

Adapting the Servo-constraints Model Inversion Method for Robotic Cable Manipulation

IMSD conference, June 17, 2026

Louis Dehaybe, Olivier Brûls

Multibody and Mechatronic Systems Lab
Department of Aerospace and Mechanical Engineering
University of Liège, Belgium

Deformable objects manipulation: why ?



Interesting challenges

Complex non-linear dynamics, highly underactuated systems



Pathway

to manipulate anything



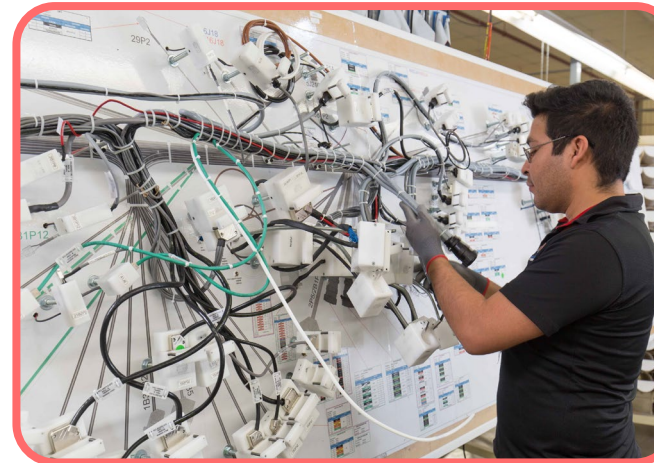
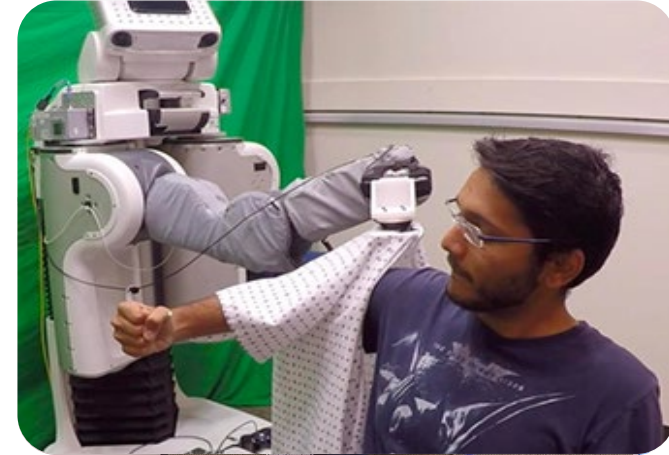
Numerous applications

Focus on cables (1D objects)

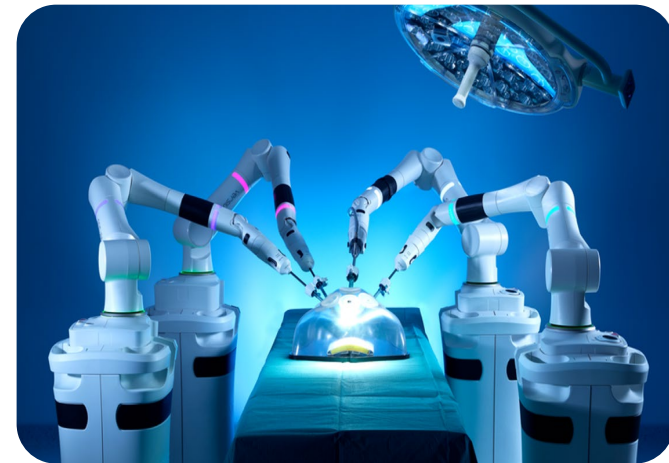
Composites fabrics



Home robotics



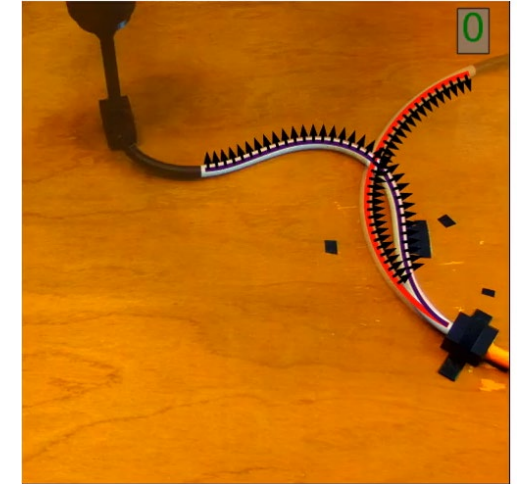
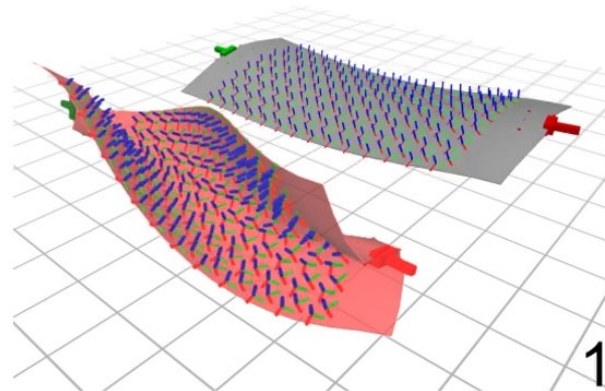
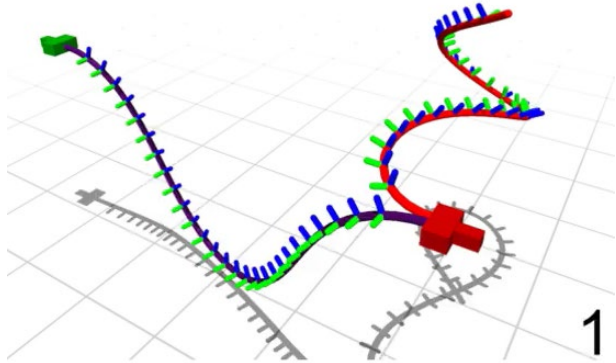
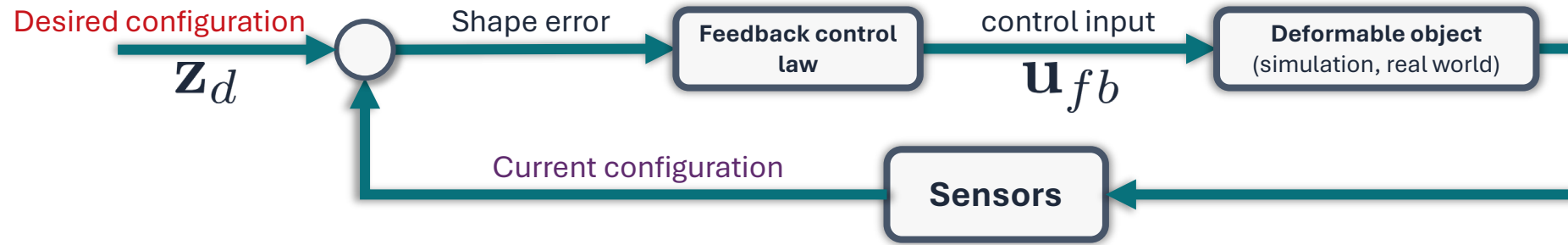
Cables manipulation



Surgery

Background (previous work [1])

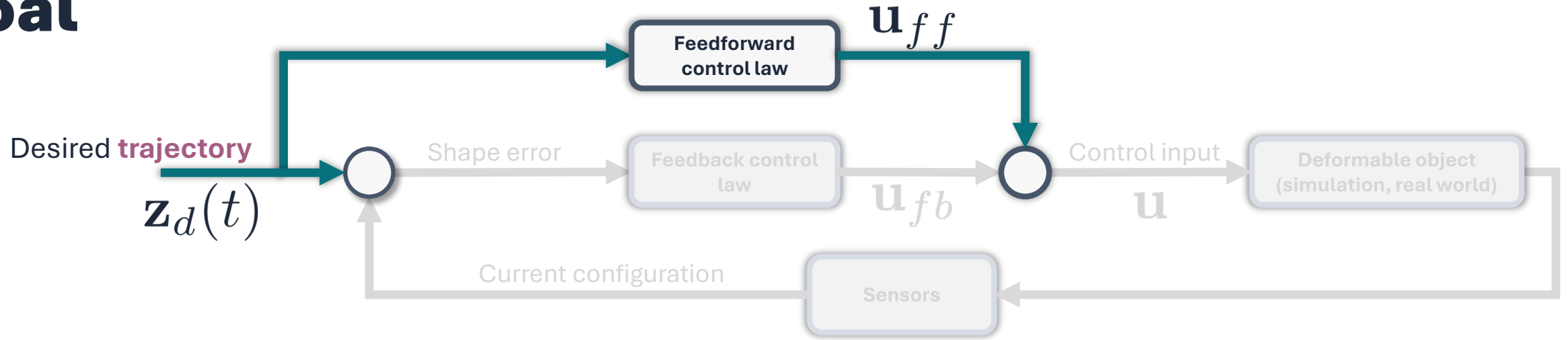
Model-free, closed-loop control via sensor feedback



Quasi-static manipulation only:

- Restricts the **scope** of feasible tasks
- Imposes **slow motion**
- Requires **constant feedback** from sensors

Goal



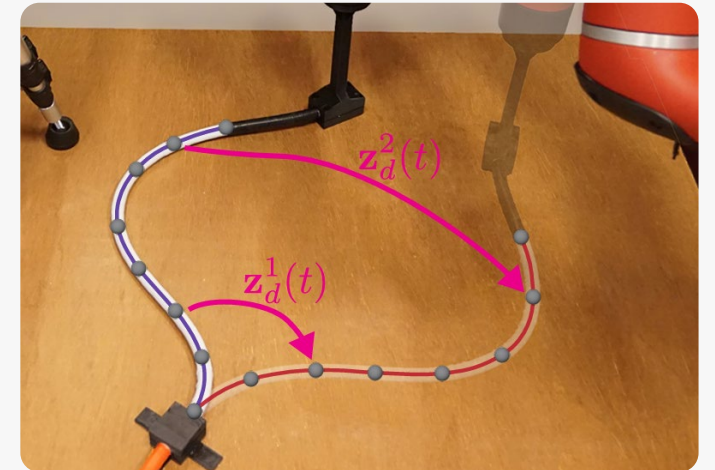
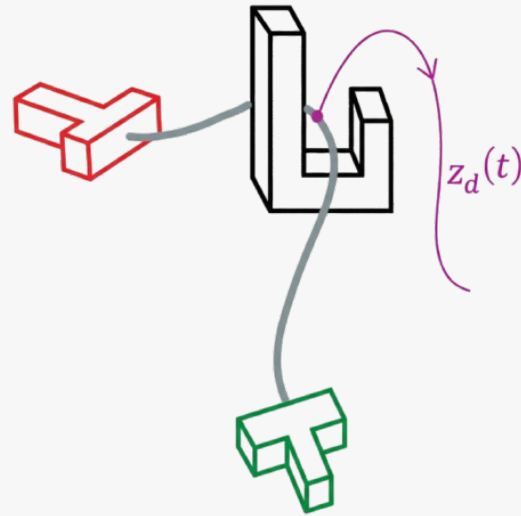
Compute **feedforward input** via servo-constraints and compare to optimal control



Enable **dynamic** ($><$ quasi-static) manipulations



Precomputed control input: no need for constant sensor feedback



Inverse dynamics via the servo-constraints approach

Cable modeled as multibody system with absolute coordinates $\mathbf{q} \in \mathbb{R}^{n_q}$ and control inputs $\mathbf{u} \in \mathbb{R}^{n_u}$

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}^T(\mathbf{q})\boldsymbol{\lambda} = \mathbf{B}\mathbf{u} \quad (1) \quad \text{(Dynamics)}$$

$$\mathbf{g}(\mathbf{q}) = \mathbf{0}_{n_g} \quad (2) \quad \text{(Holo. constraints)}$$

Servo-constraints constrain the system output to follow the desired trajectory

$$\mathbf{s}(\mathbf{q}, t) = \mathbf{h}(\mathbf{q}) - \mathbf{z}_d(t) = \mathbf{H}\mathbf{q} - \mathbf{z}_d(t) \quad (3) \quad \text{(Servo constraints)}$$

where $\mathbf{H}$ is (in our case) a constant selection matrix and $\mathbf{s} \in \mathbb{R}^{n_z}$

DAEs (1)-(3) solved numerically via implicit integration scheme (e.g., generalized- α)

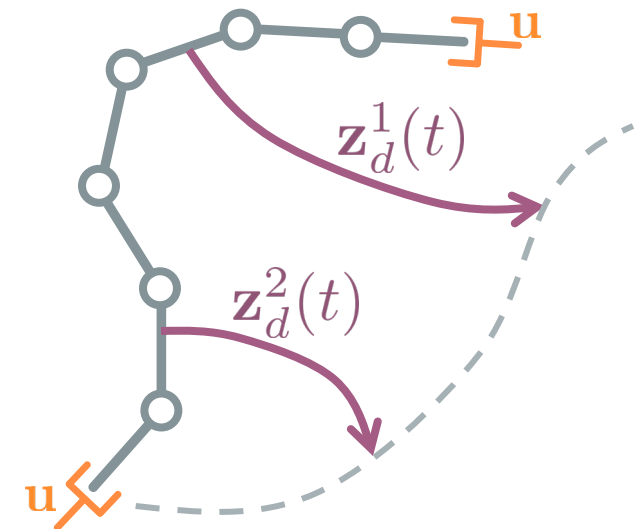
Output: $\mathbf{q}(t)$, $\boldsymbol{\lambda}(t)$, and the **feedforward control input $\mathbf{u}(t)$** producing the **desired motion**



Literature [2,3]: requires $n_z = n_u$ (square), flat or minimum phase systems, smooth and feasible trajectory

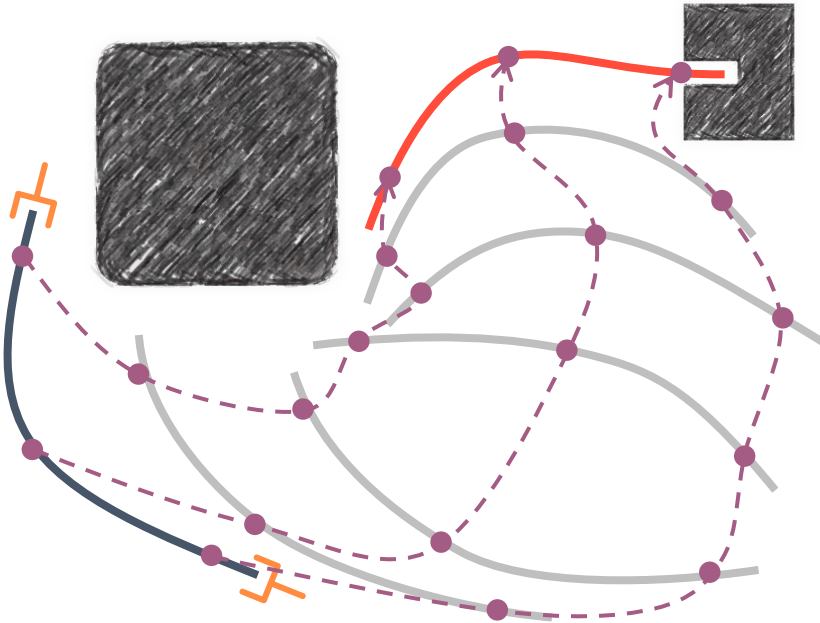
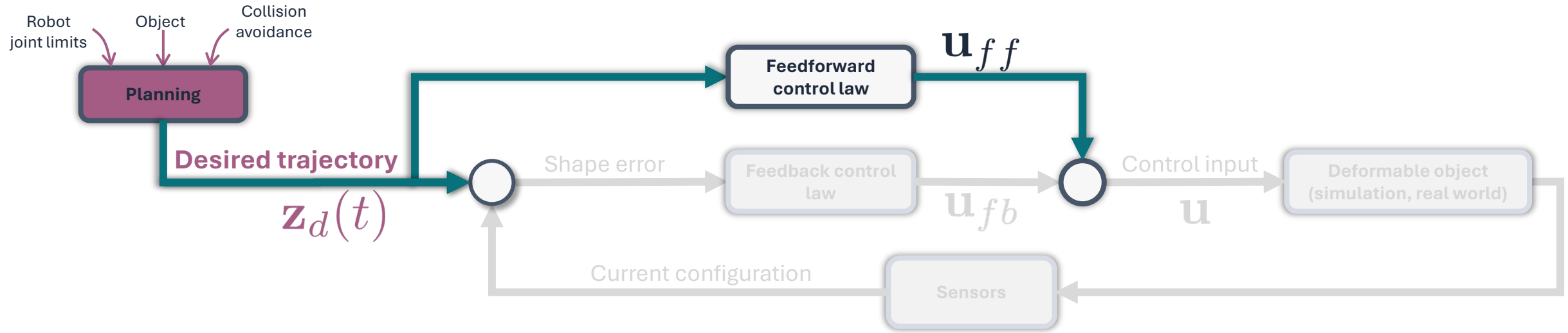
[2] Blajer, W., & Kołodziejczyk, K. (2004). Multibody System Dynamics

[3] Drücker, S., & Seifried, R. (2023). Multibody system dynamics



Difficulties in the context of cables manipulation

1) Non feasible trajectories



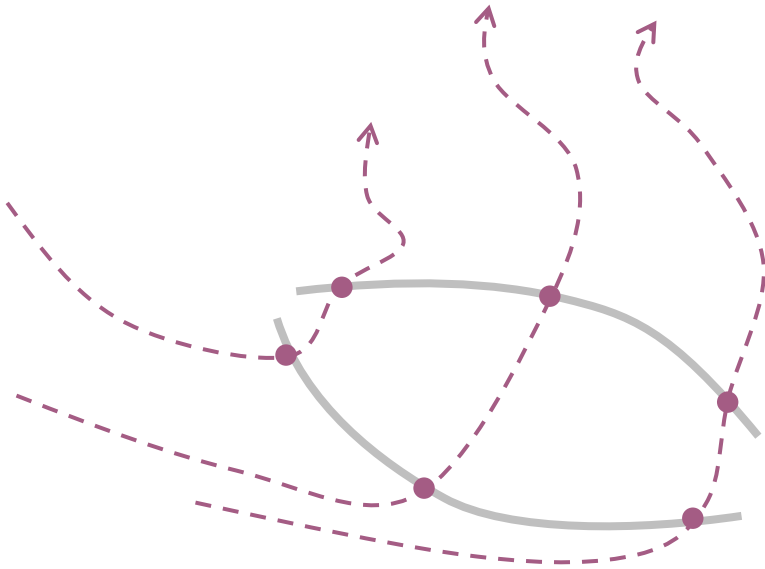
- Robotics: complex environments → Sampling-based planners (SBP): RRTs, PRMs, etc.
- Return a « **guiding path** »
- Interpolate to create a continuous **desired trajectory**

Difficulties in the context of cables manipulation

1) Non feasible trajectories

SBPs: simplified, purely geometric methods or kinematic models

No guarantee that the reference trajectory $\mathbf{z}_d(t)$ is dynamically feasible



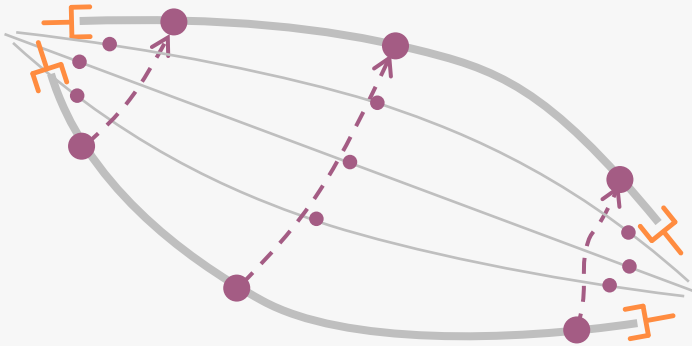
Difficulties in the context of cables manipulation

1) Non feasible trajectories

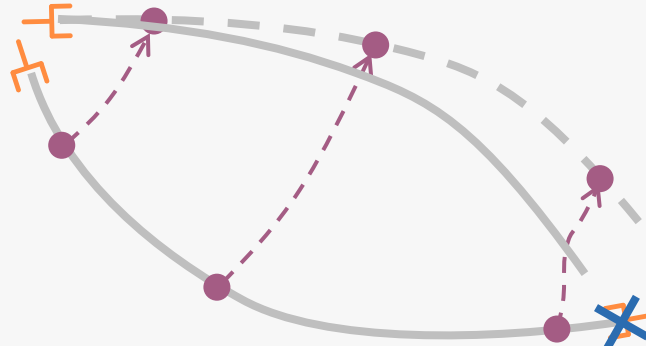
SBPs: simplified, purely geometric methods or kinematic models

No guarantee that the reference trajectory $\mathbf{z}_d(t)$ is dynamically feasible

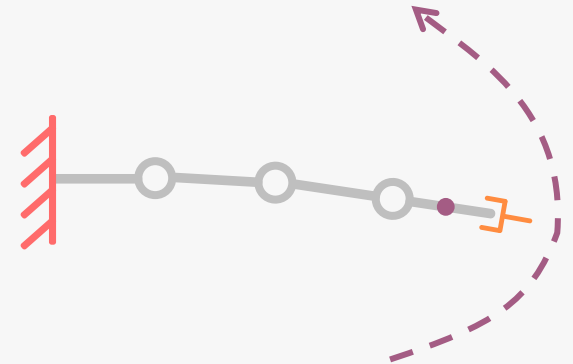
Incompatible trajectories



Lack of control authority (B)



Conflict with holonomic constraints $g(\mathbf{q}) = 0$

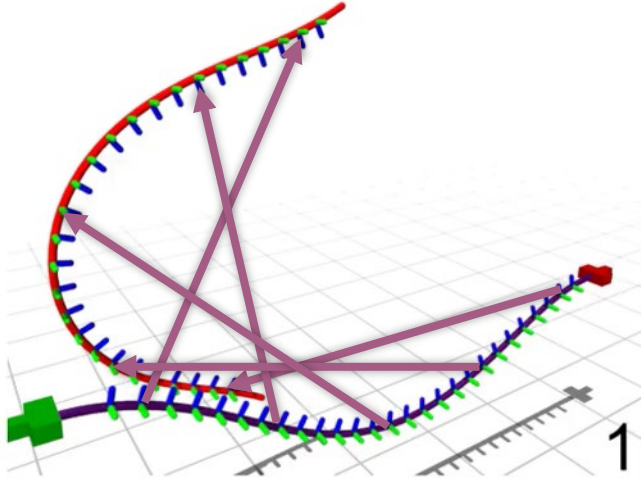


$\mathbf{s}(\mathbf{q}, t) = \mathbf{0}$ can't be fulfilled while satisfying eq. (1) and (2)

Goal: find a transformation $\mathbf{s}_\Phi = \Phi(\mathbf{s})$ such that only the error that can be reduced to zero remains

Difficulties in the context of cables manipulation

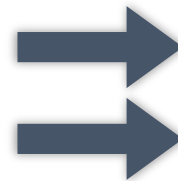
2) Non-square systems



Servo constraints approach requires
Inputs (n_u) = # Outputs (n_z)

In practice, two other situations:

1. Under-constrained : $n_u > n_z$
2. Over-constrained (typical case) : $n_u < n_z$



In this case, we would like...

- ... to minimize control inputs $\|u\|$
 - ... the least-squares solution (« best-fit »)
- « pseudo-inverse behavior »

Goal: find a transformation $s_\Phi = \Phi(s)$ s.t. $s_\Phi \in \mathbb{R}^{n_u}$ and s.t. solving for $s_\Phi = 0$
minimizes the error S in some least squares sense

Method: modification of the servo-constraints

1. Servo-constraints redefinition

Acceleration level (reduces relative degree by 2):

$$s(\mathbf{q}, t) = \mathbf{H}\mathbf{q} - \mathbf{z}_d(t) \longrightarrow \ddot{s}(\mathbf{q}, t) = \mathbf{H}\ddot{\mathbf{q}} - \ddot{\mathbf{z}}_d(t)$$

Satisfying $\ddot{s} = \mathbf{0} \ \forall t$ is not always possible, + numerical drift, we defined the **desired acceleration**

$$\mathbf{a}_d = \ddot{\mathbf{z}}_d + k_d(\dot{\mathbf{z}}_d - \mathbf{H}\dot{\mathbf{q}}) + k_p(\mathbf{z}_d - \mathbf{H}\mathbf{q})$$

2. Error minimization

Find the control inputs that **minimizes the error**

$$\min_u E = \frac{1}{2} \|\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d\|^2 + \frac{1}{2} w_u \|\mathbf{u}\|^2$$

(encourage minimal input solutions)

Stationarity condition

$$\frac{\partial E}{\partial u} = \frac{\partial(\mathbf{H}\ddot{\mathbf{q}})^T}{\partial u} (\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d) + w_u \mathbf{u} = 0$$

3. Projection matrices

It can be shown that

$$\begin{aligned} \mathbf{H}\ddot{\mathbf{q}} &= \mathbf{H}\mathbf{a}_c + \mathbf{J}_u \mathbf{u} \\ \longrightarrow \frac{\partial E}{\partial u} &= \mathbf{J}_u^T (\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d) + w_u \mathbf{u} = 0 \end{aligned}$$

where:

$\mathbf{a}_c$ is the acceleration of the constrained system if there were no control inputs

$$\mathbf{J}_u = \mathbf{H}\mathbf{P}\mathbf{M}^{-1}\mathbf{B}$$

- $\mathbf{P} = \mathbf{I} - \overline{\mathbf{G}}\mathbf{G}$ dynamically consistent **projection matrix onto the null space of $\mathbf{G}$**
→ Guarantees that control inputs **only produce valid motions on the constraint manifold**, causing no acceleration against holonomic constraints
- $\overline{\mathbf{G}} = \mathbf{M}^{-1}\mathbf{G}^T(\mathbf{G}\mathbf{M}^{-1}\mathbf{G}^T)^{-1}$
the dynamically consistent generalized inverse of $\mathbf{G}$

Method: modification of the servo-constraints

$$\mathbf{M}\ddot{\mathbf{q}} + \mathbf{f}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}^T(\mathbf{q})\boldsymbol{\lambda} = \mathbf{B}\mathbf{u}$$

(1) (Dynamics)

$$\mathbf{g}(\mathbf{q}) = \mathbf{0}_{n_g}$$

(2) (Holonomic constraints)

$$\mathbf{s}_\Phi = \mathbf{J}_u^T (\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d) + w_u \mathbf{u} = \mathbf{0}$$

(4) (Modified servo constraints)

1. Optimality embedding

By replacing the servo-constraint $\mathbf{s} = \mathbf{0}$ by $\mathbf{s}_\Phi = \mathbf{0}$, the first order **optimality condition**

$$\frac{\partial E}{\partial \mathbf{u}} = 0$$

is **directly embedded in the DAE solver**, i.e. solving for (1), (2), (4) will lead to the control input that **minimizes**

$$\|\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d\|^2 + \frac{1}{2}w_u \|\mathbf{u}\|^2$$

Prevents the DAE solver to solve a potentially impossible root-finding problem

2. No more non-square systems

$$\mathbf{s}_\Phi = \underbrace{\mathbf{B}^T \mathbf{P} \mathbf{M}^{-1} \mathbf{H}^T}_{\mathbf{J}_u^T : n_u \times n_z} (\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d) + w_u \mathbf{u}$$

Dimensions: $\mathbf{B}^T (n_u \times n_q)$, $\mathbf{P} (n_q \times n_q)$, $\mathbf{M}^{-1} (n_q \times n_z)$, $\mathbf{H}^T (n_q \times n_q)$



The transformed servo-constraints $\mathbf{s}_\Phi$ always

n_u equations and n_u control inputs
Square system to solve

Method: solving the DAEs

Solving the DAEs via generalized- α requires a **linearization for the Newton algorithm**. From eq. (4):

$$\mathbf{s}_{\Phi}^{lin} = \mathbf{s}_{\Phi} + \frac{\partial(\mathbf{J}_u^T)}{\partial \mathbf{q}} (\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d) \Delta \mathbf{q} + \mathbf{J}_u^T k_p \mathbf{H} \Delta \mathbf{q} + \mathbf{J}_u^T k_d \mathbf{H} \Delta \dot{\mathbf{q}} + \mathbf{J}_u^T \mathbf{H} \Delta \ddot{\mathbf{q}} + w_u \mathbf{I} \Delta \mathbf{u}$$



Cumbersome and expensive to compute, since

$$\mathbf{J}_u = \mathbf{H} \mathbf{P} \mathbf{M}^{-1} \mathbf{B}$$
$$\mathbf{P} = \mathbf{I} - \overline{\mathbf{G}} \mathbf{G}$$
$$\overline{\mathbf{G}} = \mathbf{M}^{-1} \mathbf{G}^T (\mathbf{G} \mathbf{M}^{-1} \mathbf{G}^T)^{-1}$$

In practice

Option 1

Use Casadi



Automatic differentiation

✓ Exact derivative cheaply

Option 2

Gauss-Newton approximation

Ignore this (second-order) term

Inexact linearization, but Newton iterations are faster

Option 3

Finite-differences

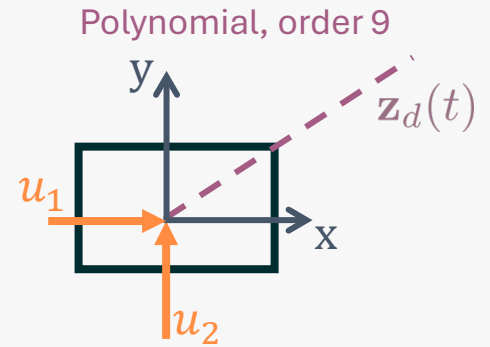
Slow !

Results (trivial example): **feasible task**

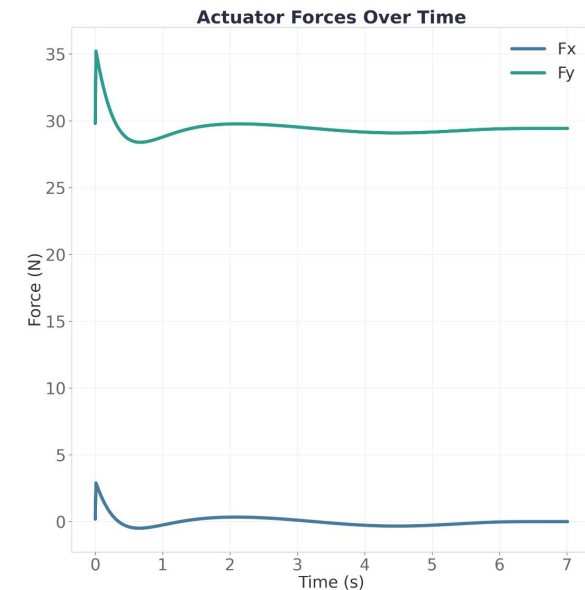
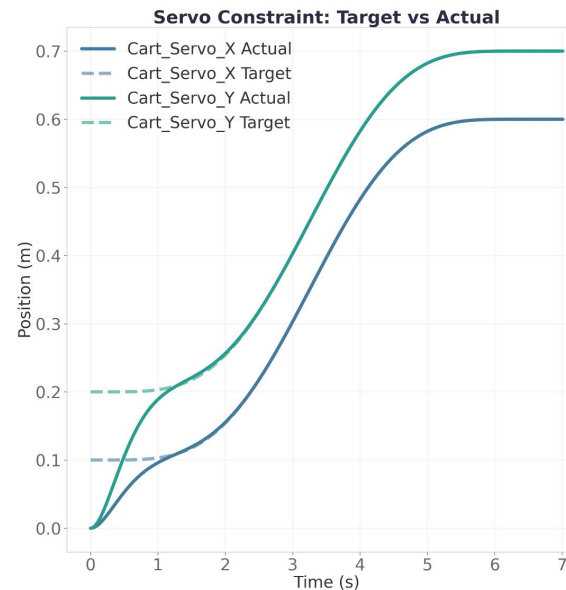
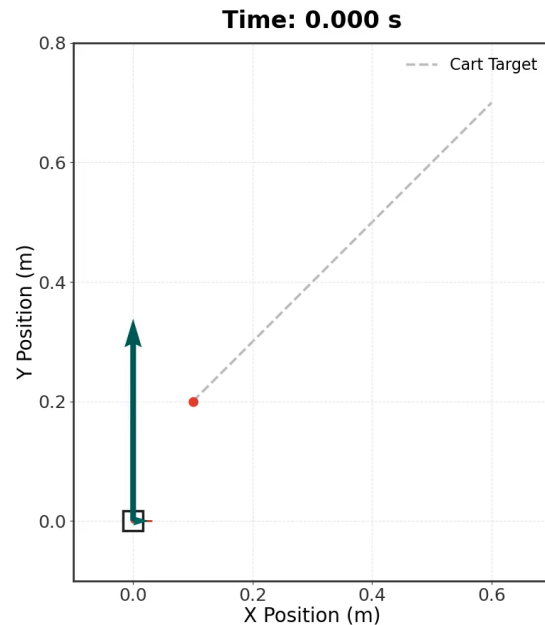
$$\mathbf{q} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mathbf{z}_d(t) = \begin{bmatrix} x_d(t) \\ y_d(t) \end{bmatrix} \quad \mathbf{h}(\mathbf{q}) = \mathbf{H}\mathbf{q} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{q} \quad \mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

No constraints $\rightarrow$ Projection $\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Consider a given error $\mathbf{e} = \begin{bmatrix} e_x \\ e_y \end{bmatrix} \rightarrow \mathbf{s}_\Phi = \underbrace{\mathbf{B}^T \mathbf{P} \mathbf{M}^{-1} \mathbf{H}^T}_{\mathbf{J}_u^T} \underbrace{(\mathbf{H}\ddot{\mathbf{q}} - \mathbf{a}_d)}_{\mathbf{e}} = \begin{bmatrix} e_x/m \\ e_y/m \end{bmatrix}$



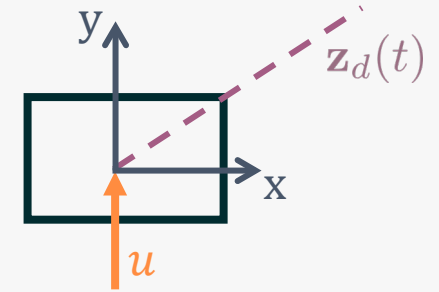
The transformation $\mathbf{J}_u^T$ projects no component of the error to 0 when the task is feasible $\rightarrow$ **Good**



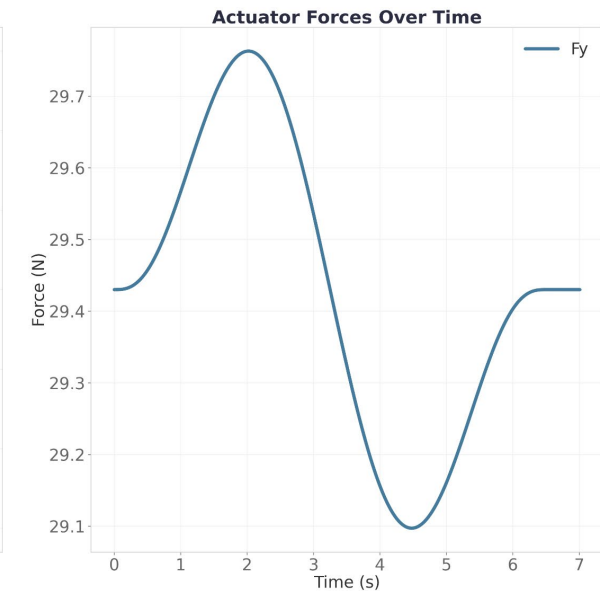
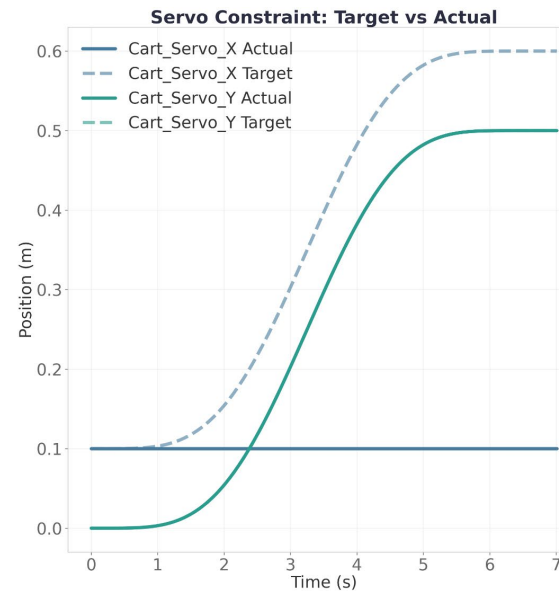
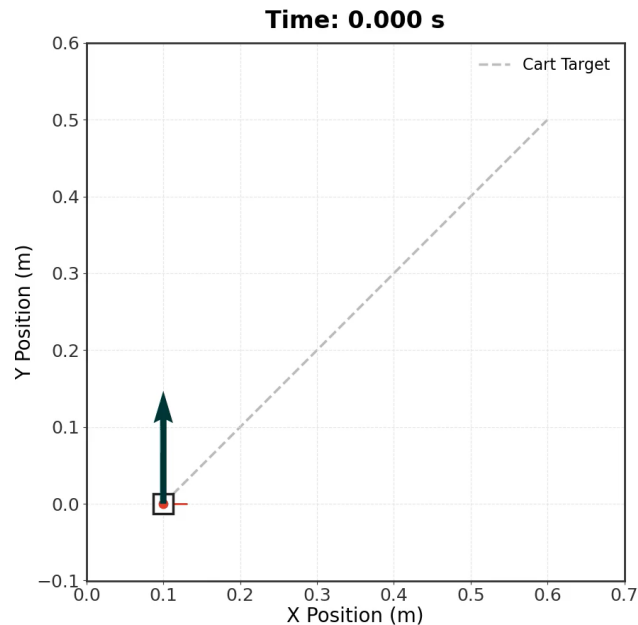
Results (trivial example):

$$\mathbf{B} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{No constraints} \rightarrow \text{Projection } \mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$\mathbf{e} = \begin{bmatrix} e_x \\ e_y \end{bmatrix} \rightarrow \mathbf{s}_\Phi = \mathbf{B}^T \mathbf{P} \mathbf{M}^{-1} \mathbf{H}^T \mathbf{e} = \mathbf{B}^T \begin{bmatrix} e_x/m \\ e_y/m \end{bmatrix} = \begin{bmatrix} 0 \\ e_y/m \end{bmatrix}$$



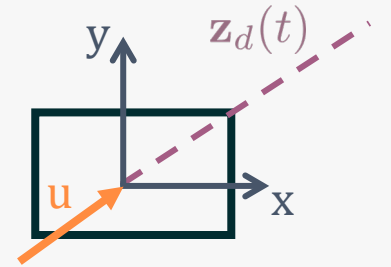
- The error e_x can not be reduced **due to lack of actuation in this direction (B matrix)** $\rightarrow$ **Projected to 0**
- $n_u = 1$, but $\mathbf{s}_\Phi \in \mathbb{R} \rightarrow$ **#input = #output (square)**



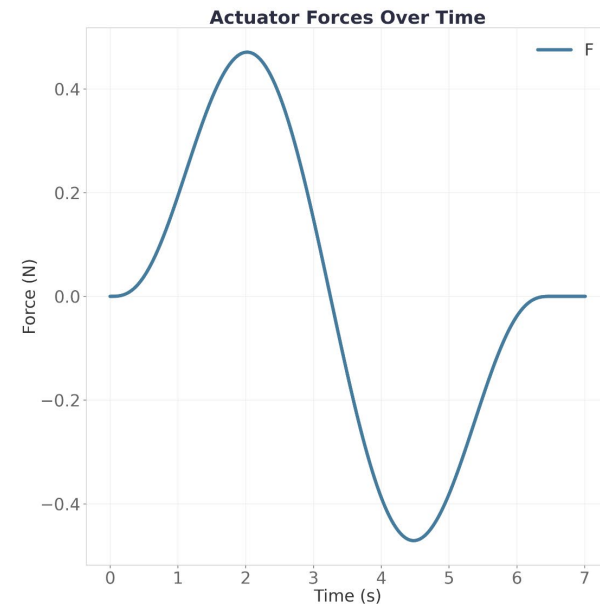
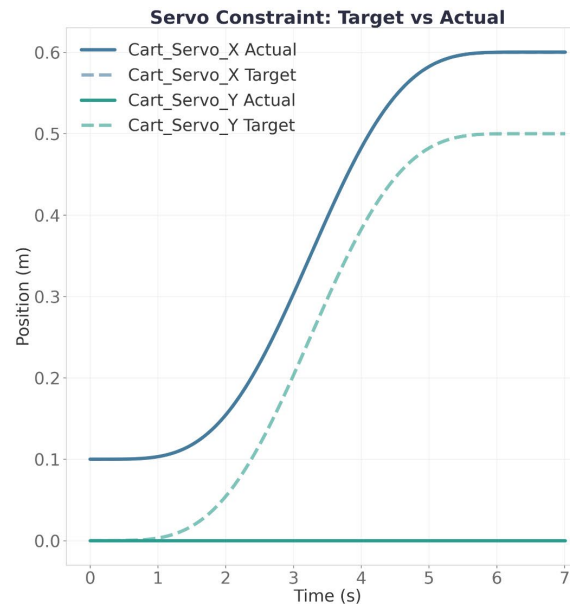
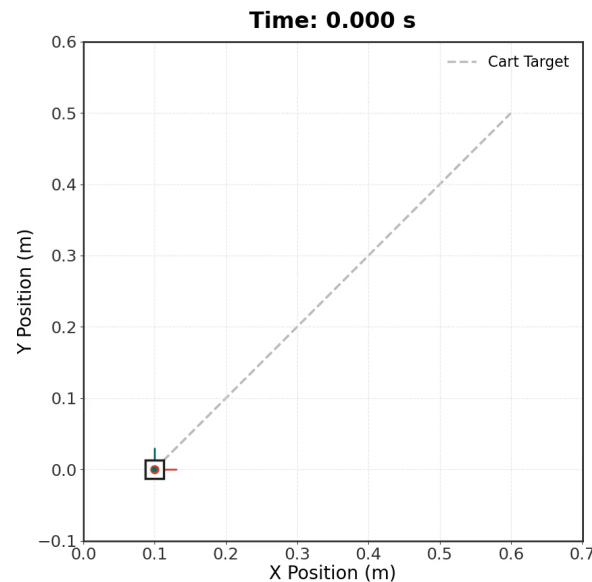
Results (trivial example): Incompatible with constraints

$$\mathbf{B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad \text{Constraint: } \mathbf{g}(\mathbf{q}) = y = 0 \rightarrow \mathbf{G} = \begin{bmatrix} 0 & 1 \end{bmatrix} \rightarrow \mathbf{P} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\mathbf{s}_\Phi = \mathbf{B}^T \mathbf{P} \mathbf{M}^{-1} \mathbf{H}^T \mathbf{e} = \mathbf{B}^T \mathbf{P} \begin{bmatrix} e_x/m \\ e_y/m \end{bmatrix} = \mathbf{B}^T \begin{bmatrix} e_x/m \\ 0 \end{bmatrix} = \begin{bmatrix} e_x/m \\ 0 \end{bmatrix}$$



The error e_y can not be reduced due to the constraint $\mathbf{g}(\mathbf{q}) = 0 \rightarrow$ Projected to 0



Results: comparison with OC (multiple shooting)

$$\min_{\substack{\mathbf{x}^0, \dots, \mathbf{x}^N \\ \mathbf{u}^0, \dots, \mathbf{u}^{N-1}}} J = \Delta t \sum_{k=0}^{N-1} \left(\|\mathbf{h}(\mathbf{q}^k) - \mathbf{y}(t^k)\|^2 + \beta \|\mathbf{u}^k\|^2 \right) + \text{Extra}$$

 Generalized- α integration (or Casadi collocation + index reduction via Baumgarte)

s.t. $\mathbf{f}(\mathbf{x}^k, \mathbf{u}^k) - \mathbf{x}^{k+1} = 0 \quad \forall k = 0, \dots, N-1$

$\mathbf{x}^0 - \bar{\mathbf{x}}^0 = 0 \quad (\text{Initial condition})$

$\mathbf{x}^N - \bar{\mathbf{x}}^N = 0 \quad (\text{Terminal conditions})$

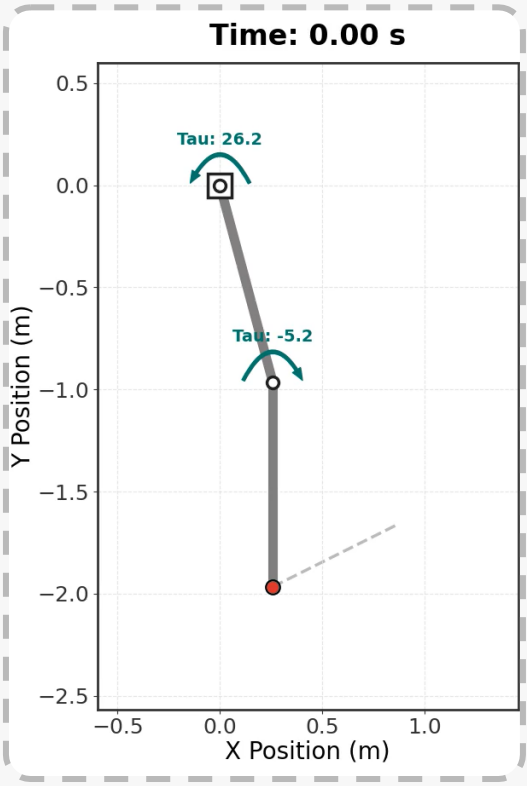
Extra = penalization of control input variations from one step to another (→ smooth control input curve)

Results:

Feasible

$n_u = 2 = n_z$
(square)

Fully-actuated

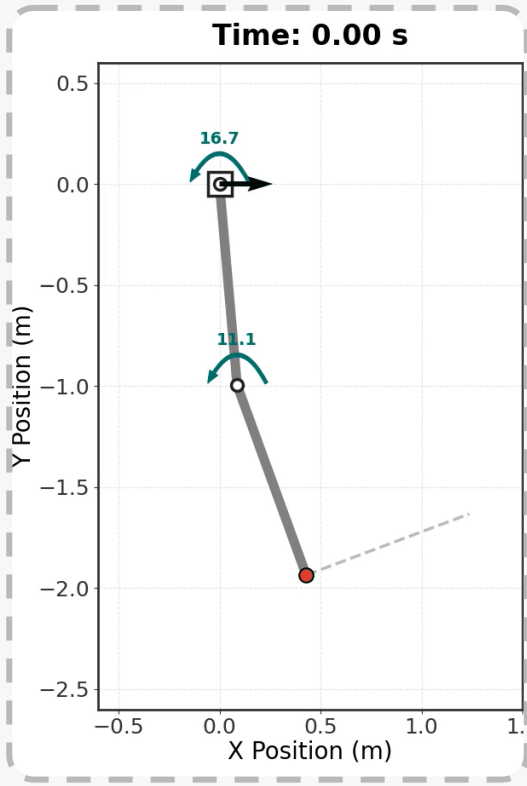


ID 0.08s
OC 38.7s

Feasible

$n_u = 3 > n_z = 2$
(underconstrained)

Fully-actuated

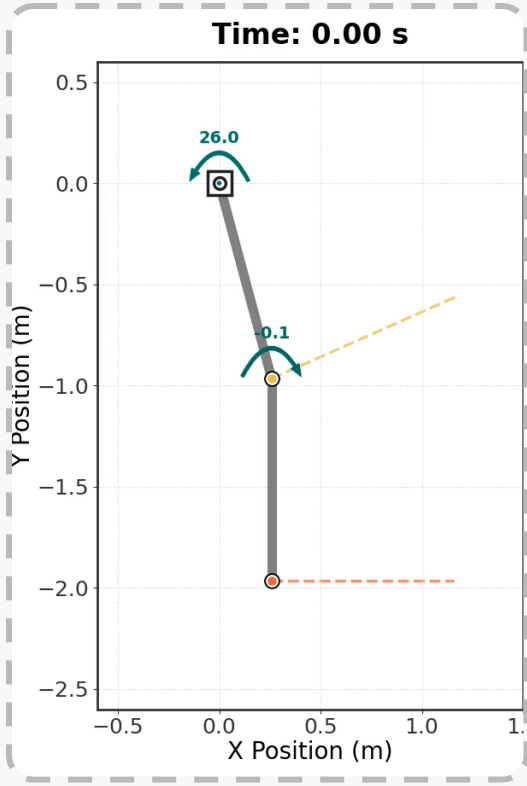


0.1s (but tricky to converge)
80.1s

Unfeasible (incompatibility)

$n_u = 3 < n_z = 4$
(overconstrained)

Fully-actuated

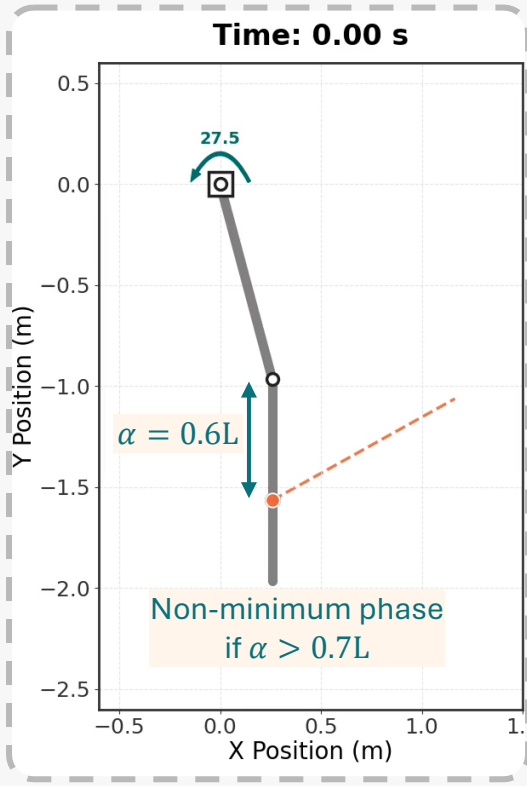


0.12s
69s

Unfeasible (control authority)

$n_u = 1 < n_z = 2$
(overconstrained, non-
collocated)

Under-actuated



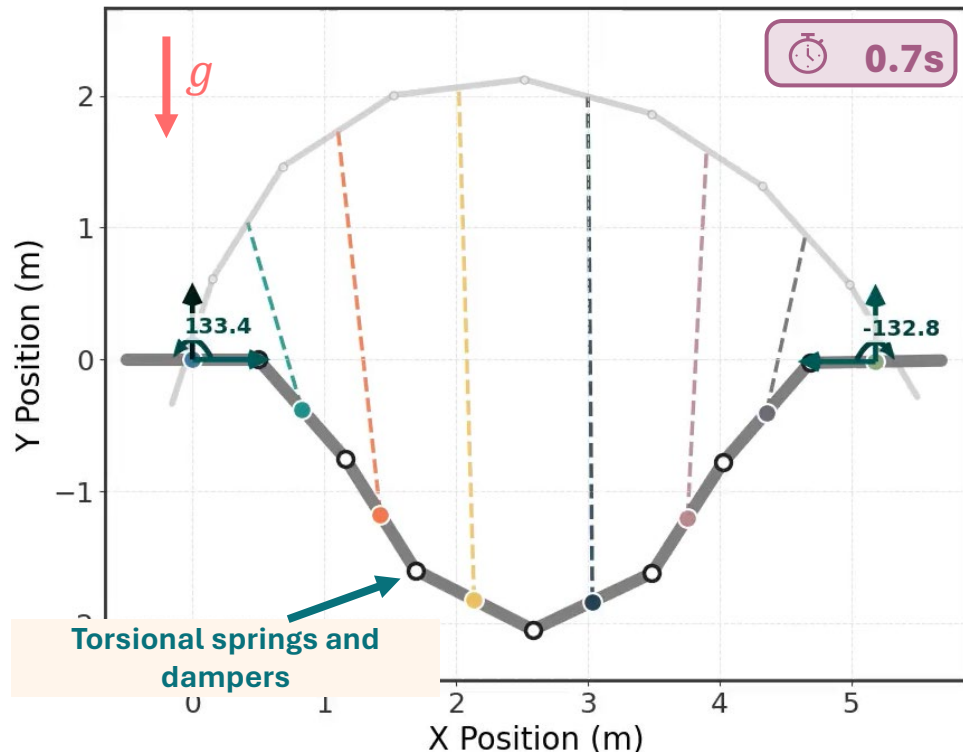
0.08s
32.5s

T = 7s. For ID: h = 0.02 (350 steps). For OC: 115 shooting intervals, with 3 time steps in each (= 345 steps)

Results: cable manipulation

These properties allow to use servo-constraints for cables manipulation with “rough” desired trajectories

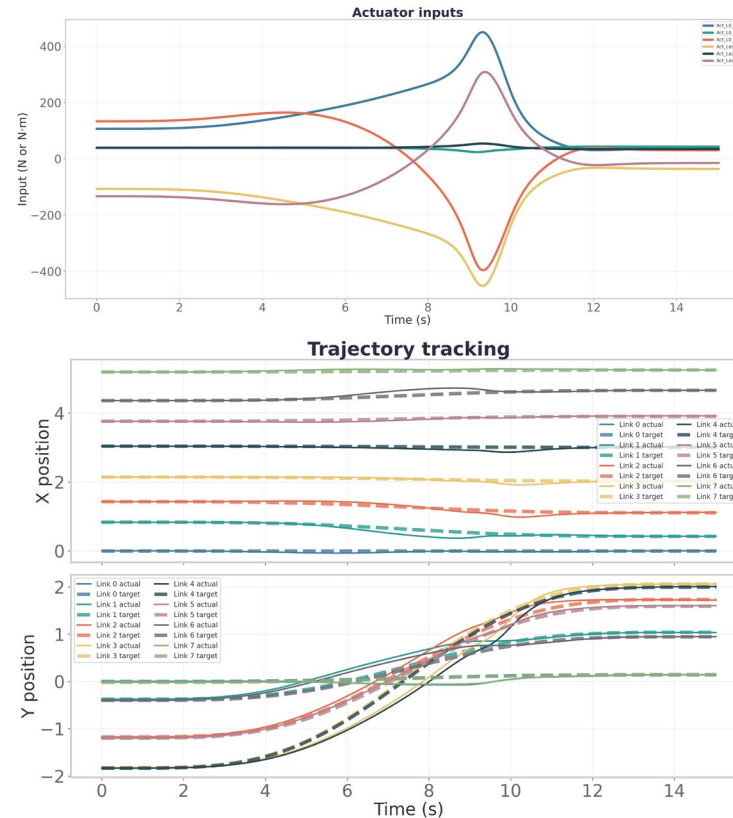
Time: 0.00 s



$T = 15s$, 750 time steps

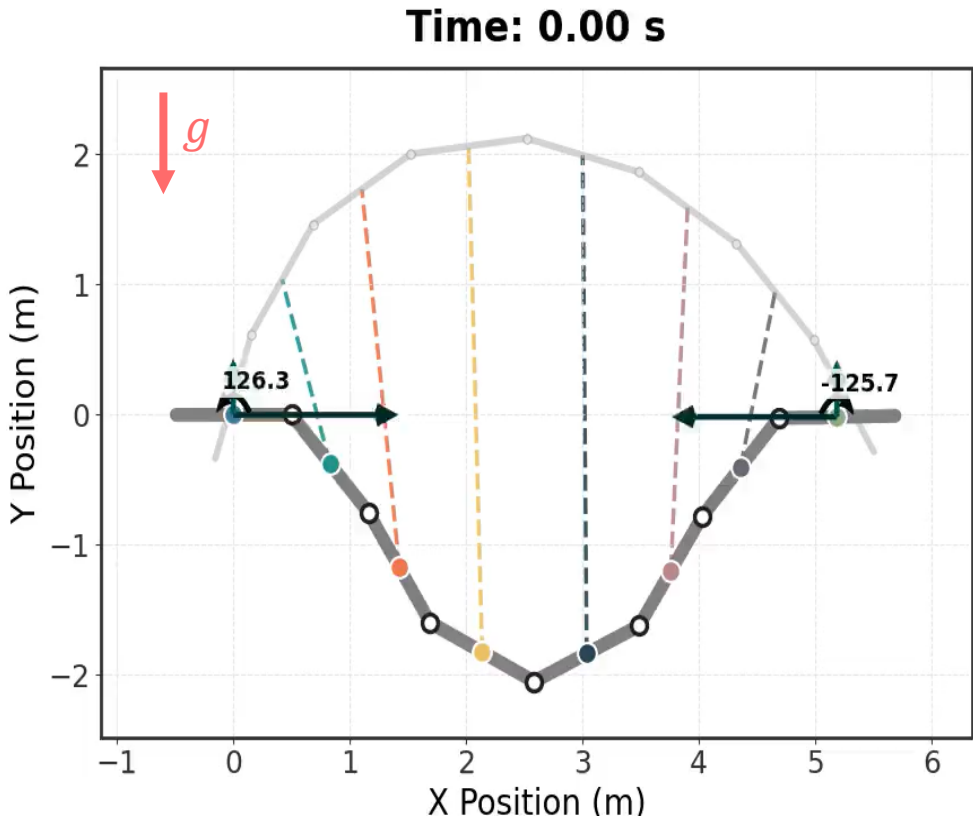
OC: did not converge

$w_u = 0$ (control inputs not penalized)

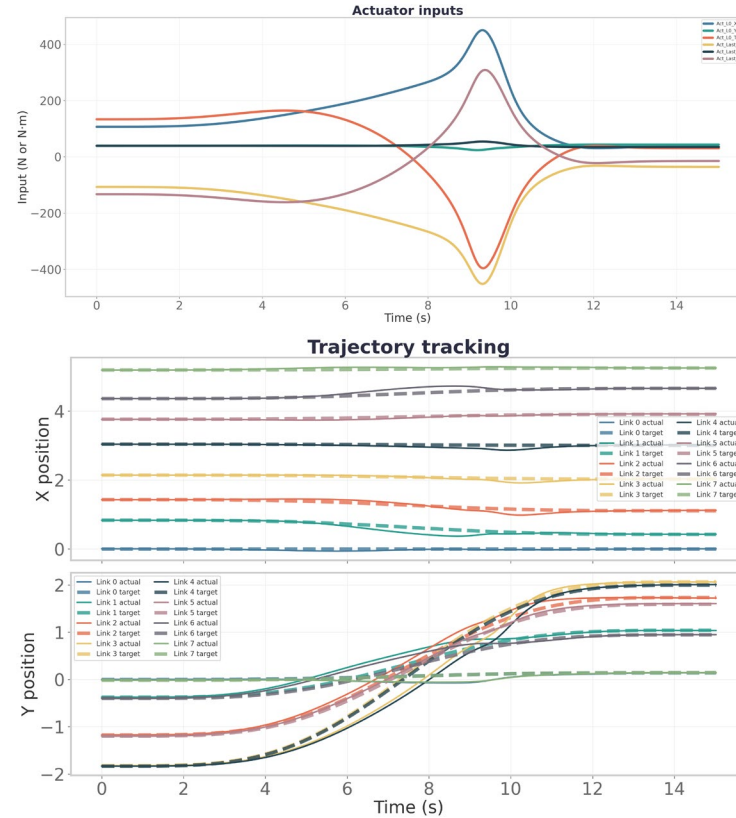


Results: cable manipulation

These properties allow to use servo-constraints for cables manipulation **with “rough” desired trajectories**

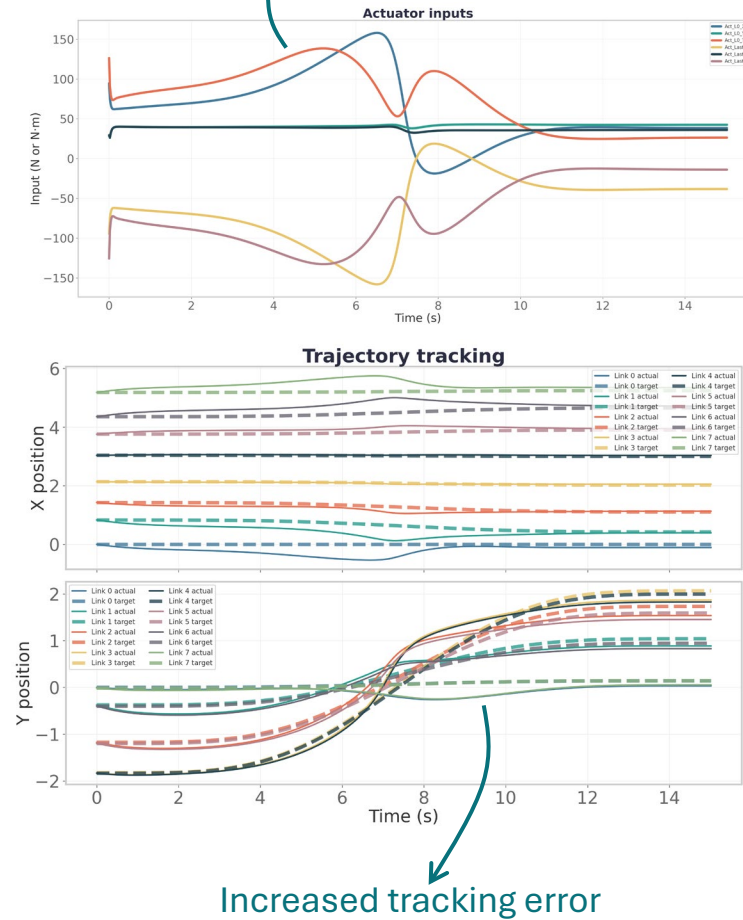


$w_u = 0$ (control inputs not penalized)



Reduced control input

$w_u = 1e-1$

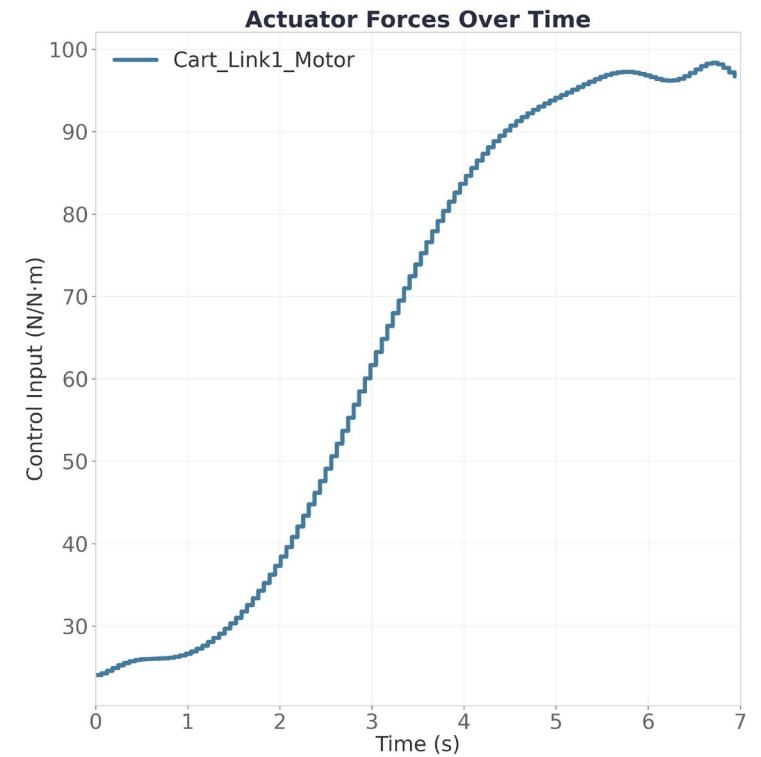
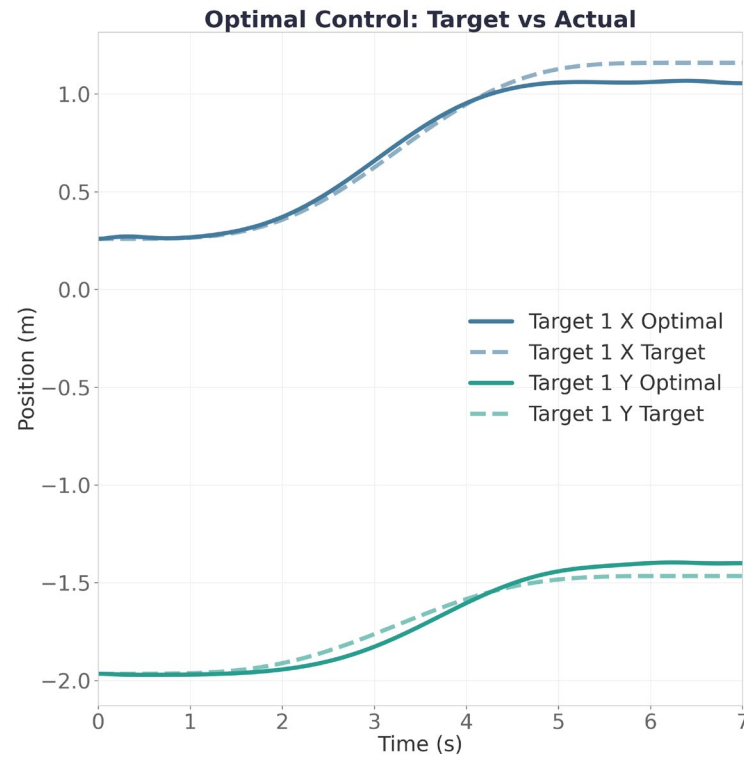
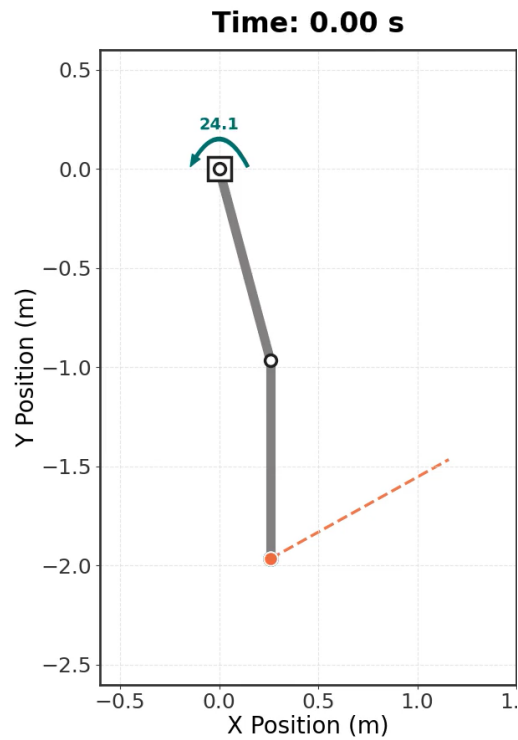


Increased tracking error

Limitations

ID via servo-constraints **can not deal with non-minimum phase systems** without stable inversion techniques

Under-actuated control of the DP tip is NMP (unstable internal dynamics) : no (bounded) solution with ID, while **OC** returns a solution



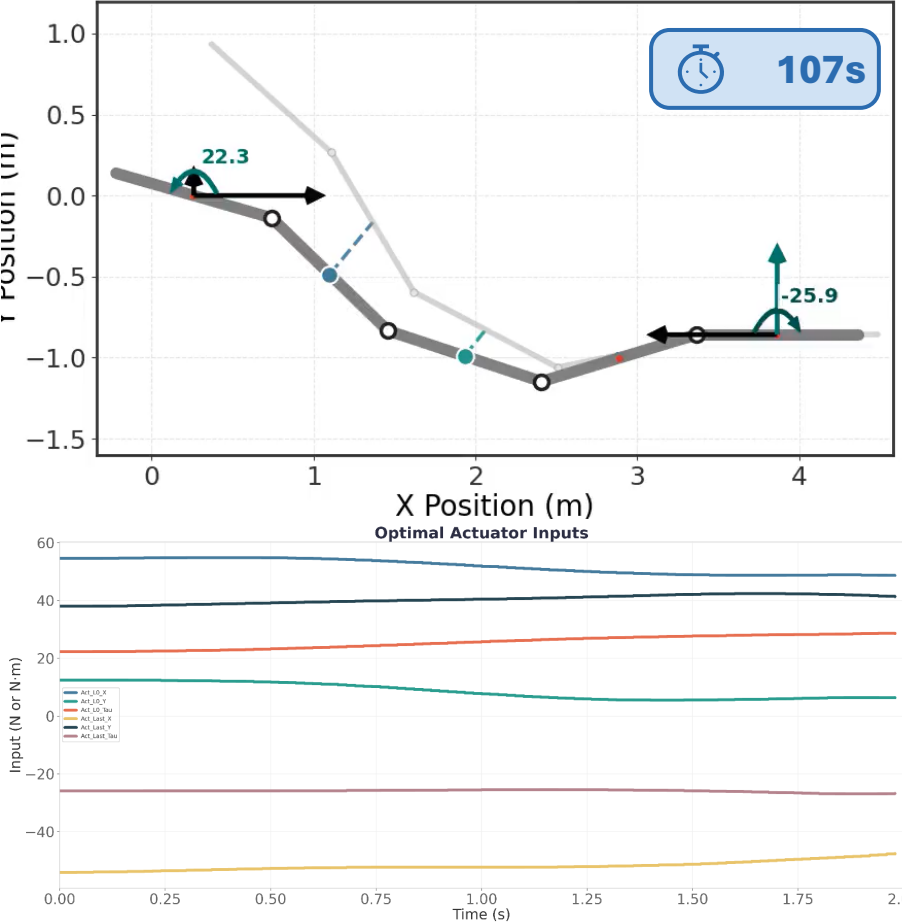
Limitations

ID via servo-constraints **can not deal with non-minimum phase systems** without stable inversion techniques

Ex: cable with no collocated prescribed trajectory

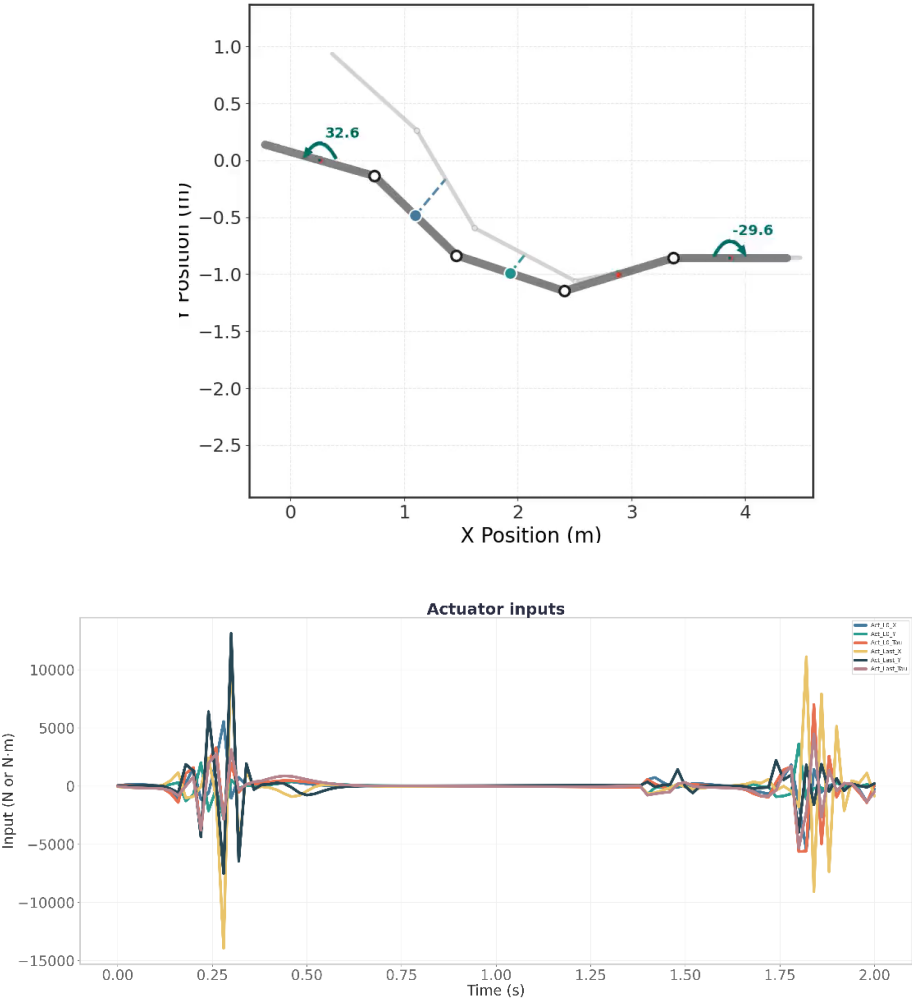
OC

Time: 0.00 s



ID

Time: 0.00 s



Conclusion

1

Servo constraints for inverse dynamics requires **square systems** and **feasible trajectories**

2

But usual robotics planning algorithms **do not provide** these guarantees



Therefore: **modification of the servo-constraints**: only the error that **can be reduced** remains



OUTCOME

- Application to **very under-actuated** systems (cables) becomes possible
- **“Best-fit” solution** when desired **trajectories are not feasible**



ADVANTAGES

- Servo constraints can **reduce computation time** a lot compared to OC
- Provides **exact solution** that best tracks the trajectory without tuning a cost function



LIMITATIONS

- OC remains necessary if a **predictive** approach is necessary or **if NMP (or stable inversion)**



PERSPECTIVES

- Consider **flexible elements and 3D** applications (SE3)
- Study the **NMP properties** of the system depending on the output choice
- Integrate planning + feedforward + feedback

References

- [1] Dehaybe, L., & Bröls, O. (2026). SE (3)-based representation of object deformations for robotic shape servoing: Formulation and assessment. *Robotics and Autonomous Systems*, 105403.
- [2] Blajer, W., & Kołodziejczyk, K. (2004). A geometric approach to solving problems of control constraints: theory and a DAE framework. *Multibody System Dynamics*, 11(4), 343-364.
- [3] Drücker, S., & Seifried, R. (2023). Trajectory-tracking control from a multibody system dynamics perspective. *Multibody system dynamics*, 58(3), 341-363.

