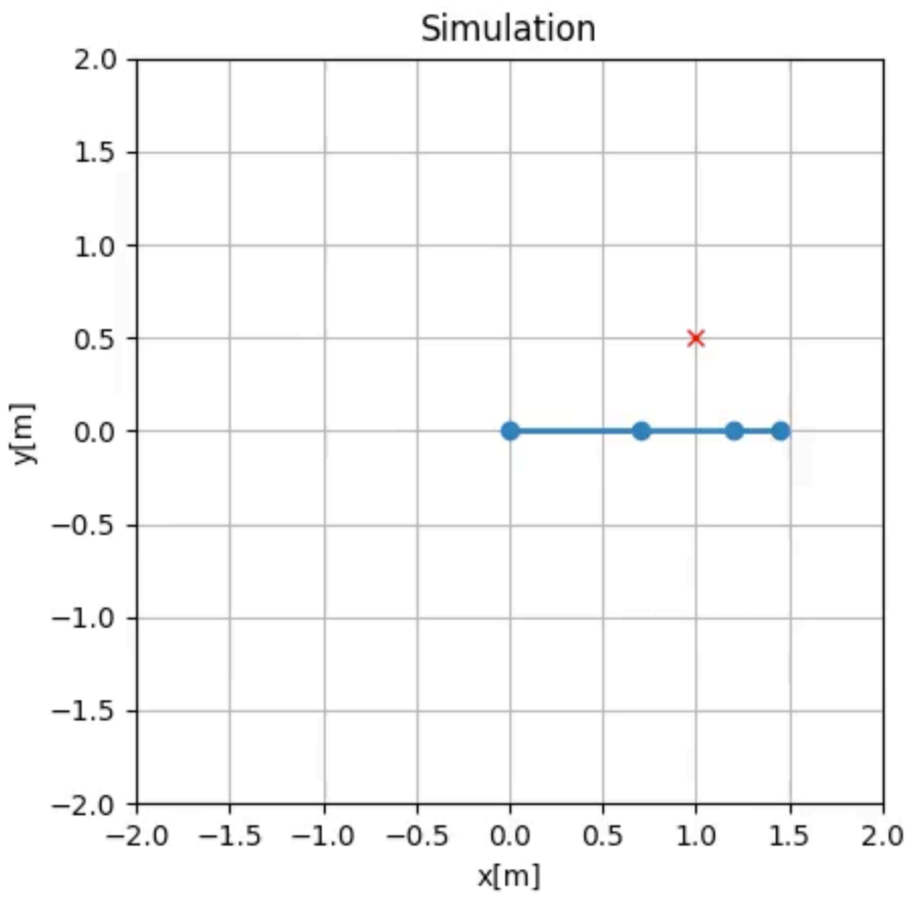
$$\sigma_o = \sigma_o(\mathbf{q}) \in \mathbb{R}^{3\times1}$$

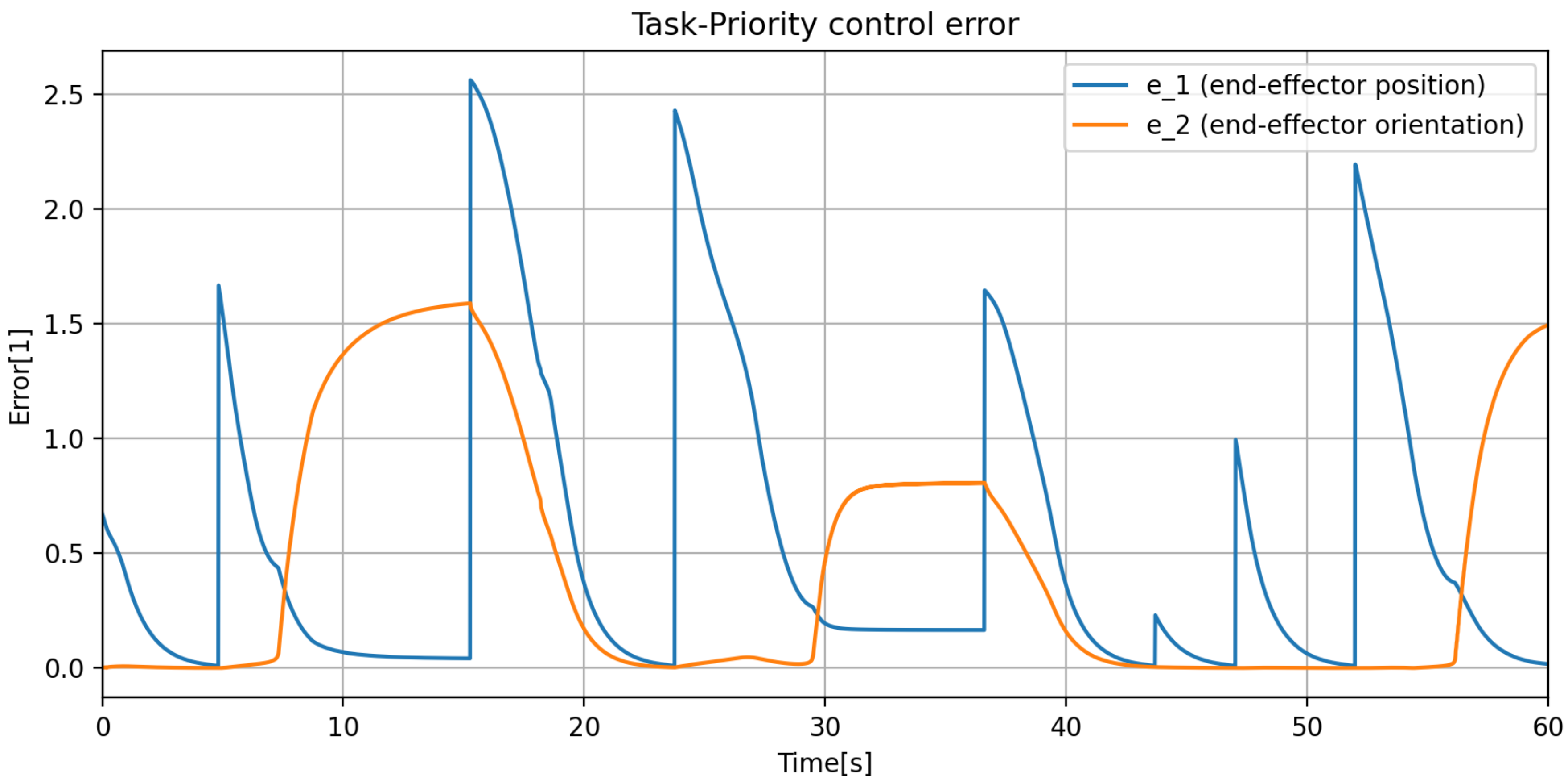
Angular velocity part of the end-effector Jacobian (3 bottom rows)

$$\tilde{\sigma}_o = w\epsilon_d - w_d\epsilon - \epsilon \times \epsilon_d$$

$$J_o = J_o(\mathbf{q}) = J_\omega(\mathbf{q}) \in \mathbb{R}^{3\times n}$$

End-effector orientation task at the second priority level





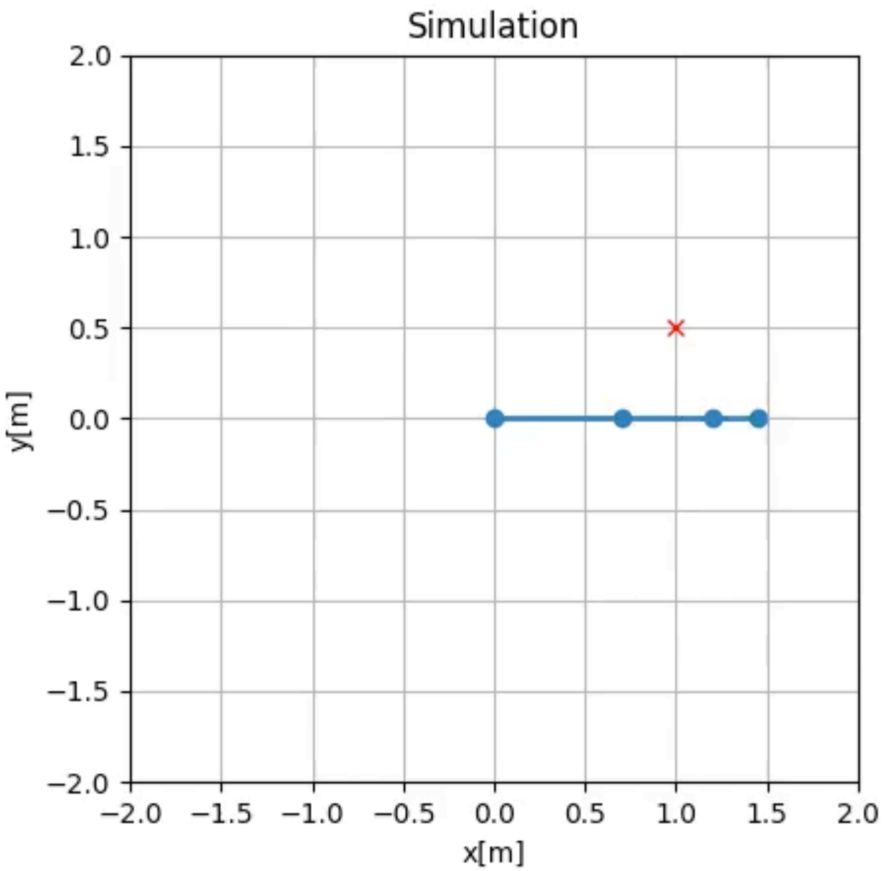
Task definition

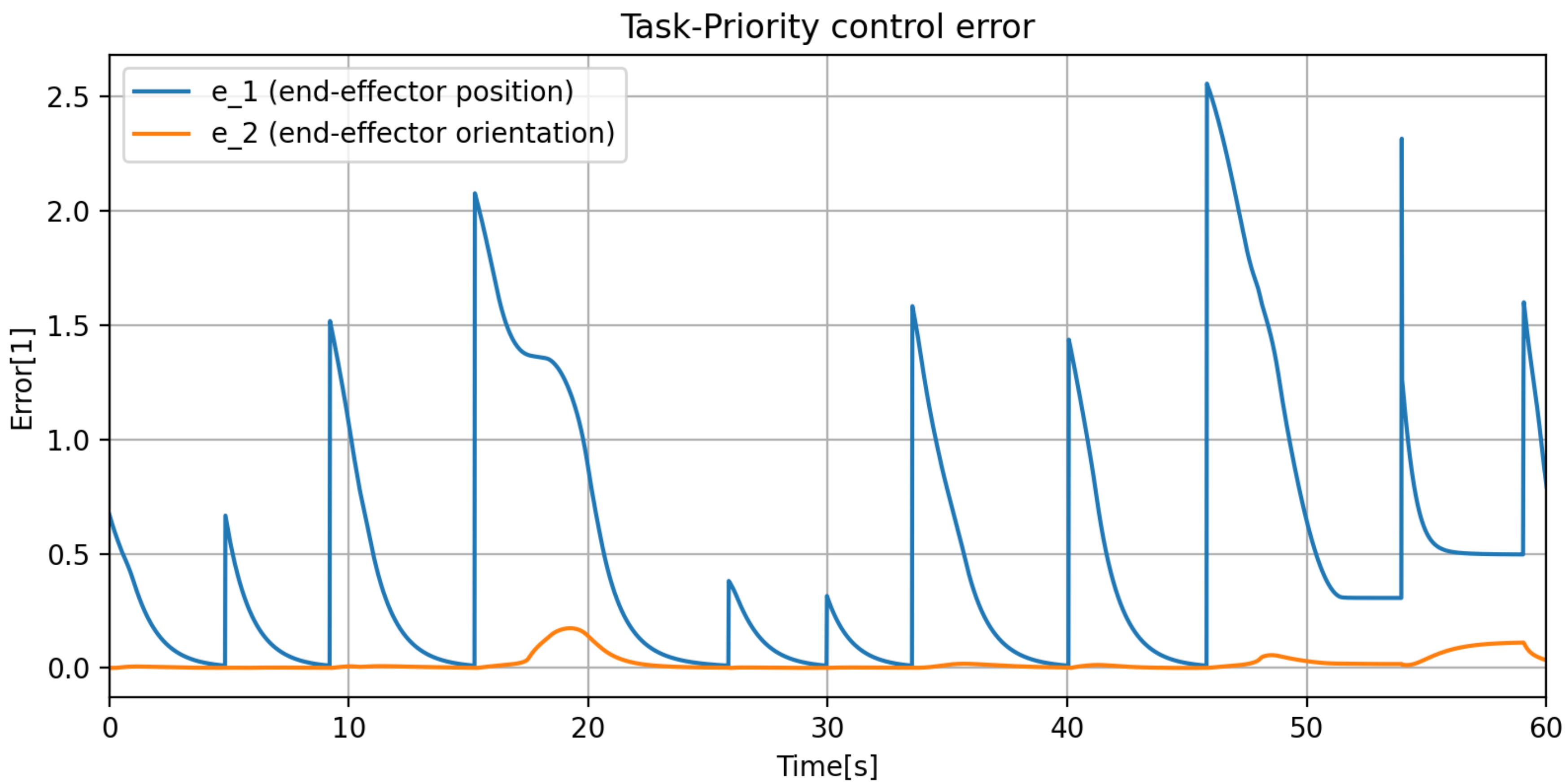
$$\sigma_c = \sigma_c(\mathbf{q}) \in \mathbb{R}^{6\times 1}$$

$$\tilde{\sigma}_c = \begin{bmatrix} \eta_{1,d} - \eta_1 \\ w\epsilon_d - w_d\epsilon - \epsilon \times \epsilon_d \end{bmatrix}$$

Full end-effector Jacobian

$$J_c = J_c(\mathbf{q}) = J(\mathbf{q}) \in \mathbb{R}^{6\times n}$$





Task definition

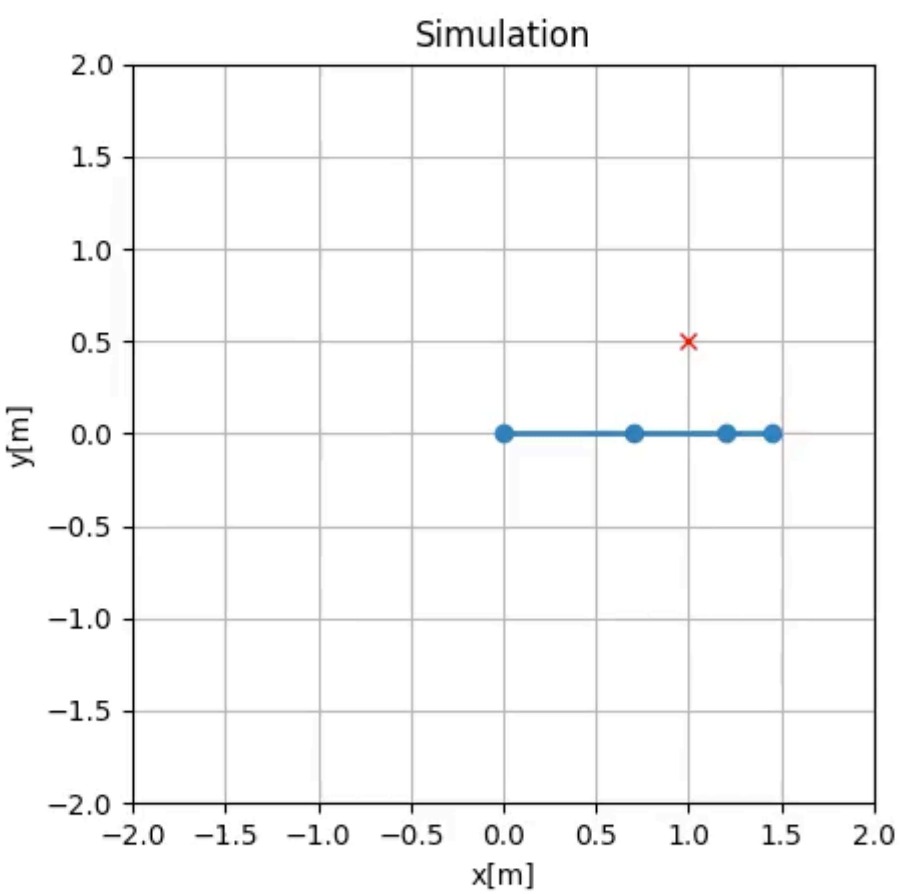
$$\sigma_{ji} = \sigma_{ji}(\mathbf{q}) \in \mathbb{R}$$

$$\tilde{\sigma}_{ji} = \sigma_{ji,d} - \sigma_{ji}$$

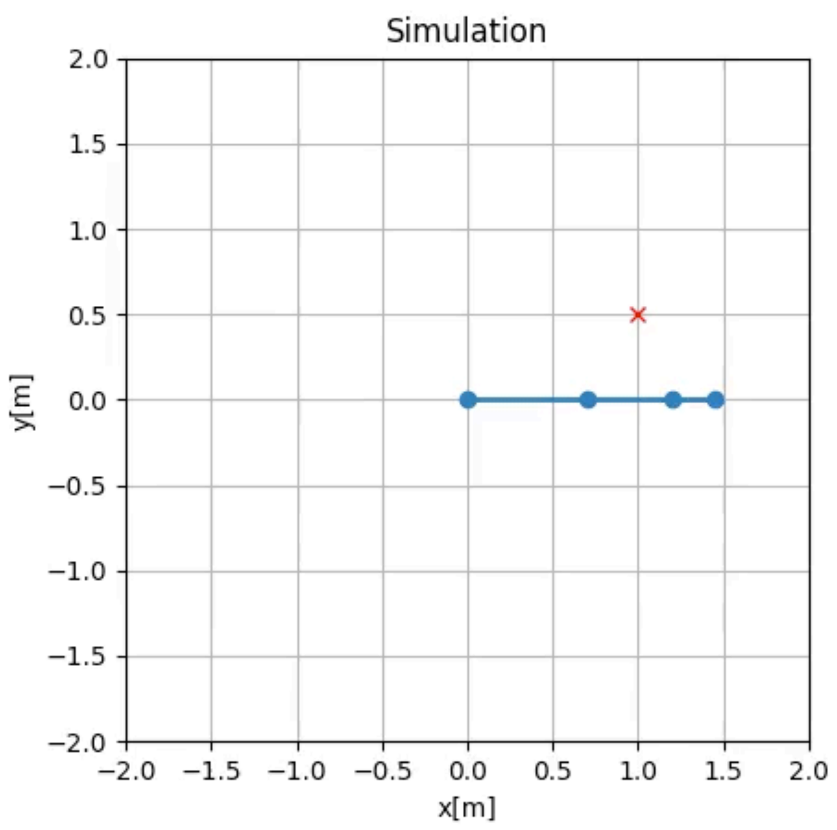
Single-entry row matrix

$$\begin{array}{l} \qquad\quad\ 0\ \ 1\ \ \dots\ \ i\ \ \dots\ \ n \\ J_{ji} = [0,0,\dots,1,\dots,0] \in \mathbb{R}^{1\times n} \end{array}$$

Joint position task at the second priority level



Joint position task at the first priority level



4. Practical extensions: Velocity scaling/limitting

Problem

The quasi-velocities being the **output of the Task-Priority algorithm** may **not** be **possible or safe to achieve** by the actual robotic platform. **Velocity limits** are commonly defined for each of the DOF, closely related to the performance of the utilised drives.

$$|\zeta_i| \leq \zeta_{i,\max} \quad i = 1 \ldots n$$

Solution

The **output quasi-velocities** have to be **scaled** to not exceed any of the velocity limits. The **ratios between the velocities** have to remain **identical** to not affect the solution of the algorithm (linearisation).

$s = \max\limits_{i \in 1 \ldots n} \left(\dfrac{|\zeta_i|}{\zeta_{i,\max}} \right)$

if $s > 1$

    return $\dfrac{\zeta}{s}$

else

    return ζ

end if

Idea

Introduce a **modification to the Task-Priority algorithm** which allows for specifying which **DOF of the system** are the **preferred** ones **to be utilised**. This preference can result from optimising different parameters, e.g., set-point regulation time, energy consumption, accuracy, tracking performance etc.

Solution

A **weighting matrix** can be introduced in the definition of the **Jacobian inverse function**, which will affect the resulting velocities by changing the **cost incurred** by using **specific DOF** of the robot.

Weighting matrix

$$W \in \mathbb{R}^{n \times n}, \quad W = \mathrm{diag}(w_1, w_2, \ldots, w_n)$$

The higher the weight the less preferred the use of the related DOF.

Pseudoinverse

$$\zeta = J^{\dagger}(\mathbf{q})\dot{x}_E \quad \rightarrow \quad \zeta = W^{-1}J^T(\mathbf{q})\left(J(\mathbf{q})W^{-1}J^T(\mathbf{q})\right)^{-1}\dot{x}_E$$

DLS

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

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

Example (mobile vehicle-manipulator system)