Universitat de Girona

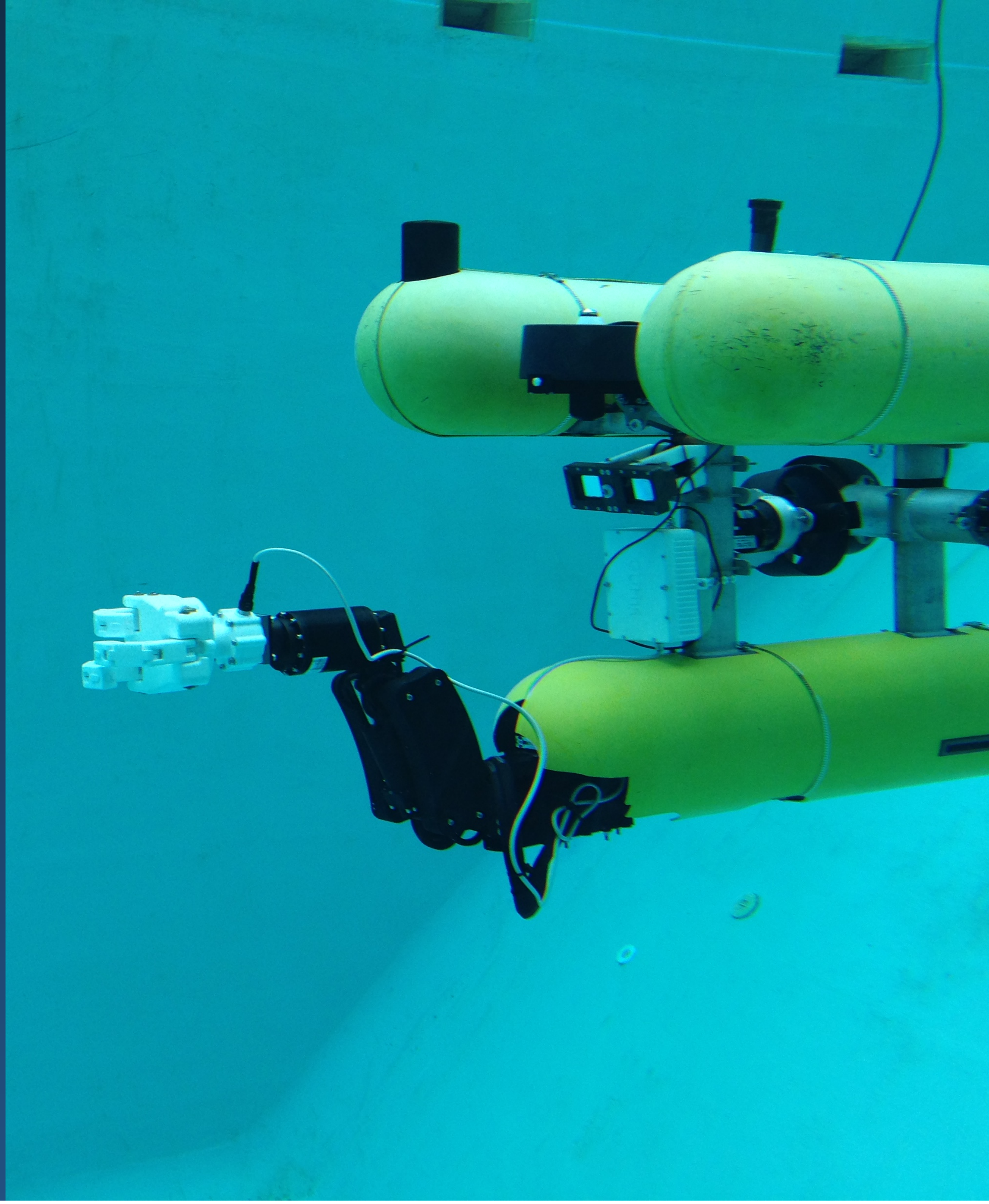
IFRoS MIRS

HANDS-ON INTERVENTION:

Vehicle-Manipulator Systems

Patryk Cieślak

patryk.cieslak@udg.edu



Lecture 4: Task-Priority kinematic control (part 2)

Contents

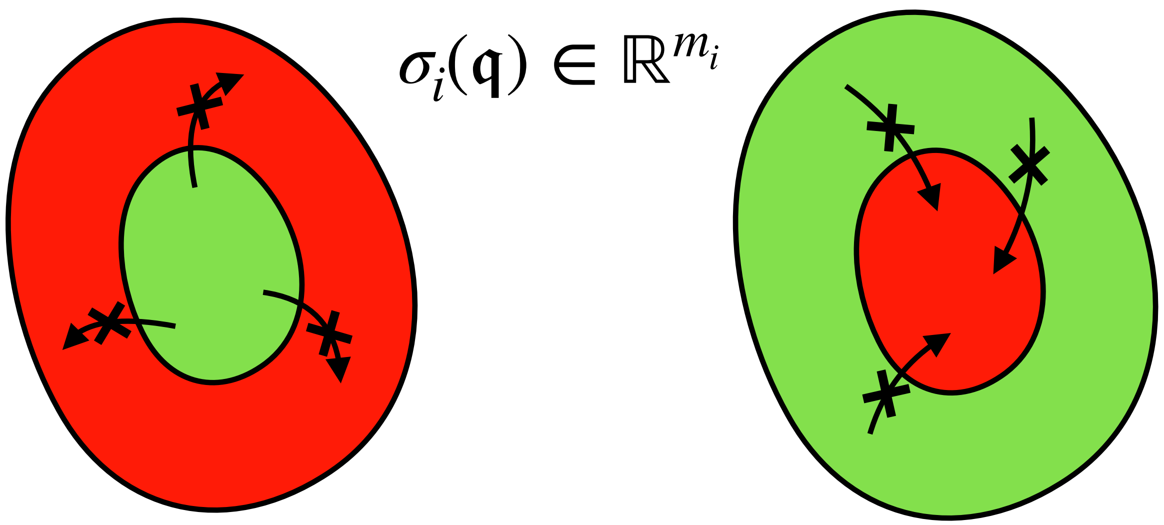
1.1. Inequality tasks: Concept

Equality task

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

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

Inequality task (set task)



- The goal is to keep the task variable in a specific set of values or avoid a specific set of values.
- Set tasks are most often related to system safety and therefore they need to be positioned at the top of the task hierarchy.
- It is necessary to implement a switching mechanism, which enables the task only in case of approaching the border of the safe set, to ensure minimal influence on the successful completion of the main control goal (e.g., approaching a desired end-effector pose).
- Task switching can lead to discontinuities in the commanded quasi-velocities, depending on its implementation.

Task definition

$$\sigma_{li} = \sigma_{li}(\mathbf{q}) = \mathbf{q}_i$$

$$\mathcal{S}_{li} = [\mathbf{q}_{i,\min}, \mathbf{q}_{i,\max}]$$ Safe set

$$\boxed{\sigma_{li} \in S_{li}}$$ Goal

$$\boxed{\begin{aligned} \dot{x}_{li} &= 1 \\ J_{li} &= [0,0,\ldots,1,\ldots,0] \in \mathbb{R}^{1\times n} \end{aligned}}$$

Task switching

$\alpha_{li} \in \mathbb{R}$ Activation threshold

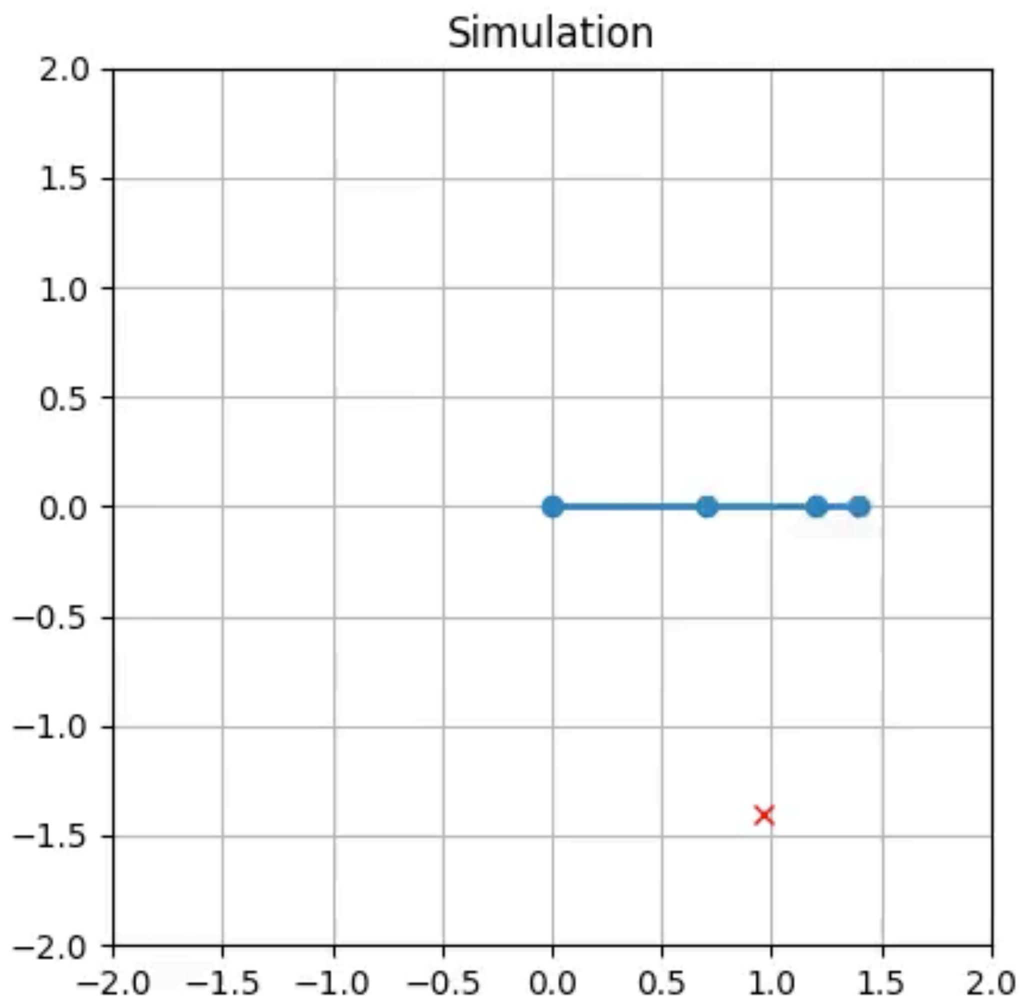
$\delta_{li} \in \mathbb{R}$ Deactivation threshold

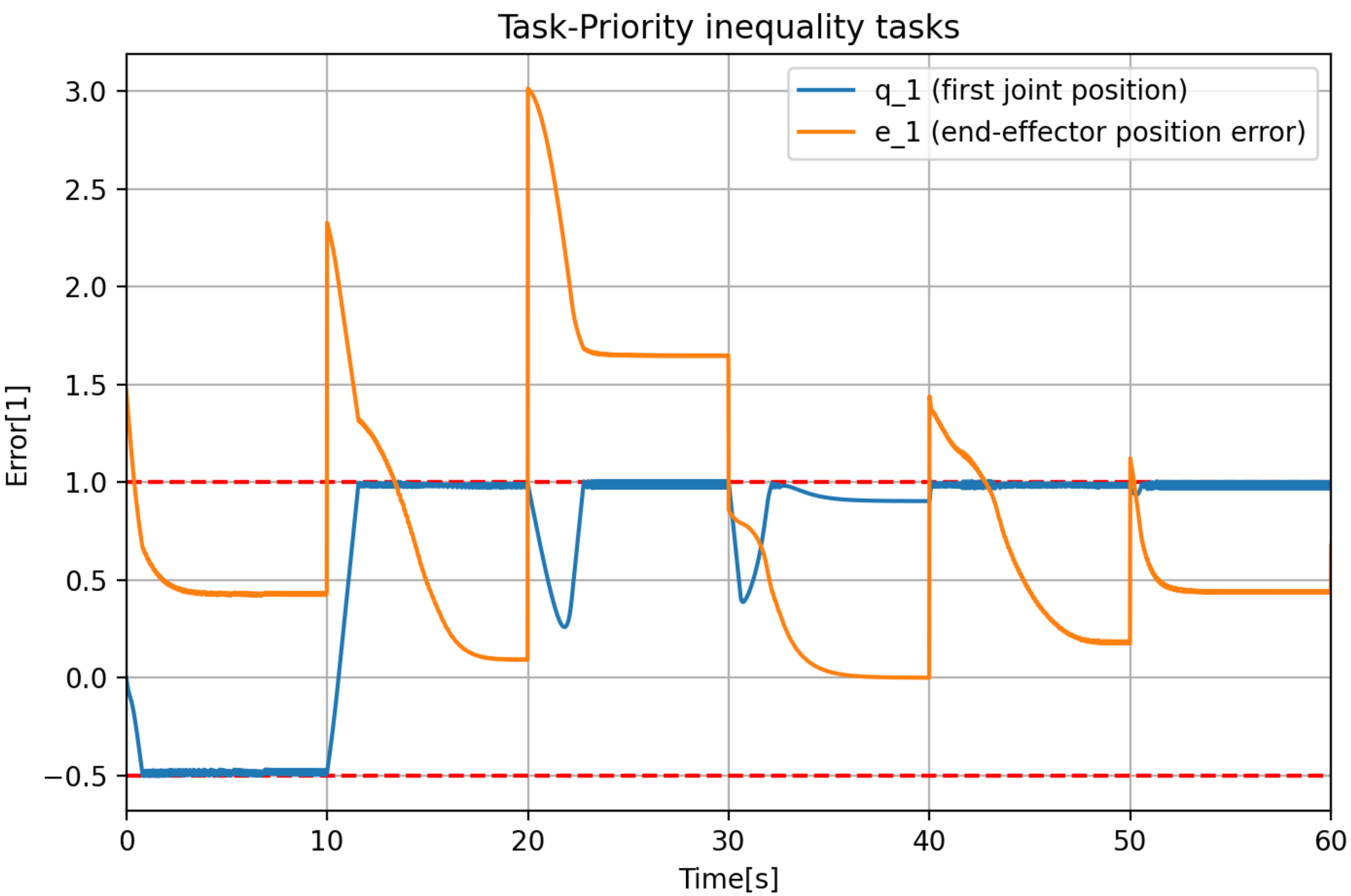
$\boxed{\delta_{li} > \alpha_{li}}$ To avoid chatter!

$$a_{li}(\mathbf{q}) = \begin{cases} -1, & a_{li} = 0 \wedge \mathbf{q}_i \geq \mathbf{q}_{i,\max} - \alpha_{li} \\ 1, & a_{li} = 0 \wedge \mathbf{q}_i \leq \mathbf{q}_{i,\min} + \alpha_{li} \\ 0, & a_{li} = -1 \wedge \mathbf{q}_i \leq \mathbf{q}_{i,\max} - \delta_{ji} \\ 0, & a_{li} = 1 \wedge \mathbf{q}_i \geq \mathbf{q}_{i,\min} + \delta_{ji} \end{cases}$$

Activation function

Simulation


Simulation
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


Task-Priority inequality tasks
q_1 (first joint position)
e_1 (end-effector position error)
Error[1]
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
0 10 20 30 40 50 60
Time[s]

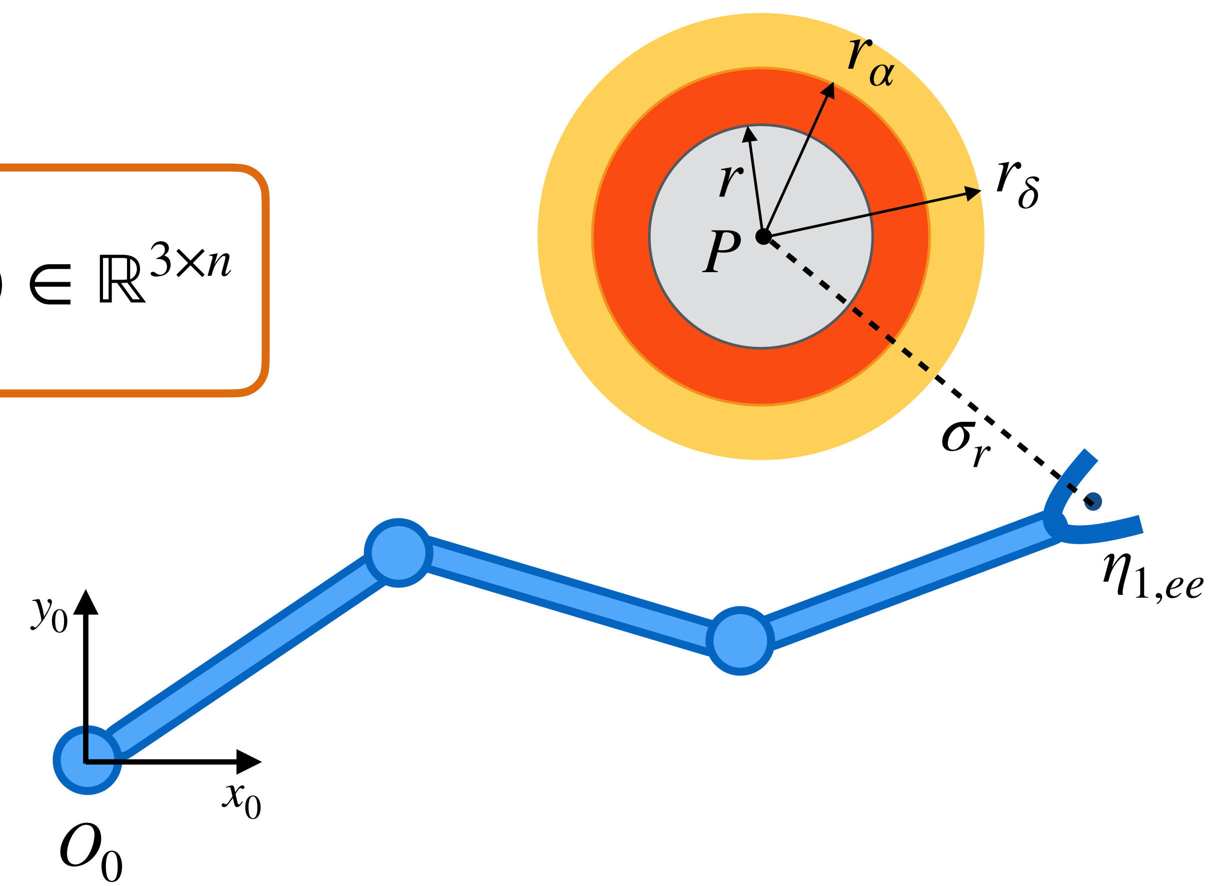
Task definition

$$\sigma_r = \sigma_r(\mathbf{q}) = |\eta_{1,ee}(\mathbf{q}) - P|$$

$$\mathcal{S}_r = \mathbb{R}^3 \backslash \{\eta_1 : |\eta_1 - P| \leq r\}$$ Safe set

$$\boxed{\sigma_r \in S_r}$$ Goal

$$\boxed{\dot{x}_{li}(\mathbf{q}) = \frac{\eta_{1,ee}(\mathbf{q}) - P}{|\eta_{1,ee}(\mathbf{q}) - P|} \qquad J_r = J_r(\mathbf{q}) = J_v(\mathbf{q}) \in \mathbb{R}^{3\times n}}$$



Task switching

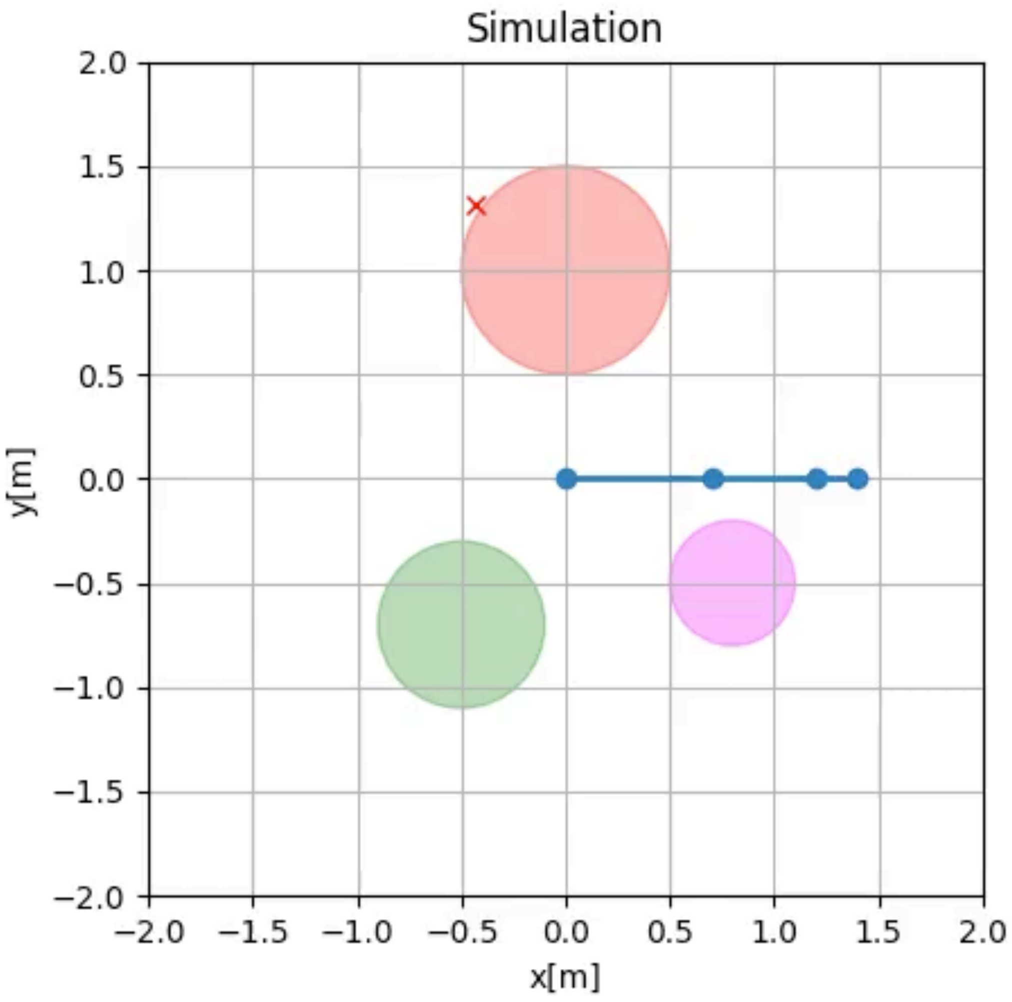
$r_\alpha \in \mathbb{R}$ Activation threshold

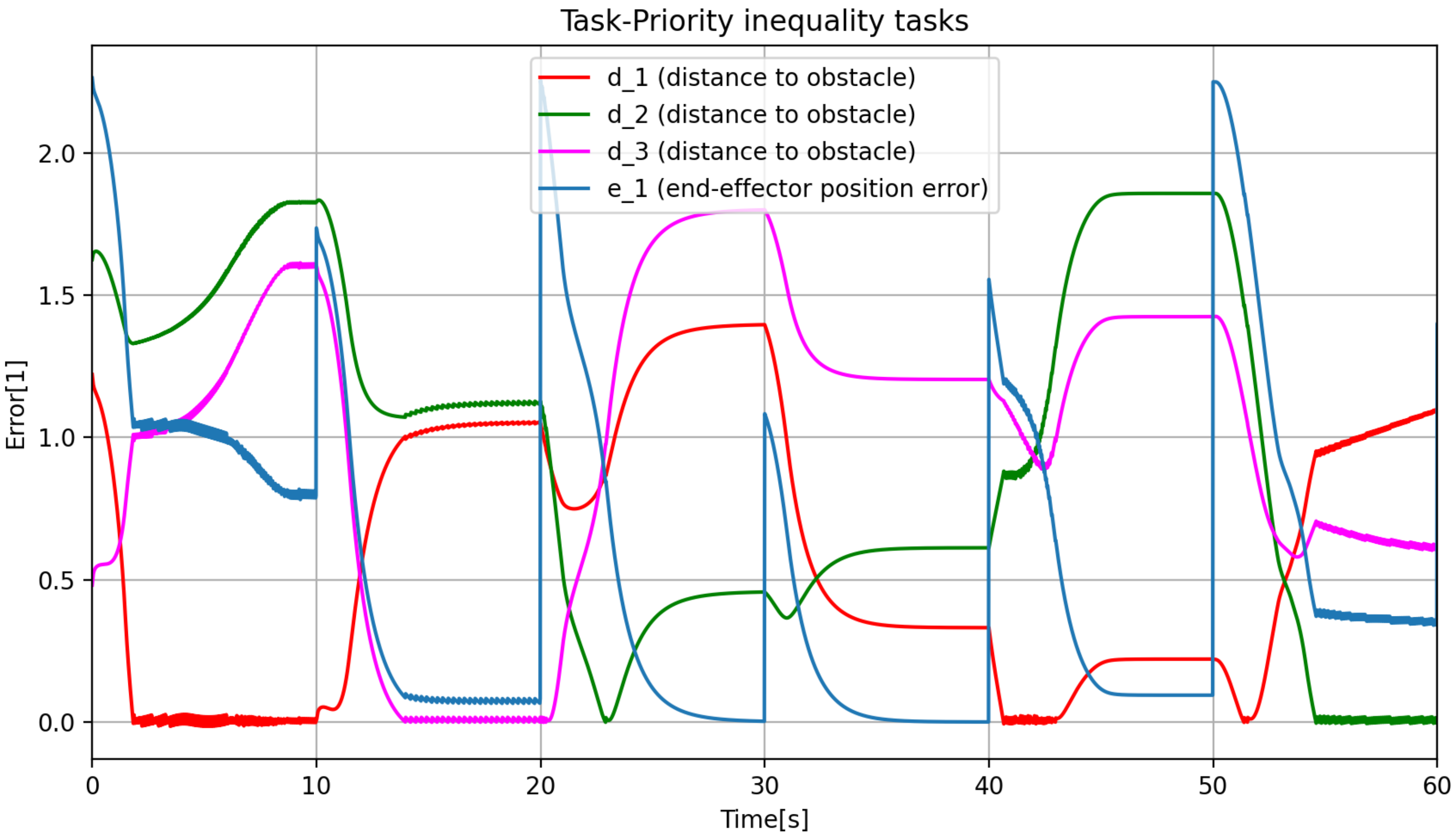
$r_\delta \in \mathbb{R}$ Deactivation threshold

$\boxed{r_\delta > r_\alpha}$ To avoid chatter!

$$a_r(\mathbf{q}) = \begin{cases} 1, & a_r = 0 \wedge |\eta_{1,ee}(\mathbf{q}) - P| \leq r_\alpha \\ 0, & a_r = 1 \wedge |\eta_{1,ee}(\mathbf{q}) - P| \geq r_\delta \end{cases}$$ Activation function

Simulation


Simulation
y[m]
x[m]
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0 −1.5 −1.0 −0.5 0.0 0.5 1.0 1.5 2.0


Task-Priority inequality tasks
d_1 (distance to obstacle)
d_2 (distance to obstacle)
d_3 (distance to obstacle)
e_1 (end-effector position error)
Error[1]
Time[s]
2.0
1.5
1.0
0.5
0.0
0 10 20 30 40 50 60

1.4. Inequality tasks: Extension of the recursive TP

The algorithm

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

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

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

for $i \in 1 \dots k$

    if $a_i(\mathbf{q}) \neq 0$

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

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

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

    else

        $\zeta_i = \zeta_{i-1}, \quad P_i = P_{i-1}$

    end if

end for

return ζ_k

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

k number of tasks

n number of robot DOF

Activation functions are always equal 1 for equality tasks.

Task is omitted.

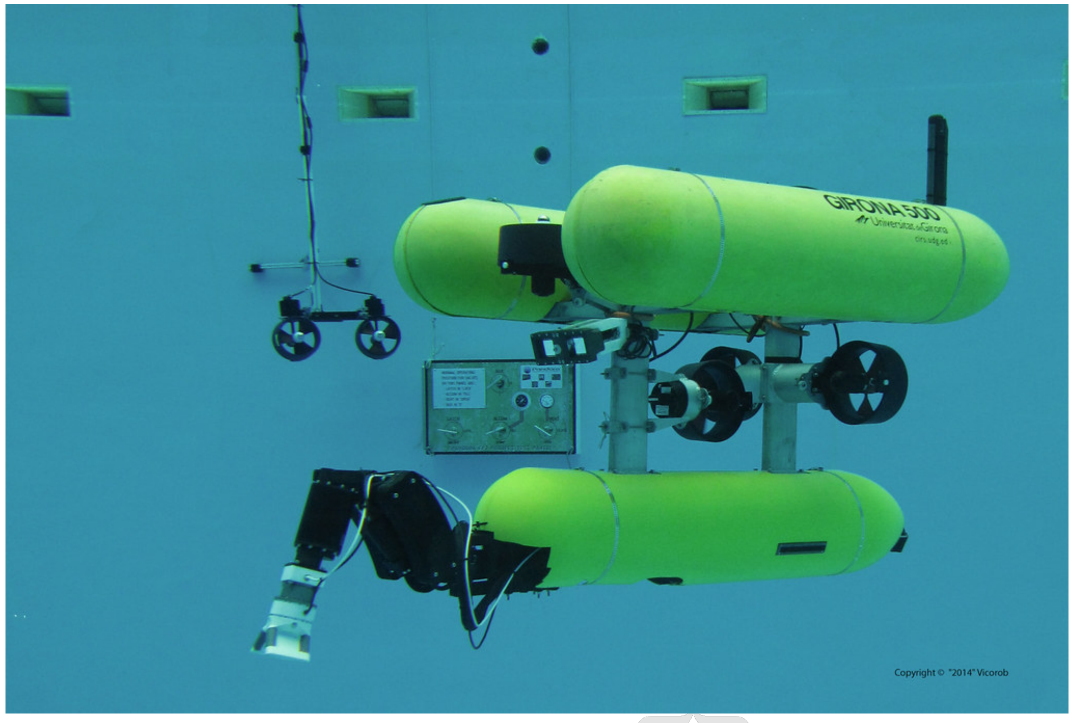
Kinematic constraints (velocity)

$A(q)\dot{q} = 0$ Pfaffian constraints

- *Holonomic* - integrable to configuration constraints
- *Non-holomomic* - not integrable to configuration constraints

Holonomic

All DOF of the robot are controllable, i.e., it is possible to apply arbitrary velocity to every DOF independent of the robot configuration.



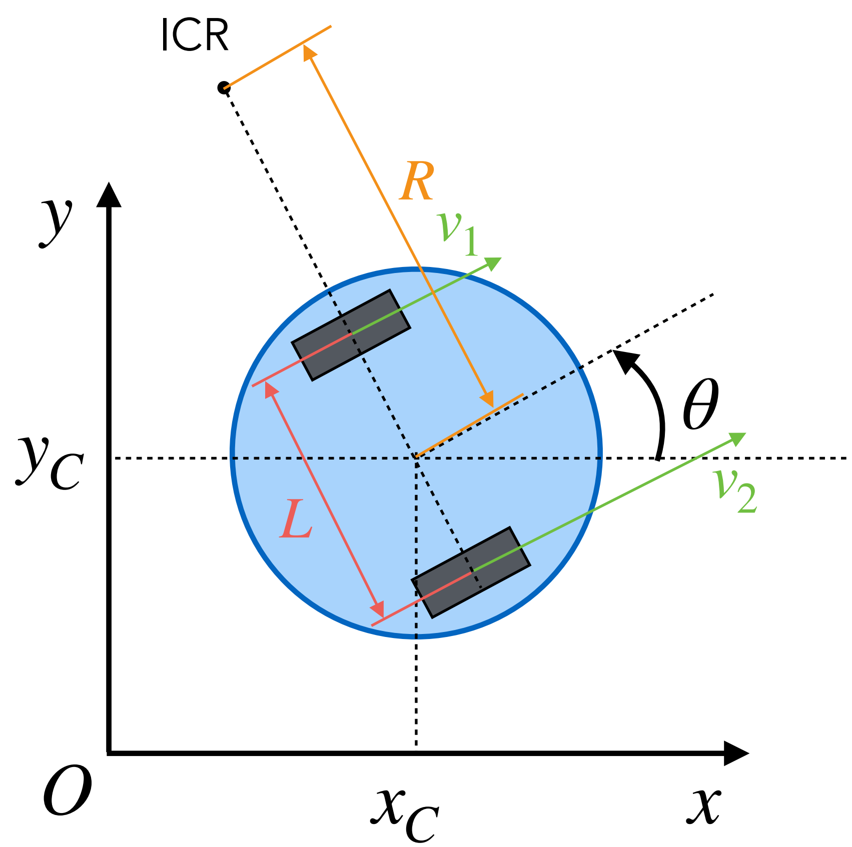
GIRONA500 - 4 DOF (pitch and roll not controlled)

KUKA VMS with Mecanum (Swedish) wheels - 3 DOF

Non-holonomic

The number of controllable DOF is less than the number of DOF of the robot, i.e., it is not possible to apply arbitrary velocity in every DOF; the admissible velocities are constrained and depend on configuration.

Unicycle

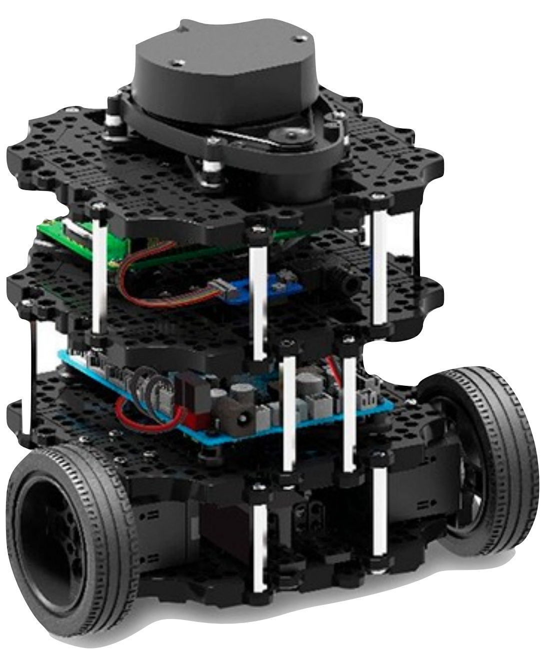


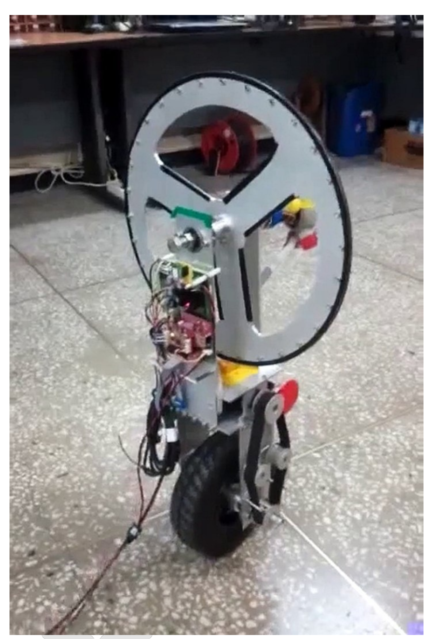
Kinematics

$$q = \begin{bmatrix} x_C & y_C & \theta \end{bmatrix}^T$$

$$\dot{x}_C \sin\theta - \dot{y}_C \cos\theta = \overbrace{\boxed{\begin{bmatrix} \sin\theta & -\cos\theta & 0 \end{bmatrix}}}^{A(q)} \dot{q} = 0$$

Non-holonomic constraint
(robot cannot move laterally without moving forward)





Differential drive

$$\dot{\theta}(R - L/2) = v_1$$

$$\dot{\theta}(R + L/2) = v_2$$

$$R = \frac{L}{2}\frac{v_1 + v_2}{v_2 - v_1}$$

$$\dot{\theta} = \frac{v_2 - v_1}{L}$$

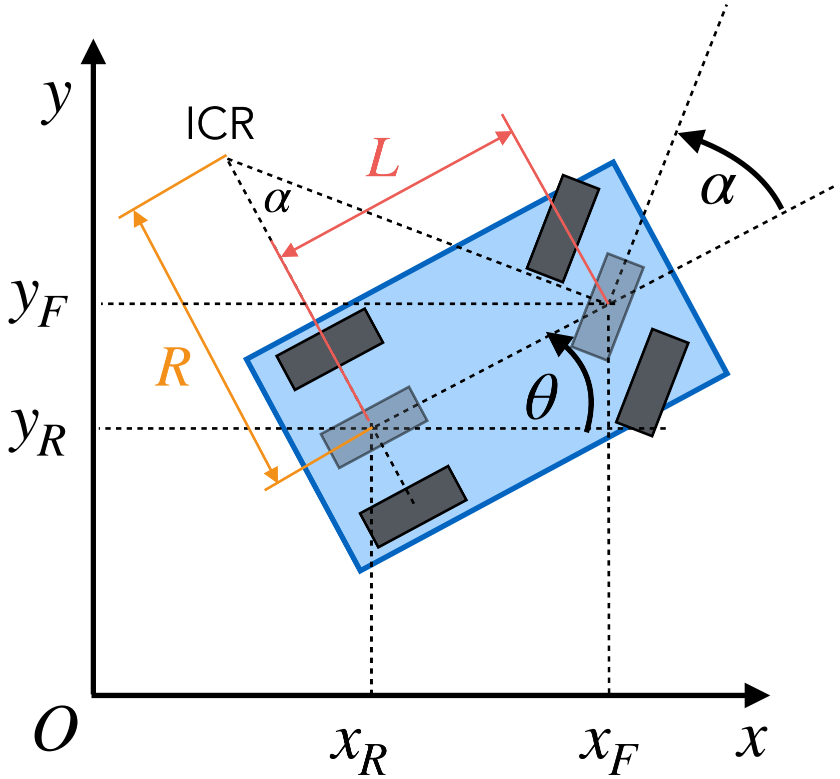
3 special cases:

$v_1 = v_2 \rightarrow R = \infty, \quad \dot{\theta} = 0$

$v_1 = -v_2 \rightarrow R = 0, \quad \dot{\theta} = 2v_2/L$

$v_1 = 0 \rightarrow R = L/2, \quad \dot{\theta} = v_2/L$

Car-like



Kinematics

$$q = \begin{bmatrix} x_R & y_R & \theta & \alpha \end{bmatrix}^T$$

$$\dot{x}_R \sin\theta - \dot{y}_R \cos\theta = 0$$
$$\dot{x}_F \sin\theta - \dot{y}_F \cos\theta = 0$$

$$x_F = x_R + L\cos\theta, \quad \dot{x}_F = \dot{x}_R - L\sin\theta$$
$$y_F = y_R + L\sin\theta, \quad \dot{y}_F = \dot{y}_R + L\cos\theta$$

$$\dot{x}_R \sin\theta - \dot{y}_R \cos\theta = 0$$
$$\dot{x}_R \sin(\theta+\alpha) - \dot{y}_R \cos(\theta+\alpha) - L\dot{\theta}\cos\alpha = 0$$

$$\underbrace{\boxed{\begin{bmatrix} \sin\theta & -\cos\theta & 0 & 0 \\ \sin(\theta+\alpha) & -\cos(\theta+\alpha) & -L\cos\alpha & 0 \end{bmatrix}}}_{A(q)} \dot{q} = 0$$

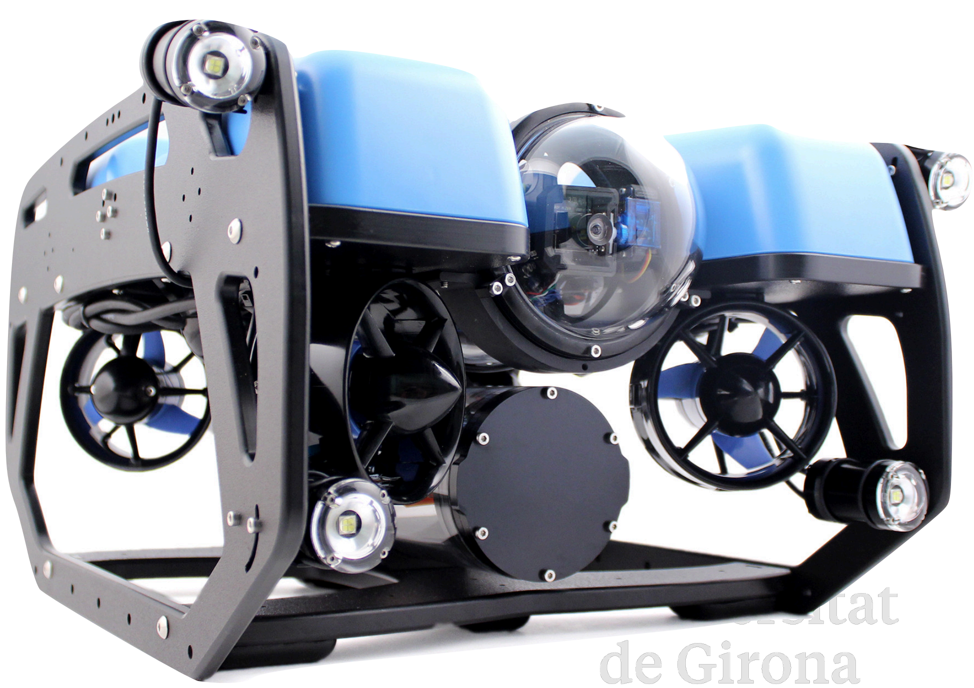
Rear-wheel drive

$$\dot{x}_R = v\cos\theta, \quad \dot{y}_R = v\sin\theta$$

$$\frac{L}{R} = \tan\alpha \implies \dot{\theta} = v\frac{\tan\alpha}{L}$$

Redundant

The number of drives (thrusters, propellers) is more than the number of DOF of the robot, i.e., it is possible to achieve the same velocity using different combination of drive velocities.



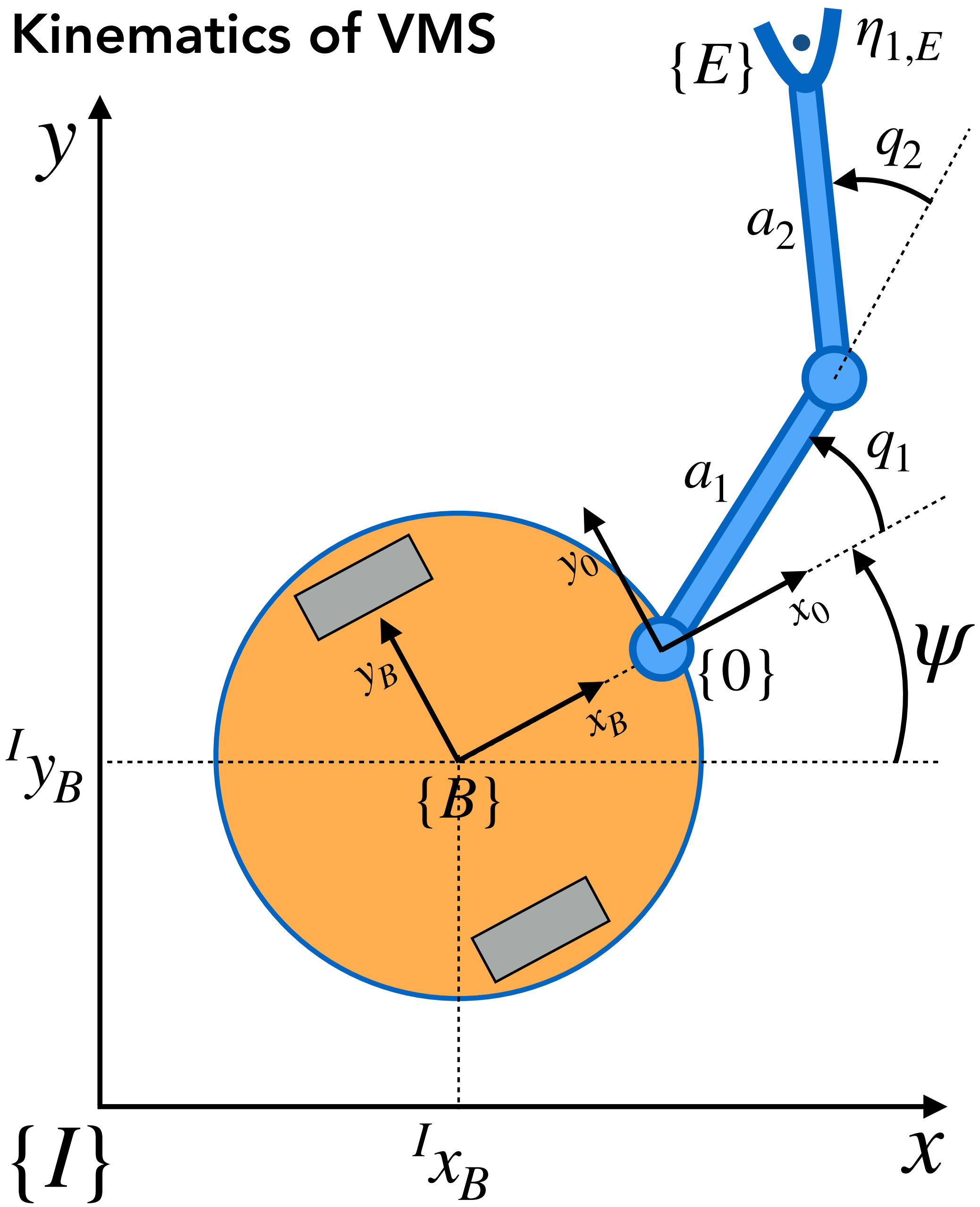
Pros and cons

- **Non-holomomic**
  + Requires less drives than DOF to reach all configurations in the workspace.
  - Not possible to command arbitrary velocity in all DOF (possible velocities depend on configuration).
  - Control and motion planning is hard (the robot cannot move between arbitrary points following a line).

- **Holonomic**
  + It is possible to command arbitrary velocity in each DOF.
  + Control and motion planning is easy (the robot can move between arbitrary points following a line).
  - Requires as many drives as DOF to reach all configurations in the workspace.

- **Redundant**
  + It is possible to command arbitrary velocity in each DOF.
  + Motion planning is easy (the robot can move between arbitrary points following a line).
  + In case of drive failure it may be possible to still actuate all DOF.
  + Redundancy can be used to increase available force/torque, improve stability, change operating point of drives, reduce response time, etc.
  + Control is harder than for holonomic systems because it requires taking into account the redundancy.
  - Requires more drives than DOF which increases complexity, power consumption, weight, costs.

Kinematics of VMS



Base pose $\eta = [\eta_1 \ \eta_2]^T = [x \ y \ z \mid \phi \ \theta \ \psi]^T \in \mathbb{R}^{6\times1}$

Manipulator joint positions $q \in \mathbb{R}^{n\times1}$

VMS configuration vector $\mathfrak{q} = [\eta^T \ q^T]^T \in \mathbb{R}^{6+n\times1}$

Mobile base transformation $^I T_B = \begin{bmatrix} ^I R_B & ^I\eta_1 \\ 0_{1\times3} & 1 \end{bmatrix}$

$$^I R_B = R_z(\psi) R_{y'}(\theta) R_{x''}(\phi) =$$

$$= \begin{bmatrix} c(\psi)c(\theta) & s(\psi)c(\theta) & -s(\theta) \\ -s(\psi)c(\phi) + c(\psi)s(\theta)s(\phi) & c(\psi)c(\phi) + s(\psi)s(\theta)s(\phi) & s(\phi)c(\theta) \\ s(\psi)s(\phi) + c(\psi)s(\theta)c(\phi) & -c(\psi)s(\phi) + s(\psi)s(\theta)c(\phi) & c(\phi)c(\theta) \end{bmatrix}$$

Manipulator base transformation $^B T_0 = \begin{bmatrix} ^B R_0 & ^B\eta_{1,0} \\ 0_{1\times3} & 1 \end{bmatrix}$

Manipulator frames (Denavit-Hartenberg) $^{n-1}T_n = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c(q_i) & -s(q_i) & 0 & 0 \\ s(q_i) & c(q_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c(\alpha_i) & -s(\alpha_i) & 0 \\ 0 & s(\alpha_i) & c(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

End-effector $^I T_E = {}^I T_B \, {}^B T_0 \prod_{i=1}^{n} {}^{i-1}T_i$

Differential kinematics of VMS

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

Jacobian

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

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

Mobile base velocity (in body frame)

Manipulator joint velocities

We know an algorithm to compute $J(\mathbf{q})$ based on the kinematic transformations along the robotic chain.

$$\Downarrow$$

We can treat the mobile base like it was part of the manipulator!

- **Linear and angular velocities** used to **control** the robot are expressed in the **body frame** $\{B\}$**.**
- Mobile base transformations have to **start with rotational joints** representing the **rate of change of RPY**, **followed by linear joints** representing **linear velocities** in body frame.
- **Position of the base** in world frame η_1 is omitted as it **does not influence velocities** of the end-effector.

$\dot{\psi}$	$\dot{\theta}$	$\dot{\phi}$	v_x	v_y	v_z	$\dot{q}$
$R_z(\psi)R_x(-90^o)$	$R_z(\theta + 90^o)R_x(90^o)$	$R_z(\phi)$	$T_z(0)R_x(-90^o)$	$T_z(0)R_z(-90^o)R_x(90^o)$	$T_z(0)^BT_0$	$DH(d, q, a, \alpha)$

Problem

- **Manipulator can follow accurately** kinematically generated trajectories with **high speed and acceleration** while the **mobile base** has a much **slower response time** (high inertia) and generally **lower accuracy** of the velocity feedback.
- Kinematic control (like the Task-Priority algorithm) does not account for any dynamical effects by design.
- **Base dynamics** are often **much harder to model accurately than dynamics of the manipulator** chain so the velocity control loop will exhibit higher errors and longer stabilisation time.
- A **typical effect** encountered when kinematically controlling the VMS is **lack of convergence or slow convergence** of the end-effector position and **undamped oscillations**.

Solution

- The **simplest solution** is to **slow down the whole system significantly**, which results in improved stability at the expense of **longer mission time**.
- A **smarter solution** is to **slow down the system based on the distance to the goal** and avoid unnecessary motions of the base if the goal is in the workspace of the manipulator.

Observations

- **Velocity control loops of the manipulator** are often running at a **much higher rate than the ones of the mobile base** (it is possible to command velocity to the manipulator at a higher rate).
- **Manipulator** can achieve **higher velocities and accelerations** than the base.
- **Manipulator** is **more accurate** than the base.
- **Motion of the mobile base** can be severely **impacted by environmental disturbances**, especially if it is a floating base (AUV, UAV).

Possible improvement: multi-rate control

- The idea is to **control the manipulator at a higher frequency than the base**.
- The pose of the base should be measured at the same frequency as the control of the manipulator.
- The TP algorithm is used to compute desired velocities of the whole system at a low frequency and desired velocities of the manipulator only at a high frequency. During the second type of update the base is treated as fixed.
- **Manipulator can compensate for the disturbances acting on the base** with fast corrections.