universität freiburg

Numerical Optimal Control for Nonsmooth Dynamic Systems

Moritz Diehl[1]

joint work with Armin Nurkanovic[1], Anton Pozharskiy[1], Christian Dietz[1,2], Sebastian Albrecht[2]

[1] Department of Microsystems Engineering and Department of Mathematics, University of Freiburg, Germany

[2] Siemens Foundational Technology, Munich, Germany

Workshop on Optimization for Learning and Control,
IMT School for Advanced Studies Lucca, June 4-6 2025

Continuous-Time Optimal Control Problems (OCP)

Continuous-Time OCP with Ordinary Differential Equation (ODE) Constraints

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)), \; t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Continuous-Time Optimal Control Problems (OCP)

Continuous-Time OCP with Ordinary Differential Equation (ODE) Constraints

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Can in most applications assume convexity of all "outer" problem functions: L_c, E, h, r.

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Three levels of difficulty:

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$

(b) Nonlinear smooth ODE: $f \in \mathcal{C}^1$

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\text{s.t.} \quad x(0) = \bar{x}_0$$

$$\dot{x}(t) = f(x(t),u(t))$$

$$0 \geq h(x(t),u(t)),\ t \in [0,T]$$

$$0 \geq r(x(T))$$

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$

(b) Nonlinear smooth ODE: $f \in \mathcal{C}^1$

(c) **Nonsmooth Dynamics (NSD):**

- f not differentiable (NSD1),
- f not continuous (NSD2), or even
- f not finite valued, discontinuous state $x(t)$ (NSD3)

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)), \ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$

(b) Nonlinear smooth ODE: $f \in \mathcal{C}^1$

(c) **Nonsmooth Dynamics (NSD):**

- f not differentiable (NSD1),
- f not continuous (NSD2), or even
- f not finite valued, discontinuous state $x(t)$ (NSD3)

First focus on smooth cases (a) and (b).

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$

(b) Nonlinear smooth ODE: $f \in \mathcal{C}^1$

First focus on smooth cases (a) and (b).

Recall: Runge-Kutta Discretization for Smooth Systems

Ordinary Differential Equation (ODE)

$$\dot{x}(t) = \underbrace{f(x(t), u(t))}_{=:v(t)}$$

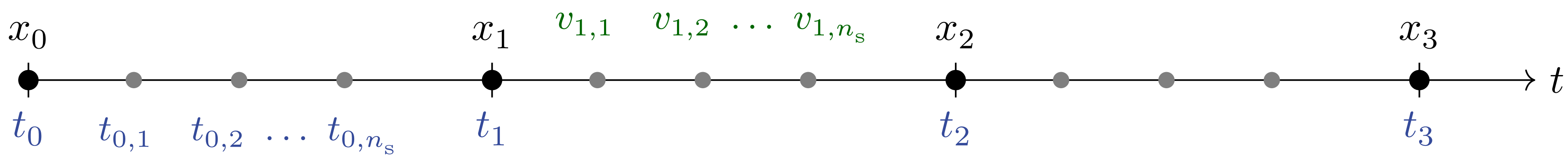
Initial Value Problem (IVP)

$$\begin{aligned} x(0) &= \bar{x}_0 \\ v(t) &= f(x(t), u(t)) \\ \dot{x}(t) &= v(t) \\ & t \in [0, T] \end{aligned}$$

Discretization: N Runge-Kutta steps of each n_s stages

$$\begin{aligned} x_{0,0} &= \bar{x}_0, \qquad\qquad \Delta t = \tfrac{T}{N} \\ v_{k,j} &= f(x_{k,j}, u_k) \\ x_{k,j} &= x_{k,0} + \Delta t \textstyle\sum_{n=1}^{n_s} a_{jn} v_{k,n} \\ x_{k+1,0} &= x_{k,0} + \Delta t \textstyle\sum_{n=1}^{n_s} b_n v_{k,n} \\ & j = 1, \ldots, n_s, \quad k = 0, \ldots, N-1 \end{aligned}$$

For fixed controls and initial value: square system with $n_x + N(2n_s + 1)n_x$ unknowns, implicitly defined via $n_x + N(2n_s + 1)n_x$ equations.
(trivial eliminations in case of explicit RK methods)



Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \quad \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\begin{aligned} \text{s.t.} \quad x(0) &= \bar{x}_0 \\ \dot{x}(t) &= f(x(t),u(t)) \\ 0 &\geq h(x(t),u(t)),\ t \in [0,T] \\ 0 &\geq r(x(T)) \end{aligned}$$

- Direct methods "first discretize, then optimize"

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\text{s.t.} \quad x(0) = \bar{x}_0$$

$$\dot{x}(t) = f(x(t),u(t))$$

$$0 \geq h(x(t),u(t)), \; t \in [0,T]$$

$$0 \geq r(x(T))$$

- Direct methods "first discretize, then optimize"

1. Parameterize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \textstyle\int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}$$

- Direct methods "first discretize, then optimize"

1. Parameterize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics

$$L_\mathrm{d}(x_n, z_k, u_n) \approx \int_{t_n}^{t_{n+1}} L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t$$

Replace $\dot{x} = f(x,u)$ by

$$\begin{aligned}
x_{n+1} &= \phi_f(x_n, z_n, u_n) \\
0 &= \phi_\mathrm{int}(x_n, z_n, u_n)
\end{aligned}$$

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\min_{x(\cdot),u(\cdot)} \int_0^T L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t + E(x(T))$$

$$\begin{aligned} \text{s.t.} \quad & x(0) = \bar{x}_0 \\ & \dot{x}(t) = f(x(t),u(t)) \\ & 0 \geq h(x(t),u(t)),\ t \in [0,T] \\ & 0 \geq r(x(T)) \end{aligned}$$

- Direct methods "first discretize, then optimize"

1. Parameterize controls, e.g. $u(t) = u_n, t \in [t_n, t_{n+1}]$.
2. Discretize cost and dynamics

$$L_\mathrm{d}(x_n, z_k, u_n) \approx \int_{t_n}^{t_{n+1}} L_\mathrm{c}(x(t),u(t))\,\mathrm{d}t$$

Replace $\dot{x} = f(x,u)$ by

$$x_{n+1} = \phi_f(x_n, z_n, u_n)$$

$$0 = \phi_\mathrm{int}(x_n, z_n, u_n)$$

3. Also discretize path constraints

$$0 \geq \phi_h(x_n, z_n, u_n),\ n = 0, \dots N-1.$$

Direct Methods Transform OCP into Nonlinear Program (NLP)

Continuous time OCP

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}$$

- Direct methods "first discretize, then optimize"

Discrete time OCP (an NLP)

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \quad & \textstyle\sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k,z_k,u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi_f(x_n,z_n,u_n) \\
& 0 = \phi_{\mathrm{int}}(x_n,z_n,u_n) \\
& 0 \geq \phi_h(x_n,z_n,u_n),\ n = 0,\ldots,N-1 \\
& 0 \geq r(x_N)
\end{aligned}$$

Variables $\mathbf{x} = (x_0,\ldots,x_N)$, $\mathbf{z} = (z_0,\ldots,z_N)$ and $\mathbf{u} = (u_0,\ldots,u_{N-1})$.
Here, $\mathbf{z}$ are the intermediate variables of the integrator (e.g. Runge-Kutta)

Simplest Direct Transcription: Single Step Explicit Euler

(not recommended in practice, other Runge-Kutta methods are much more efficient)

Continuous time OCP

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_c(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}$$

- Direct methods: first discretize, then optimize

Single Step Explicit Euler NLP, with $\Delta t = \frac{T}{N}$

$$\begin{aligned}
\min_{\mathbf{x},\mathbf{u}} \quad & \sum_{k=0}^{N-1} L_c(x_k,u_k)\Delta t + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = x_n + f(x_n,u_n)\Delta t \\
& 0 \geq h(x_n,u_n),\ n = 0,\ldots,N-1 \\
& 0 \geq r(x_N)
\end{aligned}$$

Variables $\mathbf{x} = (x_0,\ldots,x_N)$ and $\mathbf{u} = (u_0,\ldots,u_{N-1})$.
(single step explicit Euler has no internal integrator variables $\mathbf{z}$)

Sparse NLP resulting from direct transcription

Discrete time OCP (an NLP)

$$
\begin{aligned}
\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \ & \textstyle\sum_{k=0}^{N-1} L_{\mathrm{d}}(x_k, z_n, u_k) + E(x_N) \\
\text{s.t.} \quad & x_0 = \bar{x}_0 \\
& x_{n+1} = \phi_f(x_n, z_n, u_n) \\
& 0 = \phi_{\mathrm{int}}(x_n, z_n, u_n) \\
& 0 \geq \phi_h(x_n, z_n, u_n), \ n = 0, \ldots, N-1 \\
& 0 \geq r(x_N)
\end{aligned}
$$

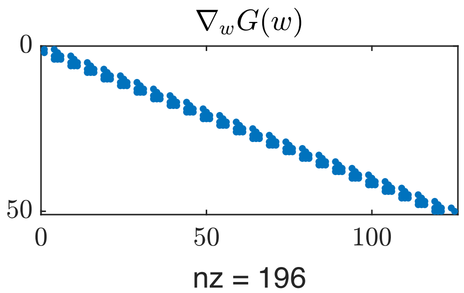
Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

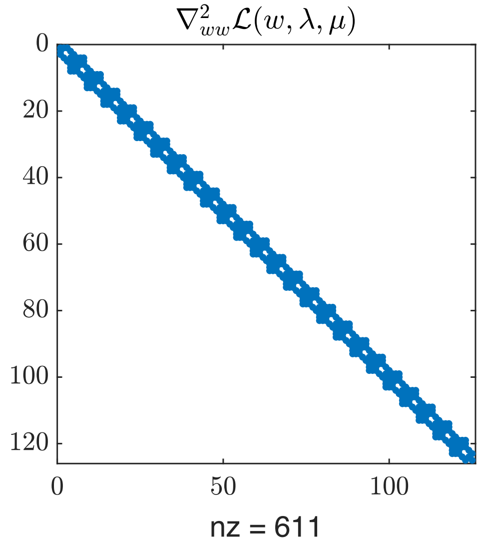
Nonlinear Program (NLP)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^{n_x}} \ & F(w) \\
\text{s.t.} \ & G(w) = 0 \\
& H(w) \geq 0
\end{aligned}
$$

Large and sparse NLP

Sparse NLP resulting from direct transcription





Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

Nonlinear Program (NLP)

$$\begin{aligned} \min_{w \in \mathbb{R}^{n_x}} \quad & F(w) \\ \text{s.t.} \quad & G(w) = 0 \\ & H(w) \geq 0 \end{aligned}$$

Large and sparse NLP

Illustrative example of direct collocation with Newton-type optimization

Illustrative nonlinear optimal control problem (with one state and one control)

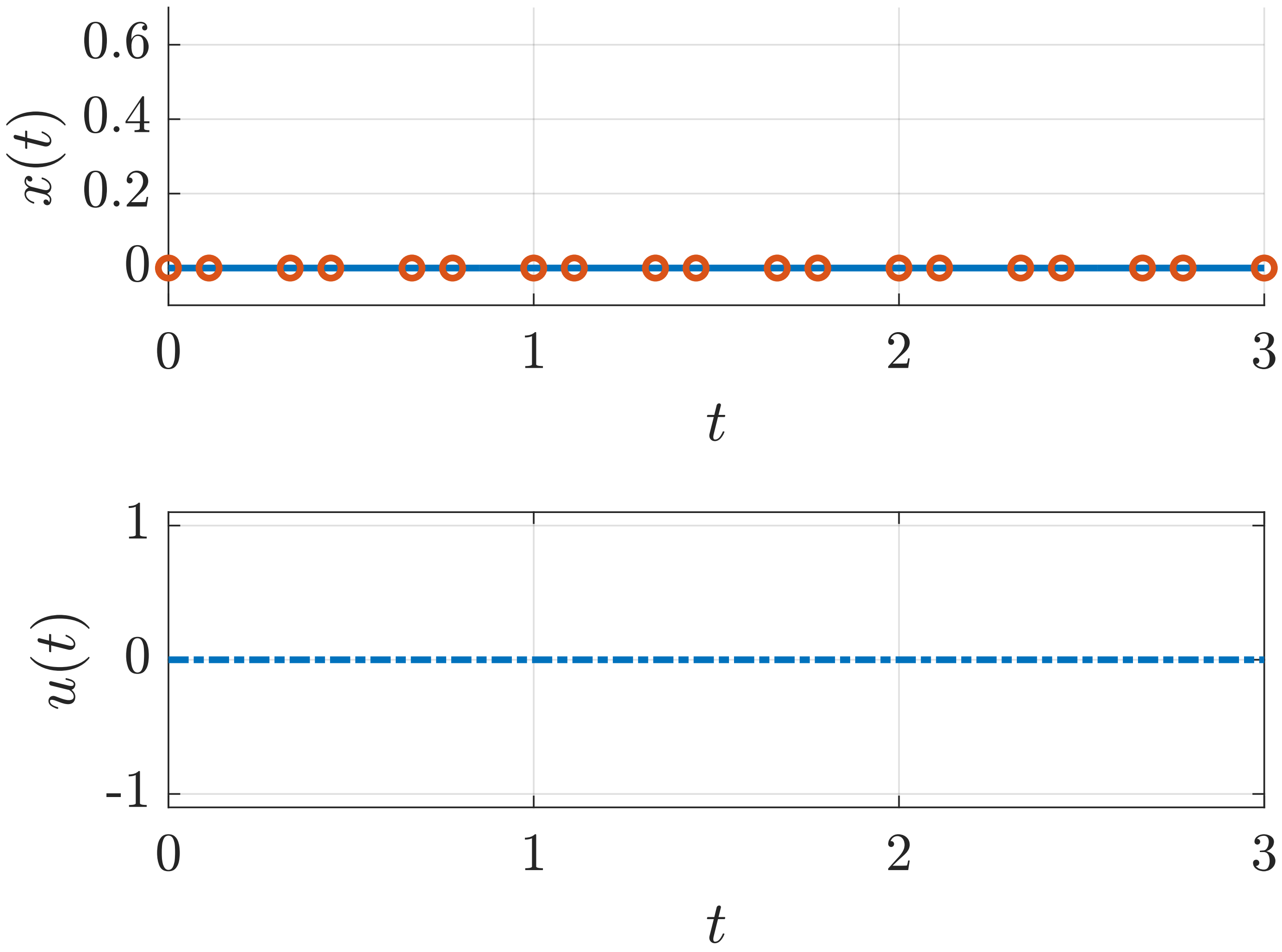
$$\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad \int_0^3 x(t)^2 + u(t)^2 \, dt$$

subject to

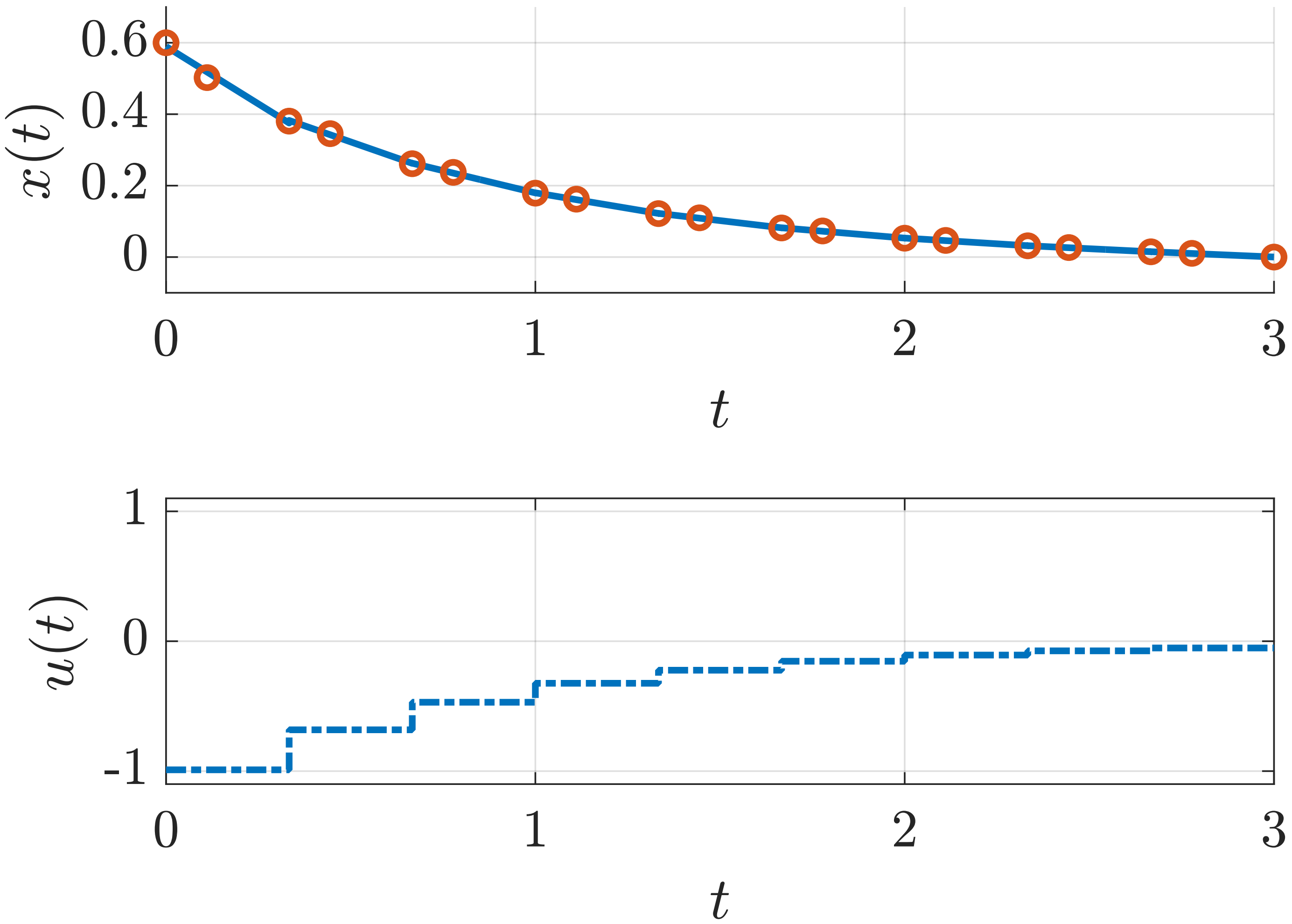
$$
\begin{aligned}
x(0) &= \bar{x}_0 && && \text{(initial value, } \bar{x}_0 = 0.6\text{)}\\
\dot{x} &= (1+x)x + u, && && \text{(ODE model)}\\
-1 \leq u(t) &\leq 1, && t \in [0,3] && \text{(bounds)}\\
x(3) &= 0 && && \text{(terminal constraint)}
\end{aligned}
$$

- choose $N = 9$ equal intervals and Radau-IIA collocation with $n_s = 2$ stages
- obtain nonlinear program with $n_x + (2n_s + 1)Nn_x + Nn_u$ variables
- initialize with zeros everywhere, solve with CasADi and Ipopt (interior point)

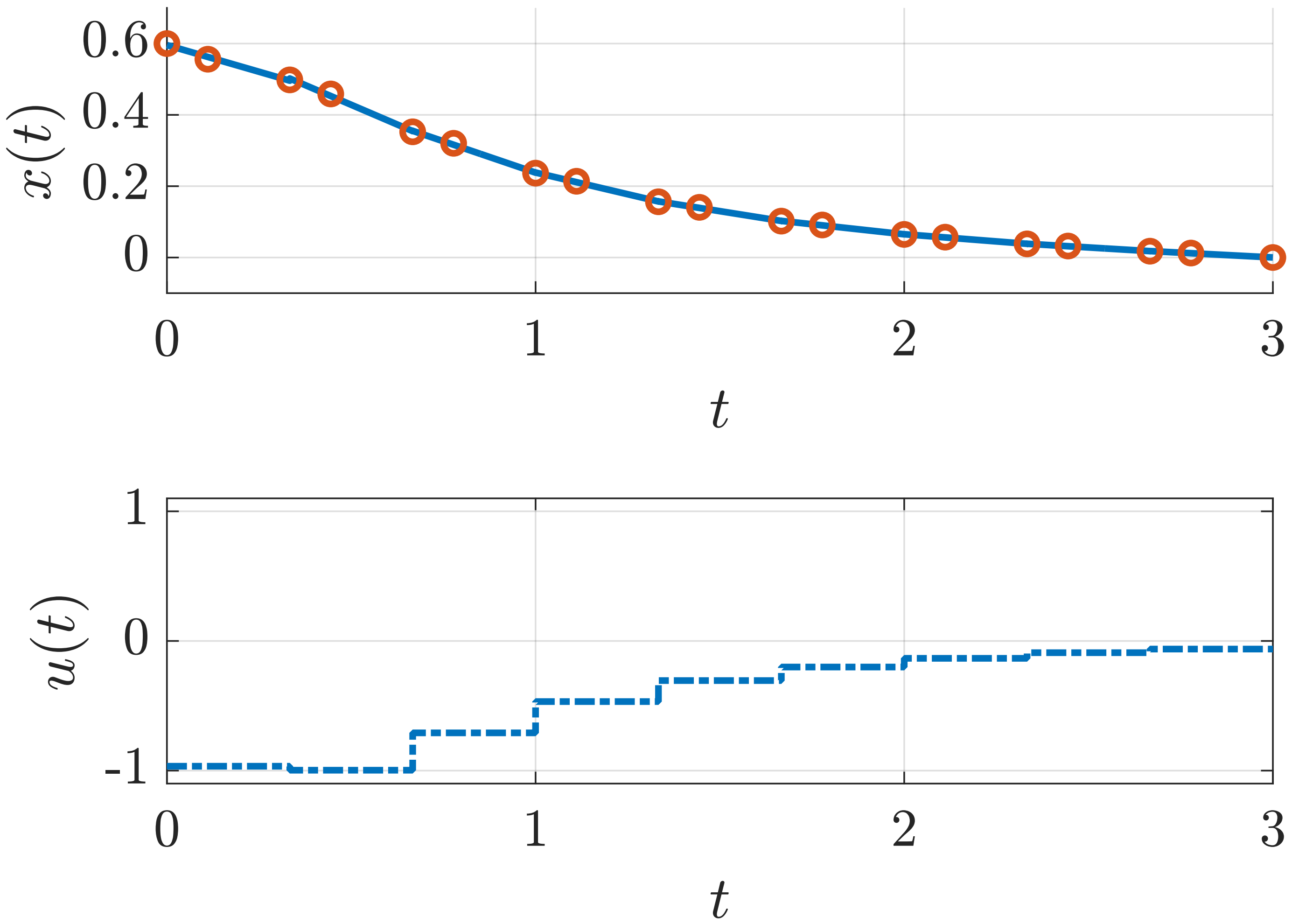
Illustrative example: Initialization



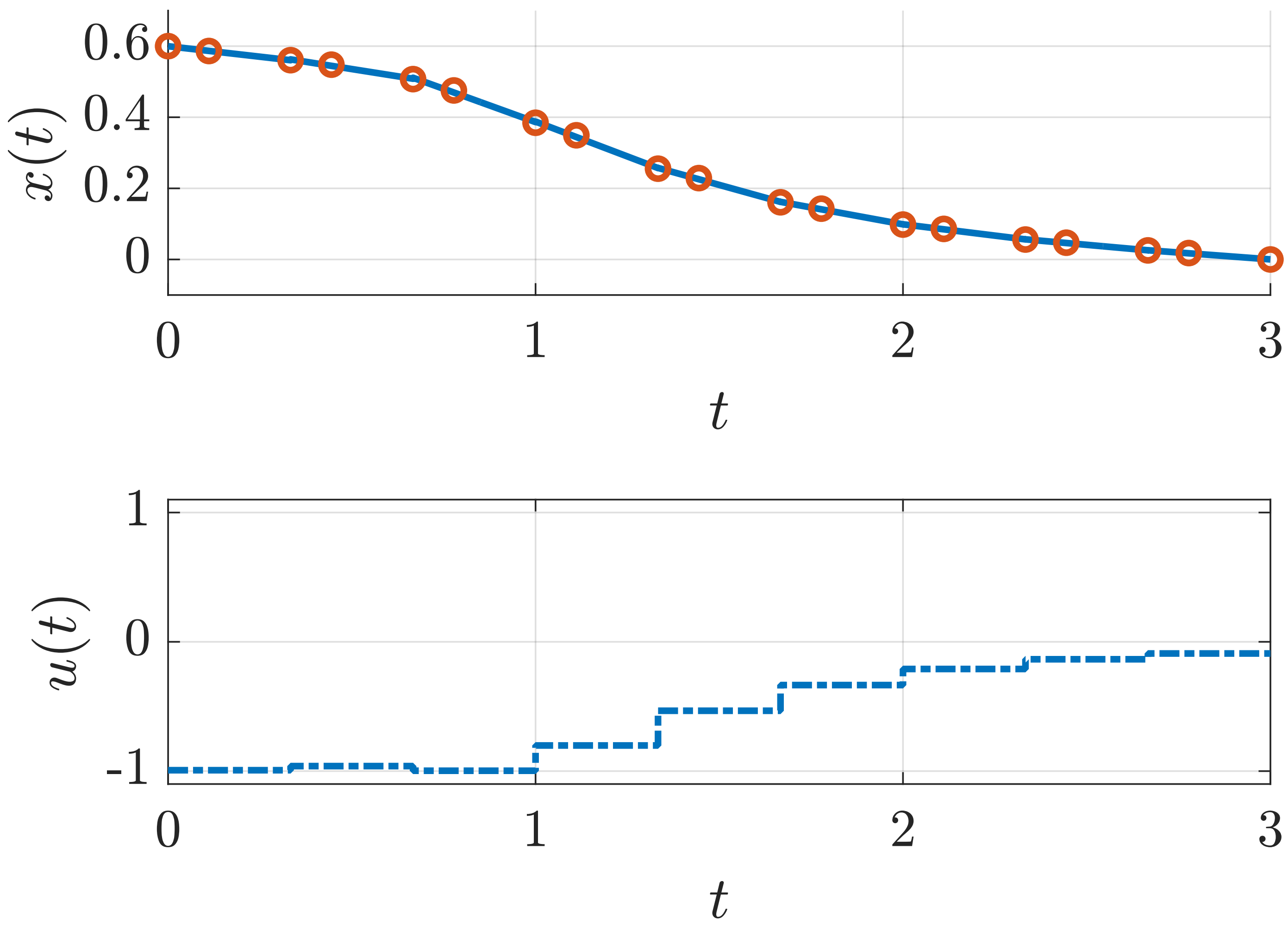
Illustrative example: First Iterate



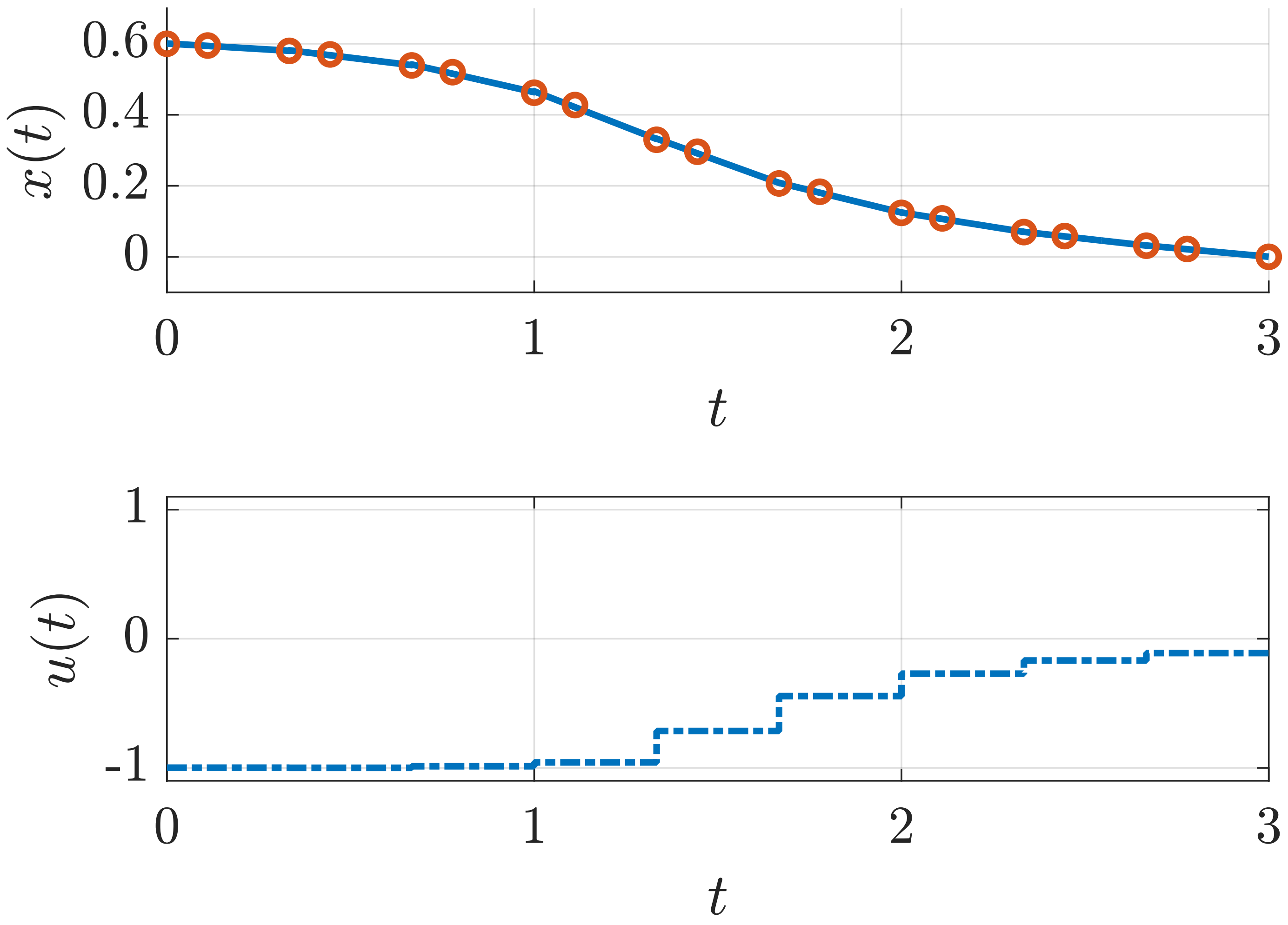
Illustrative example: Second Iterate



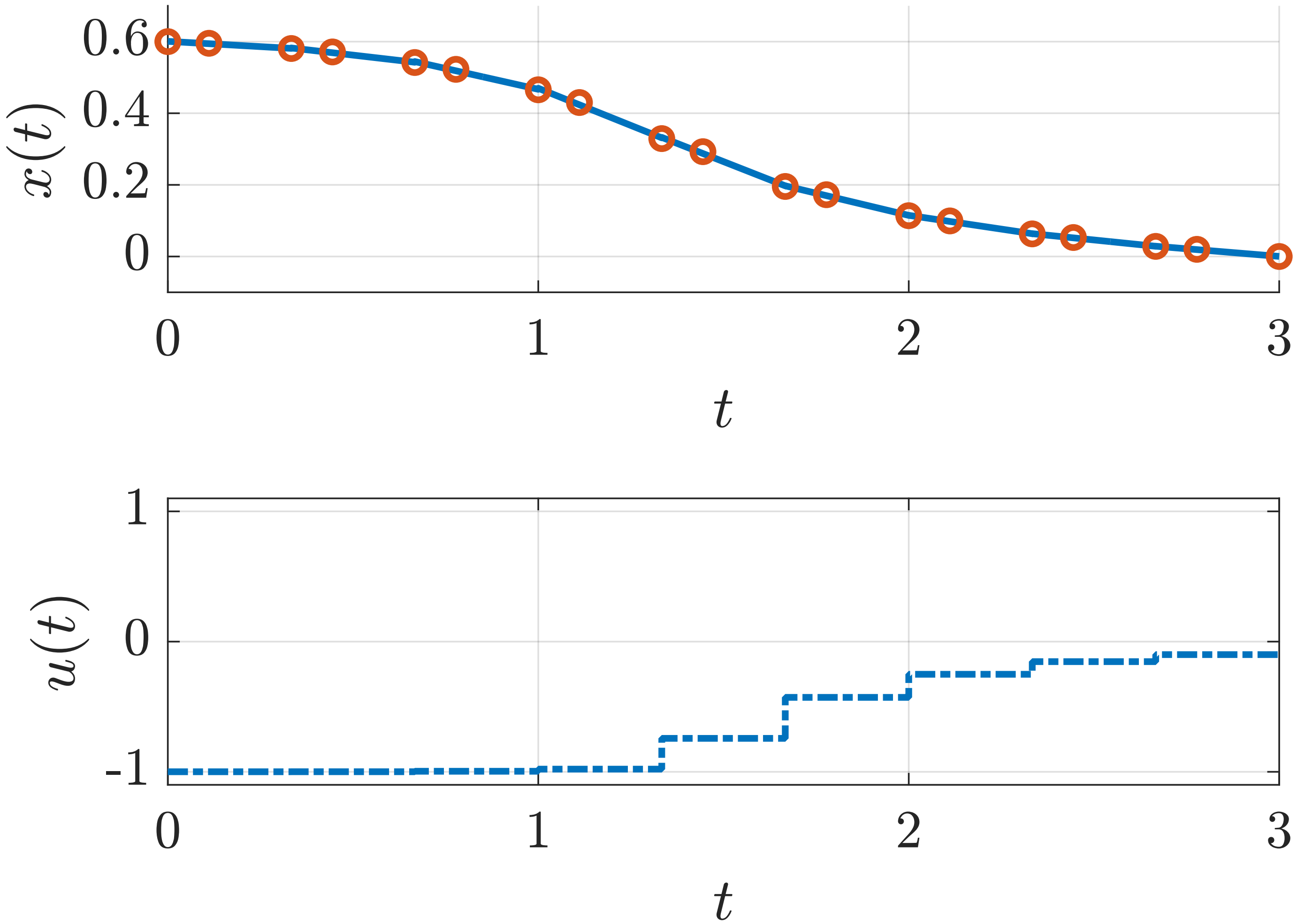
Illustrative example: Third Iterate



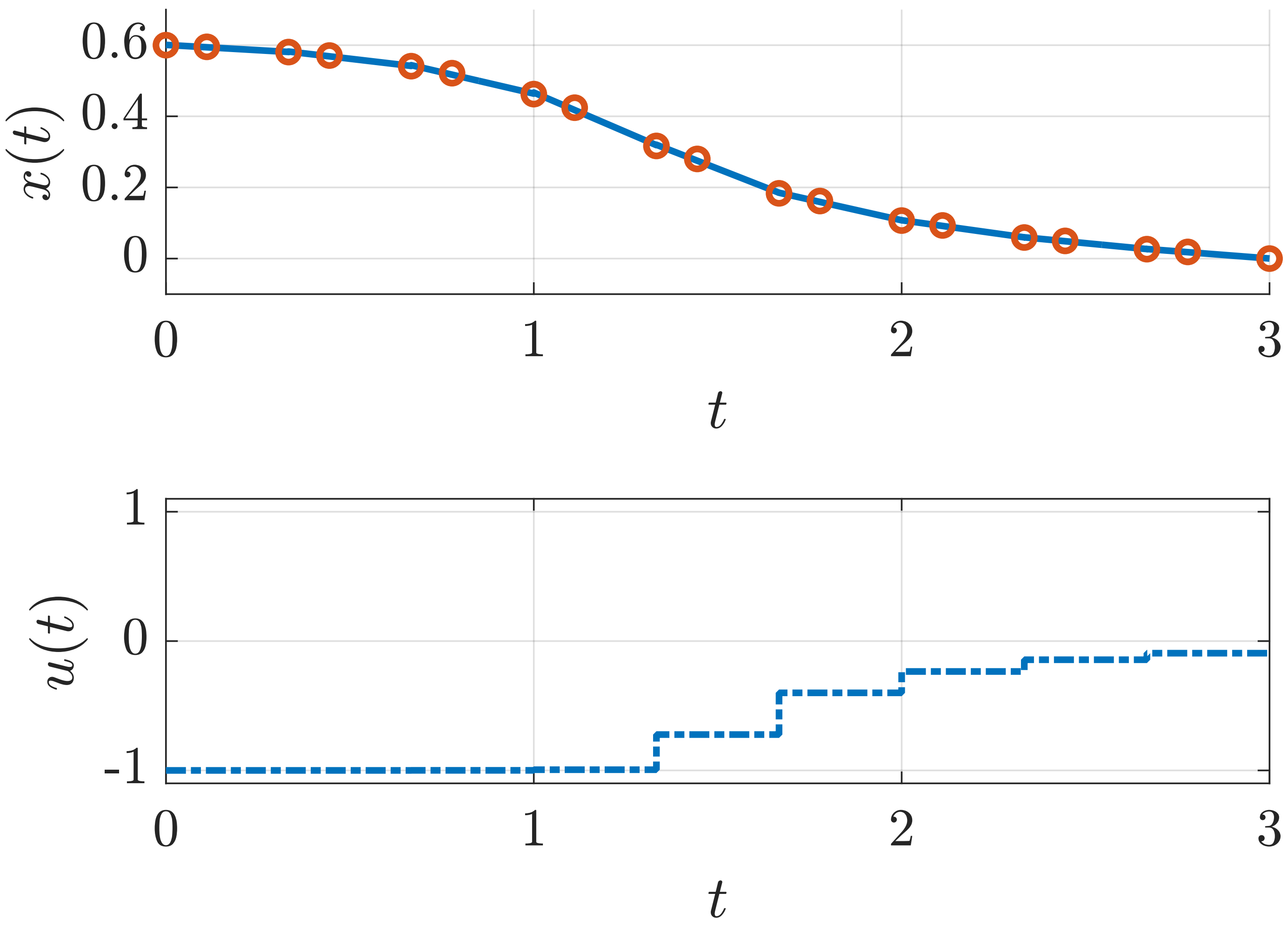
Illustrative example: Fourth Iterate



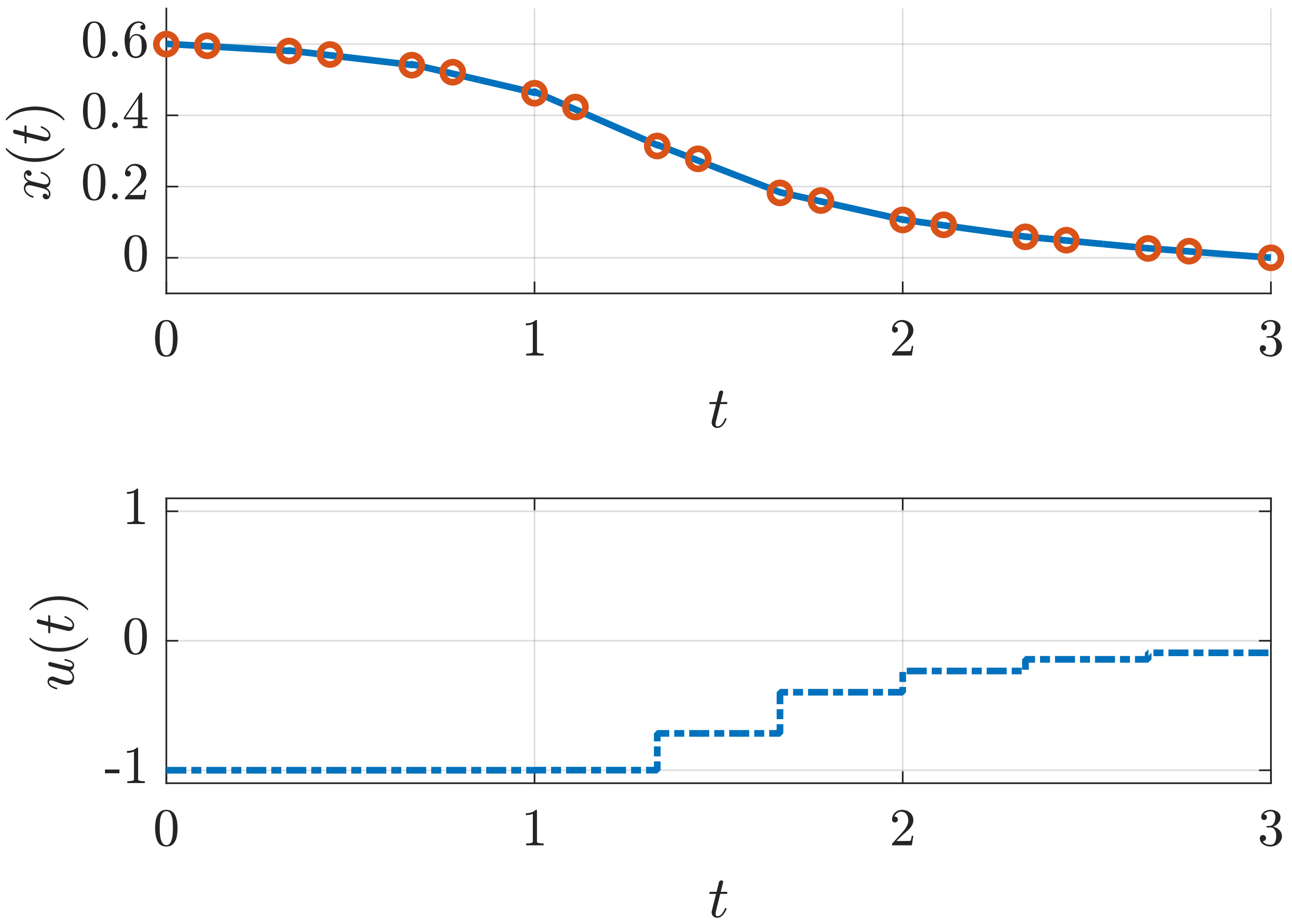
Illustrative example: Fifth Iterate



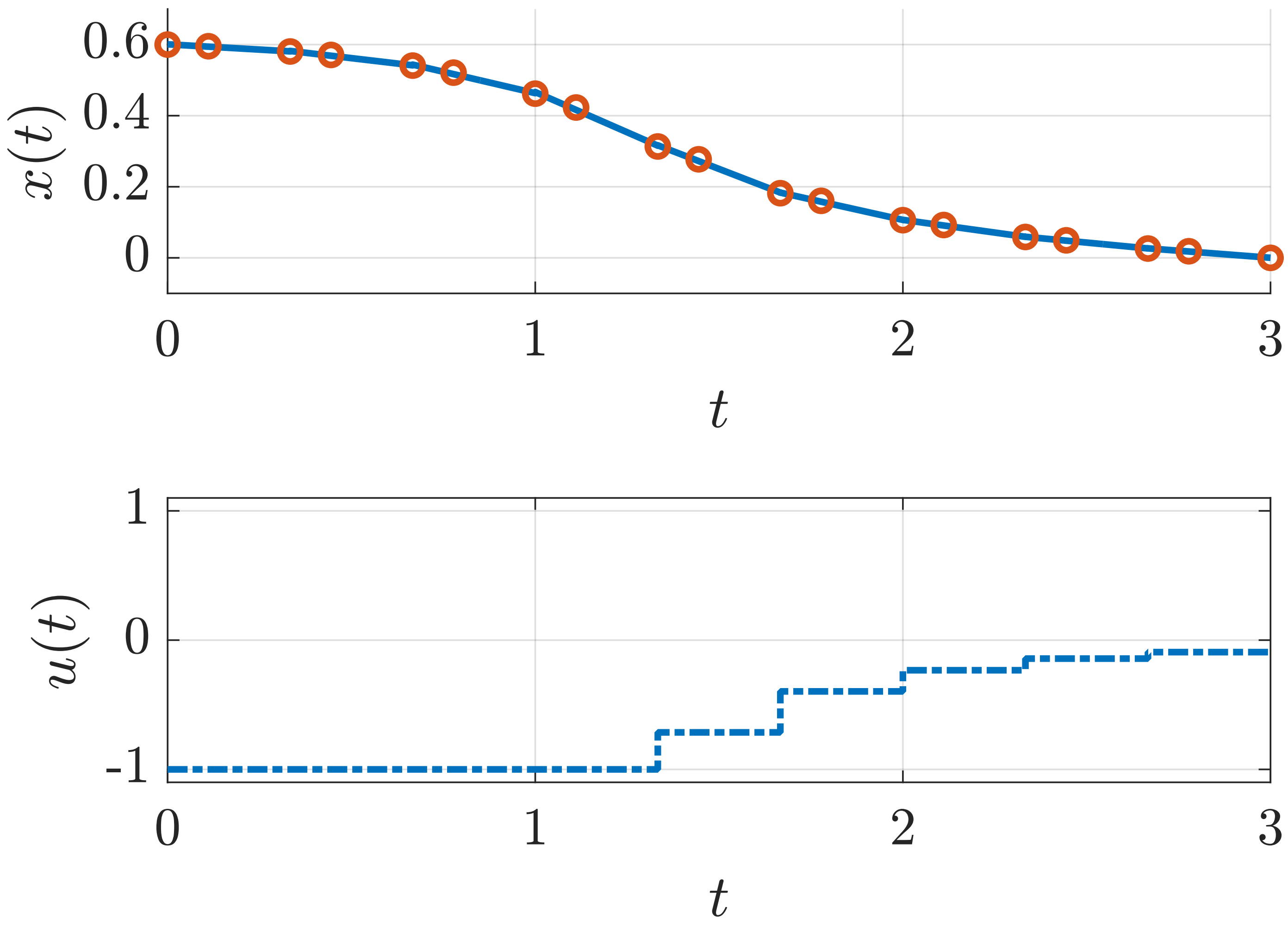
Illustrative example: Sixth Iterate



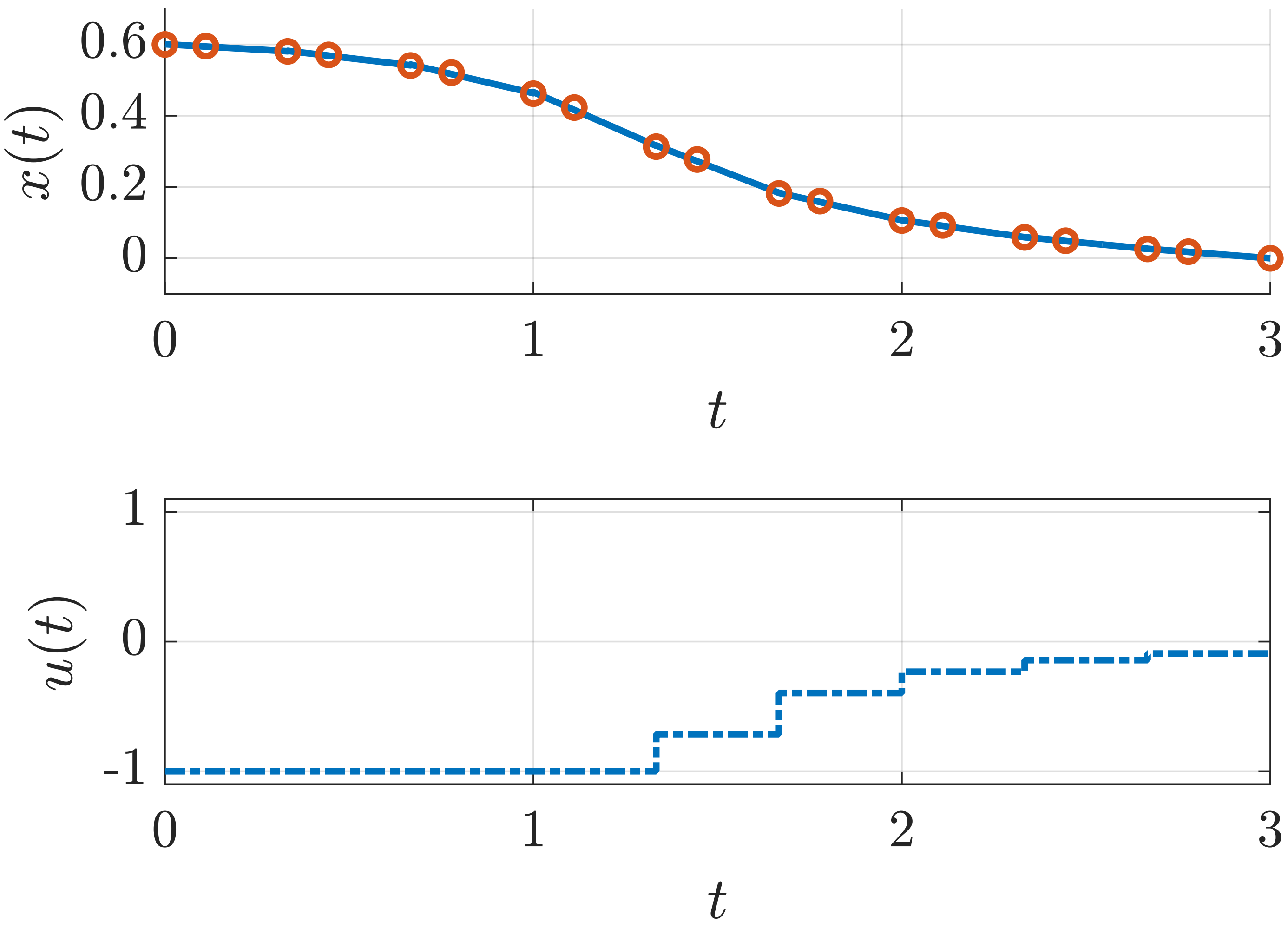
Illustrative example: Seventh Iterate



Illustrative example: Eighth Iterate



Illustrative example: Solution after Nine Newton-type Iterations



More Complex Example: Power Optimal Trajectories in Airborne Wind Energy (AWE)

formulated and solved daily by practitioners using open-source python package “AWEBox” [De Schutter et al. 2023]

For simple plane attached to a tether:

- 20 differential states (3+3 trans, 9+3 rotation, 1+1 tether)
- 1 algebraic state (tether force)
- 8 invariants (6 rotation, 2 due to tether constraint)
- 3 control inputs (aileron, elevator, tether length)

Translational:
$$\begin{bmatrix} m & 0 & 0 & x \\ 0 & m & 0 & y \\ 0 & 0 & m & z \\ x & y & z & 0 \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \\ \lambda \end{bmatrix} = \begin{bmatrix} F_x + m\left(\dot{\delta}^2 r_A + \dot{\delta}^2 x + 2\dot{\delta}\dot{y} + \ddot{\delta} y\right) \\ F_y + m\left(y\dot{\delta}^2 - 2\dot{x}\dot{\delta} - \ddot{\delta}(rA + x)\right) \\ F_z - gm \\ -\dot{x}^2 - \dot{y}^2 - \dot{z}^2 \end{bmatrix}$$

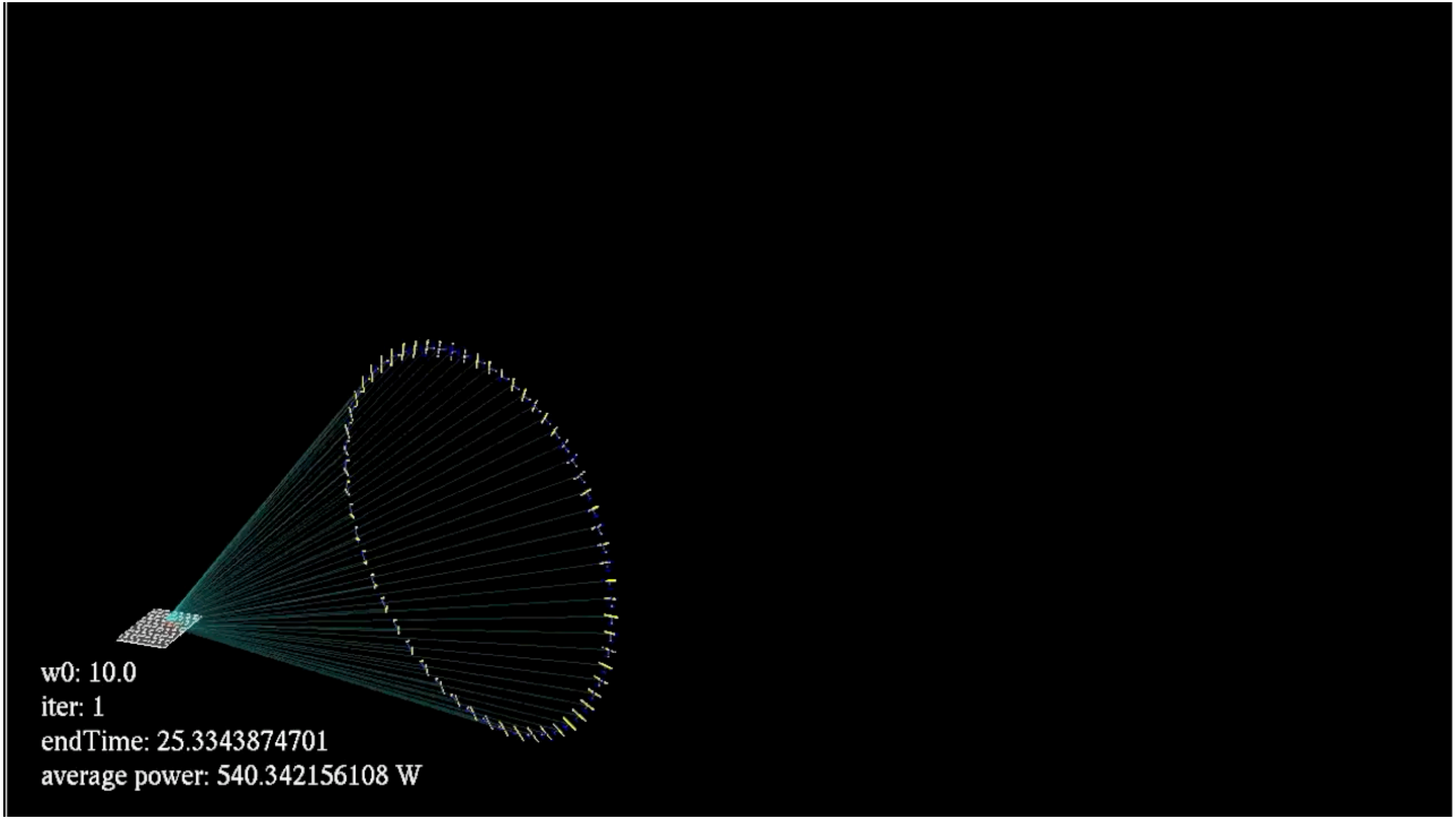
Rotational:
$$\dot{R} = R\omega_\times - R^T \begin{bmatrix} 0 \\ 0 \\ \dot{\delta} \end{bmatrix}, \qquad J\dot{\omega} = T - \omega \times J\omega, \qquad R = \begin{bmatrix} \vec{E}_x & \vec{E}_y & \vec{E}_z \end{bmatrix}$$

Aero. coefficients:
$$\vec{v} = \begin{bmatrix} \dot{x} - \dot{\delta} y \\ \dot{y} + \dot{\delta}(r_\mathrm{A} + x) \\ \dot{z} \end{bmatrix} - \vec{w}(x, y, z, \delta, t), \qquad \alpha = -\frac{\vec{E}_z^T \vec{v}}{\vec{E}_x^T \vec{v}}, \qquad \beta = \frac{\vec{E}_y^T \vec{v}}{\vec{E}_x^T \vec{v}}$$

Aero. forces/torques:
$$\vec{F}_\mathrm{A} = \frac{1}{2}\rho A \|\vec{v}\| (C_\mathrm{L} \vec{v} \times \vec{E}_y - C_\mathrm{D} \vec{v}), \quad \vec{T}_\mathrm{A} = \frac{1}{2}\rho A \|\vec{v}\|^2 \begin{bmatrix} C_\mathrm{R} \\ C_\mathrm{P} \\ C_\mathrm{Y} \end{bmatrix}$$

Newton-Type Optimization Iterations for Power Optimal Flight

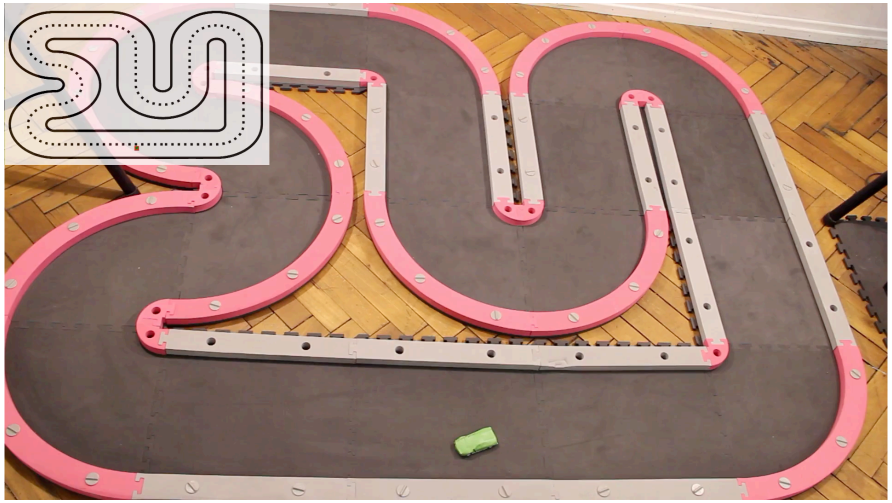
(video by Greg Horn, using CasADi and Ipopt as optimization engine)



Nonlinear Optimal Control often used for Model Predictive Control (MPC)

One widely used nonlinear MPC package is **`acados`** [Verscheuren et al. 2021]

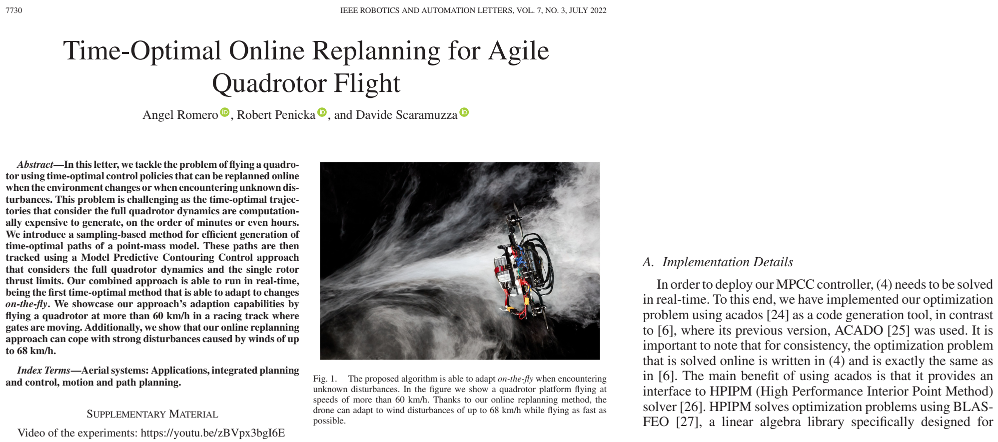
Example 1: Autonomous Driving (in Freiburg)



Example 2: Quadrotor Racing (U Zurich, Scaramuzza)

Paper: https://ieeexplore.ieee.org/abstract/document/9805699

Video: https://www.youtube.com/watch?v=zBVpx3bgI6E



7730 IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 7, NO. 3, JULY 2022

Time-Optimal Online Replanning for Agile Quadrotor Flight

Angel Romero, Robert Penicka, and Davide Scaramuzza

***Abstract*—In this letter, we tackle the problem of flying a quadrotor using time-optimal control policies that can be replanned online when the environment changes or when encountering unknown disturbances. This problem is challenging as the time-optimal trajectories that consider the full quadrotor dynamics are computationally expensive to generate, on the order of minutes or even hours. We introduce a sampling-based method for efficient generation of time-optimal paths of a point-mass model. These paths are then tracked using a Model Predictive Contouring Control approach that considers the full quadrotor dynamics and the single rotor thrust limits. Our combined approach is able to run in real-time, being the first time-optimal method that is able to adapt to changes *on-the-fly*. We showcase our approach's adaption capabilities by flying a quadrotor at more than 60 km/h in a racing track where gates are moving. Additionally, we show that our online replanning approach can cope with strong disturbances caused by winds of up to 68 km/h.**

***Index Terms*—Aerial systems: Applications, integrated planning and control, motion and path planning.**

SUPPLEMENTARY MATERIAL

Video of the experiments: https://youtu.be/zBVpx3bgI6E

Fig. 1. The proposed algorithm is able to adapt *on-the-fly* when encountering unknown disturbances. In the figure we show a quadrotor platform flying at speeds of more than 60 km/h. Thanks to our online replanning method, the drone can adapt to wind disturbances of up to 68 km/h while flying as fast as possible.

A. Implementation Details

In order to deploy our MPCC controller, (4) needs to be solved in real-time. To this end, we have implemented our optimization problem using acados [24] as a code generation tool, in contrast to [6], where its previous version, ACADO [25] was used. It is important to note that for consistency, the optimization problem that is solved online is written in (4) and is exactly the same as in [6]. The main benefit of using acados is that it provides an interface to HPIPM (High Performance Interior Point Method) solver [26]. HPIPM solves optimization problems using BLASFEO [27], a linear algebra library specifically designed for

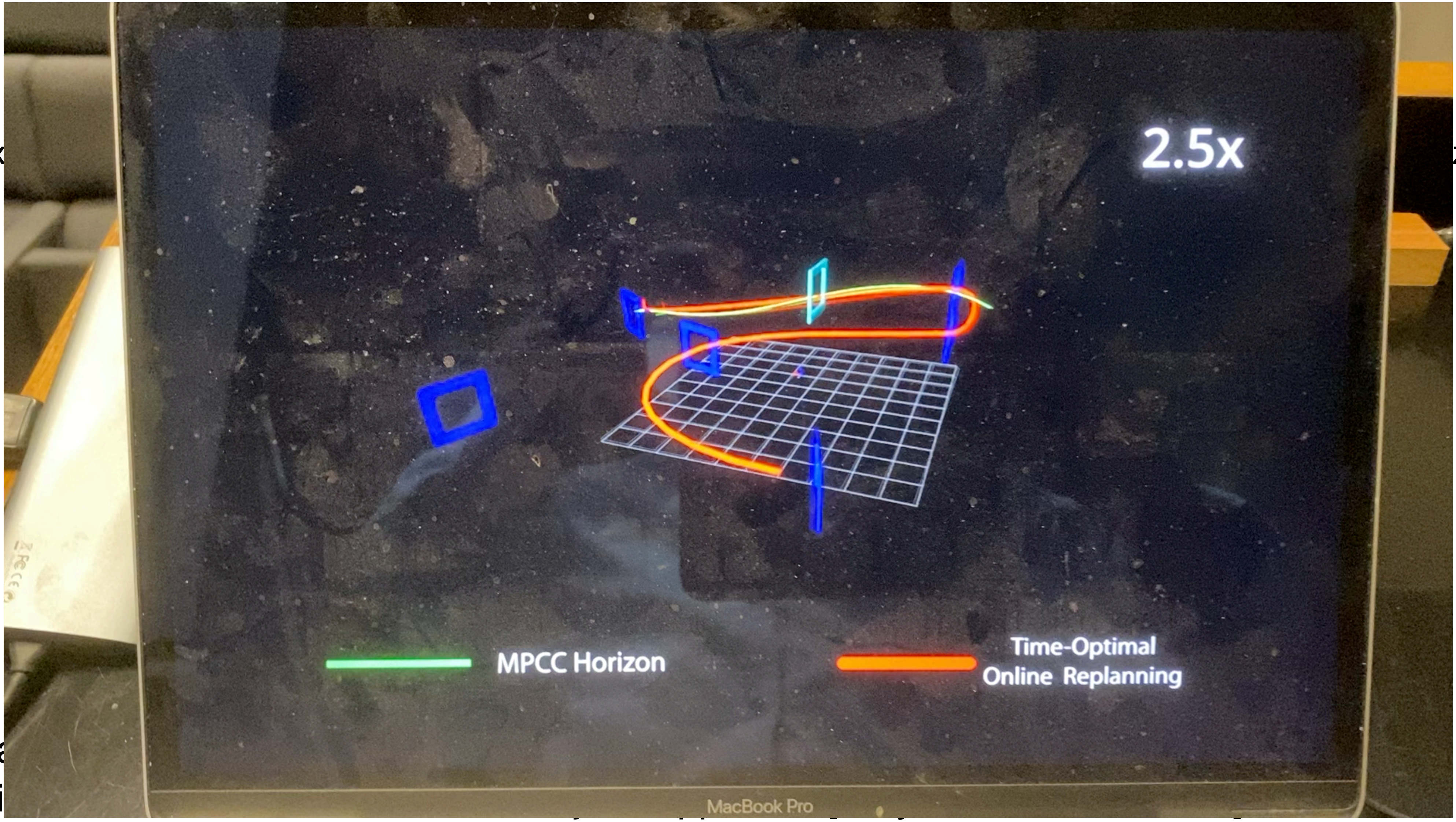
Latest **`acados`** development:

differentiable nonlinear MPC via adjoint approach [Frey et al. 2025, subm.]

Nonlinear Optimal Control often used for Model Predictive Control (MPC)

One widely used nonlinear MPC package is `acados` [Verscheuren et al. 2021]

Ex … za)



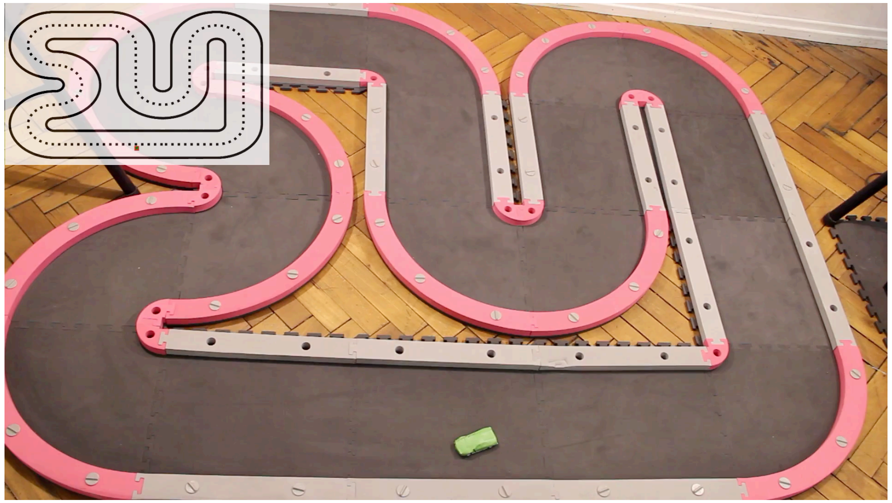
La …

di …

Nonlinear Optimal Control often used for Model Predictive Control (MPC)

One widely used nonlinear MPC package is `acados` [Verscheuren et al. 2021]

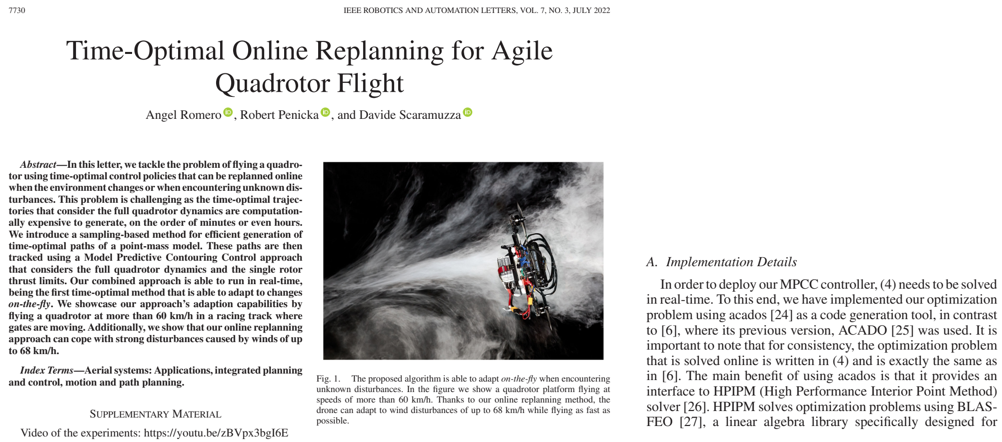
Example 1: Autonomous Driving (in Freiburg)



Example 2: Quadrotor Racing (U Zurich, Scaramuzza)

Paper: https://ieeexplore.ieee.org/abstract/document/9805699

Video: https://www.youtube.com/watch?v=zBVpx3bgI6E



7730 IEEE ROBOTICS AND AUTOMATION LETTERS, VOL. 7, NO. 3, JULY 2022

Time-Optimal Online Replanning for Agile Quadrotor Flight

Angel Romero, Robert Penicka, and Davide Scaramuzza

Abstract—**In this letter, we tackle the problem of flying a quadrotor using time-optimal control policies that can be replanned online when the environment changes or when encountering unknown disturbances. This problem is challenging as the time-optimal trajectories that consider the full quadrotor dynamics are computationally expensive to generate, on the order of minutes or even hours. We introduce a sampling-based method for efficient generation of time-optimal paths of a point-mass model. These paths are then tracked using a Model Predictive Contouring Control approach that considers the full quadrotor dynamics and the single rotor thrust limits. Our combined approach is able to run in real-time, being the first time-optimal method that is able to adapt to changes *on-the-fly*. We showcase our approach's adaption capabilities by flying a quadrotor at more than 60 km/h in a racing track where gates are moving. Additionally, we show that our online replanning approach can cope with strong disturbances caused by winds of up to 68 km/h.**

Index Terms—**Aerial systems: Applications, integrated planning and control, motion and path planning.**

SUPPLEMENTARY MATERIAL

Video of the experiments: https://youtu.be/zBVpx3bgI6E

Fig. 1. The proposed algorithm is able to adapt *on-the-fly* when encountering unknown disturbances. In the figure we show a quadrotor platform flying at speeds of more than 60 km/h. Thanks to our online replanning method, the drone can adapt to wind disturbances of up to 68 km/h while flying as fast as possible.

A. Implementation Details

In order to deploy our MPCC controller, (4) needs to be solved in real-time. To this end, we have implemented our optimization problem using acados [24] as a code generation tool, in contrast to [6], where its previous version, ACADO [25] was used. It is important to note that for consistency, the optimization problem that is solved online is written in (4) and is exactly the same as in [6]. The main benefit of using acados is that it provides an interface to HPIPM (High Performance Interior Point Method) solver [26]. HPIPM solves optimization problems using BLAS-FEO [27], a linear algebra library specifically designed for

Latest `acados` development:

differentiable nonlinear MPC via adjoint approach [Frey et al. 2025, subm.]

Next Challenge: Nonsmooth Optimal Control

Next Challenge: Nonsmooth Optimal Control

Continuous-Time OCP

$$
\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T L_{\mathrm{c}}(x(t),u(t))\,\mathrm{d}t + E(x(T)) \\
\text{s.t.} \quad & x(0) = \bar{x}_0 \\
& \dot{x}(t) = f(x(t),u(t)) \\
& 0 \geq h(x(t),u(t)),\ t \in [0,T] \\
& 0 \geq r(x(T))
\end{aligned}
$$

Three levels of difficulty:

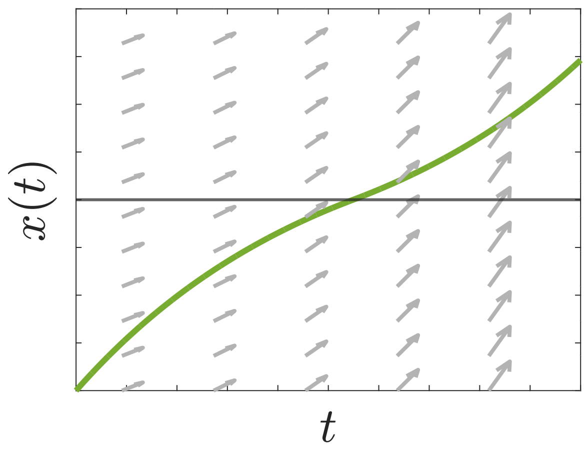
(a) Linear ODE: $f(x,u) = Ax + Bu$

(b) Nonlinear smooth ODE: $f \in \mathcal{C}^1$

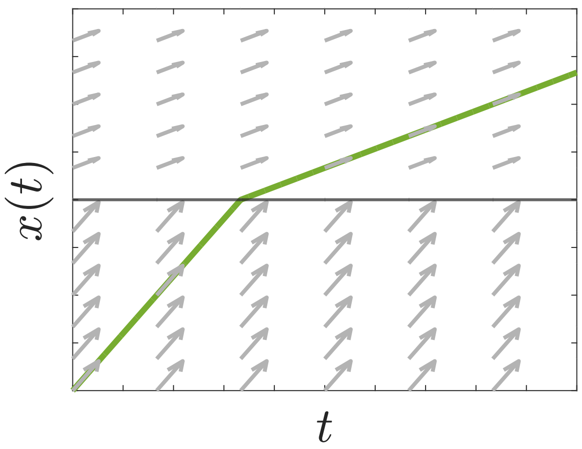
(c) **Nonsmooth Dynamics (NSD):**

- f not differentiable (NSD1),
- f not continuous (NSD2), or even
- f not finite valued, discontinuous state $x(t)$ (NSD3)

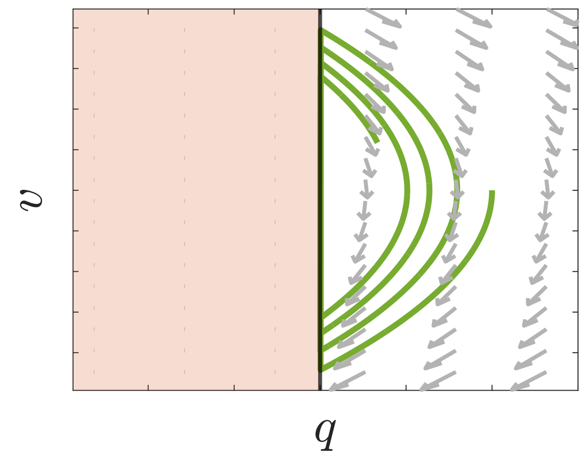
Ordinary differential equation (ODE) with a nonsmooth right-hand side (RHS).



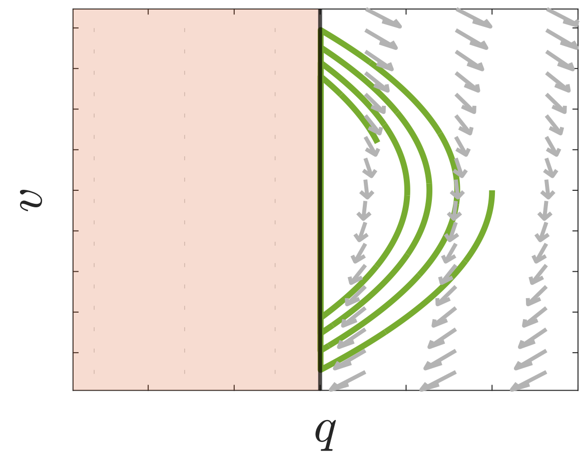
NSD1
non-differentiable RHS



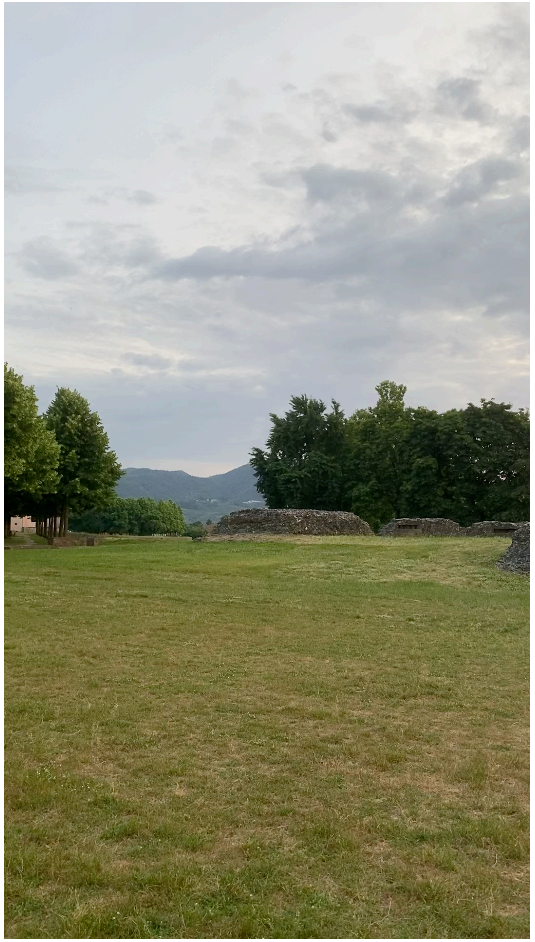
NSD2
discontinuous RHS



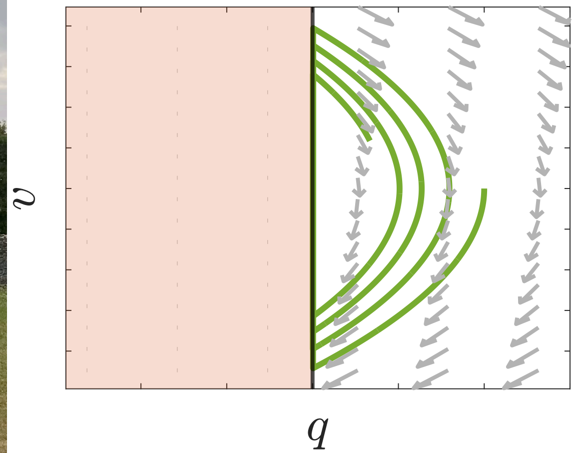
NSD3
state dependent jump



NSD3
state dependent jump



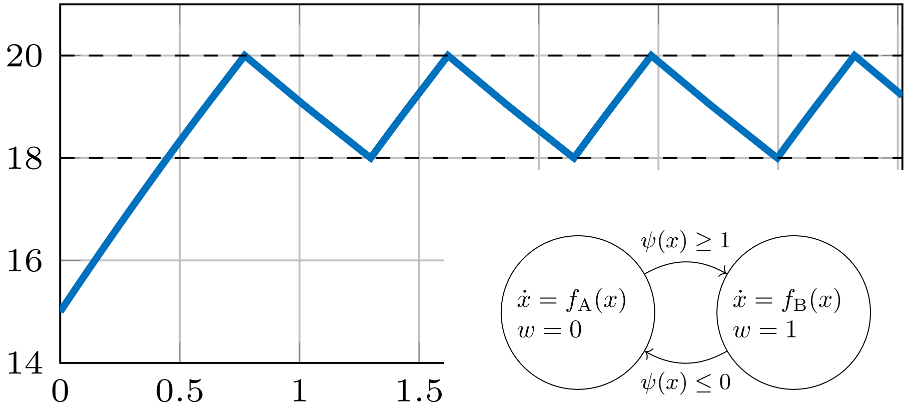
Bouncing Ball (NSD3)



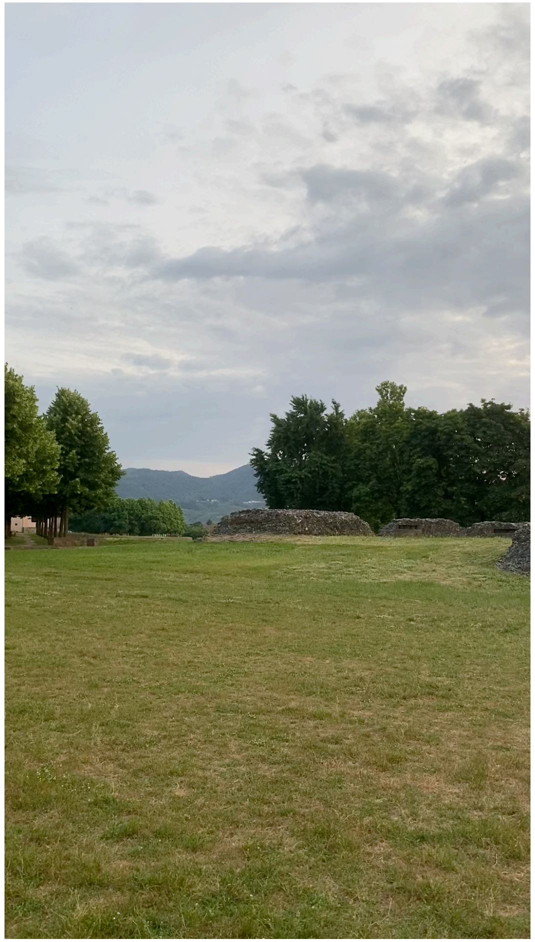
NSD3
state dependent jump

Nonsmooth differential equations - hybrid systems

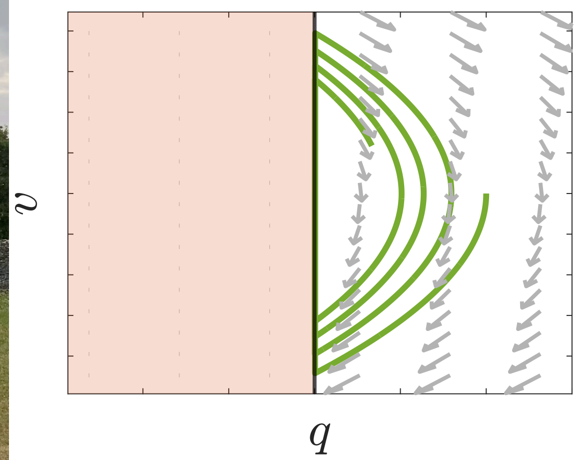
Classification of Nonsmooth Dynamics (NSD)



State Machine in Hysteresis Control (NSD3)



Bouncing Ball (NSD3)

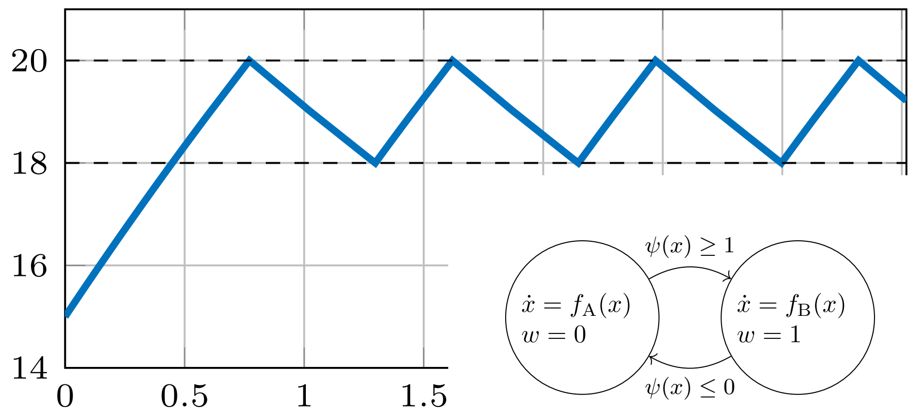


NSD3

state dependent jump

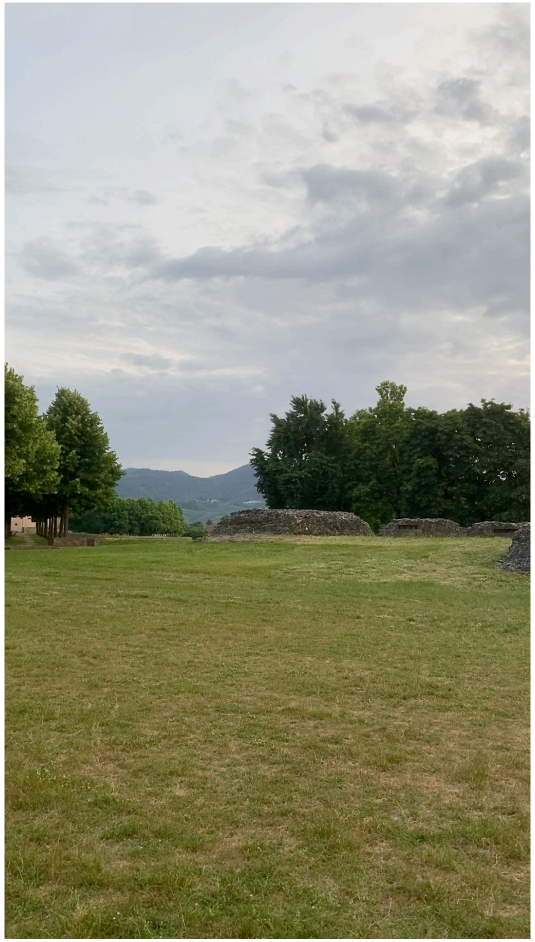
Nonsmooth differential equations - hybrid systems

Classification of Nonsmooth Dynamics (NSD)

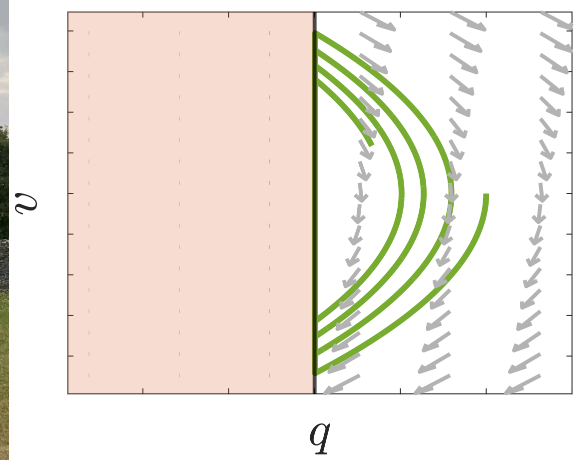


State Machine in Hysteresis Control (NSD3)

Walking Robot (unitree at LAAS, NSD3)



Bouncing Ball (NSD3)



NSD3

state dependent jump

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

Some recent progress in Nonsmooth Optimal Control:

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

Some recent progress in Nonsmooth Optimal Control:

- Can transform many NSD3 systems into (easier) NSD2 via **time-freezing**

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

Some recent progress in Nonsmooth Optimal Control:

- Can transform many NSD3 systems into (easier) NSD2 via **time-freezing**
- Can discretize NSD2 systems with highly accurate **Finite Elements with Switch Detection (FESD),** removing spurious local minimisers

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

Some recent progress in Nonsmooth Optimal Control:

- Can transform many NSD3 systems into (easier) NSD2 via **time-freezing**
- Can discretize NSD2 systems with highly accurate **Finite Elements with Switch Detection (FESD),** removing spurious local minimisers
- Can solve the resulting **Mathematical Programs with Complementarity Constraints (MPCC)** via homotopy of NLP (e.g. based on IPOPT)

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

Some recent progress in Nonsmooth Optimal Control:

- Can transform many NSD3 systems into (easier) NSD2 via **time-freezing**
- Can discretize NSD2 systems with highly accurate **Finite Elements with Switch Detection (FESD),** removing spurious local minimisers
- Can solve the resulting **Mathematical Programs with Complementarity Constraints (MPCC)** via homotopy of NLP (e.g. based on IPOPT)
- Can use open-source software **NOSNOC,** from MATLAB and Python

`github.com/nosnoc/nosnoc`

Can Newton-Type Optimization be Useful for NSD3 Systems ?

Surprisingly, Yes !

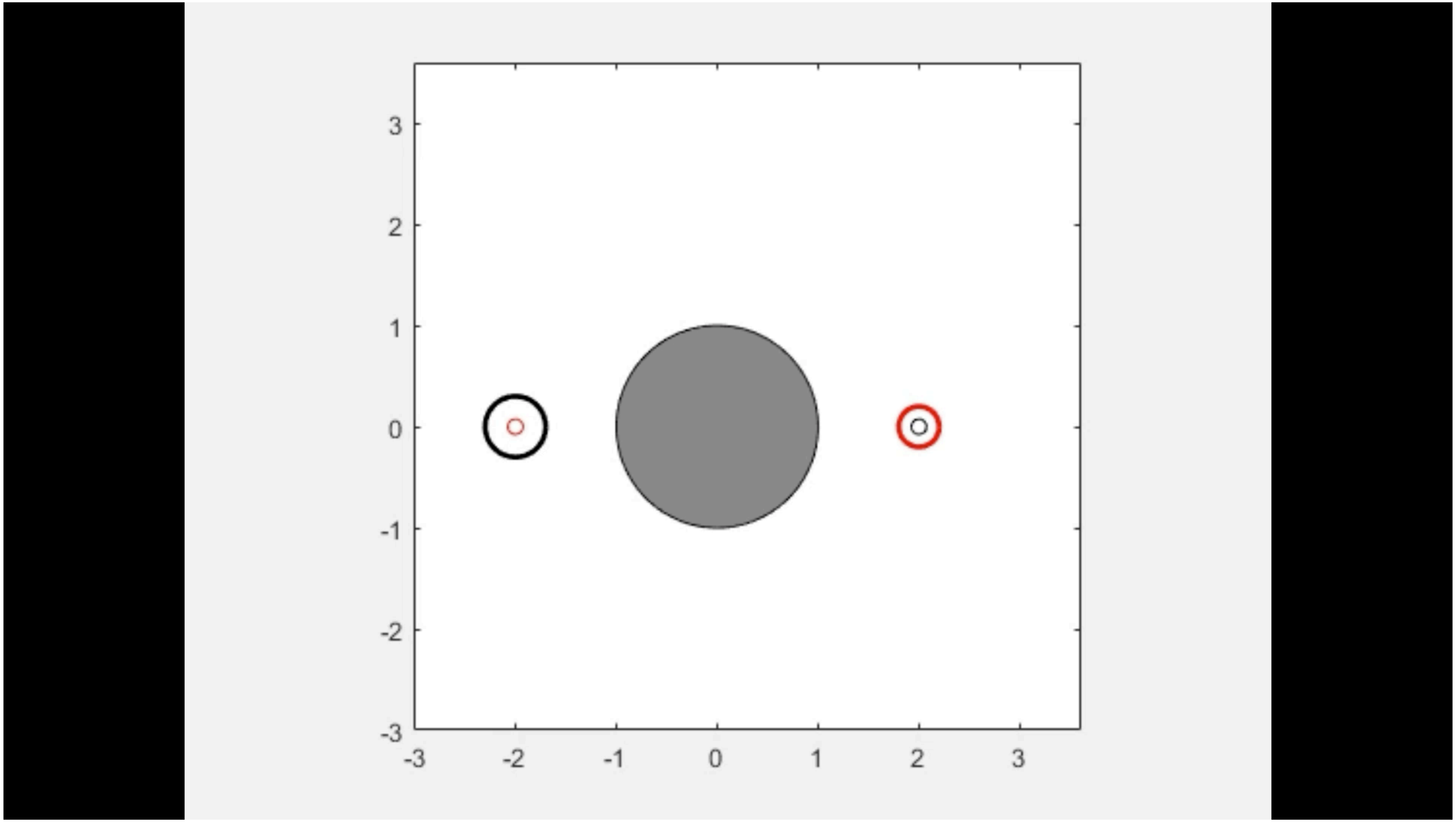
Some recent progress in Nonsmooth Optimal Control:

- Can transform many NSD3 systems into (easier) NSD2 via **time-freezing**
- Can discretize NSD2 systems with highly accurate **Finite Elements with Switch Detection (FESD),** removing spurious local minimisers
- Can solve the resulting **Mathematical Programs with Complementarity Constraints (MPCC)** via homotopy of NLP (e.g. based on IPOPT)
- Can use open-source software **NOSNOC,** from MATLAB and Python

`github.com/nosnoc/nosnoc`

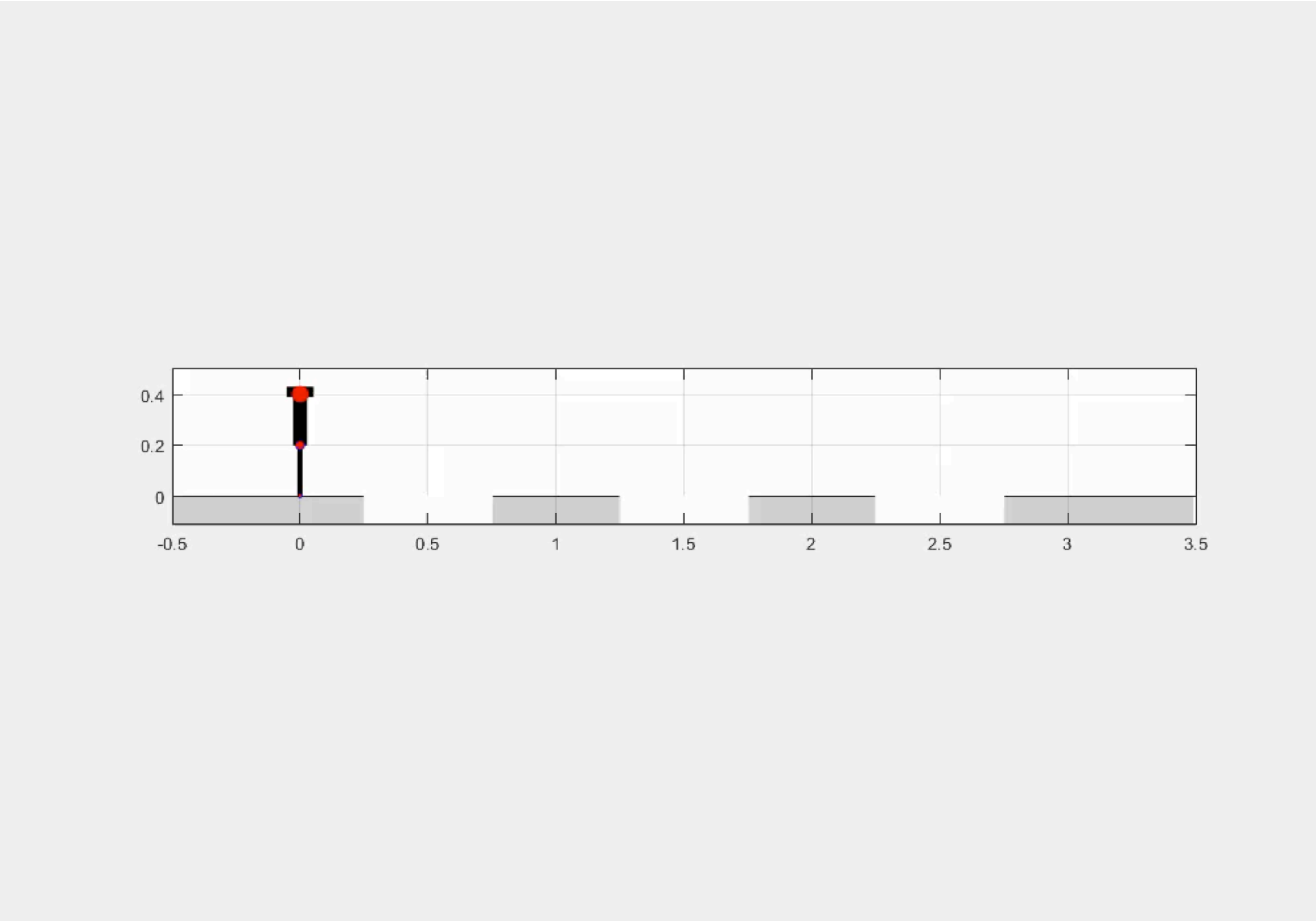
PhD and Postdoc Work by **Armin Nurkanovic** (currently serving as replacement professor of mathematical optimization at Technical University of Braunschweig)

NOSNOC examples

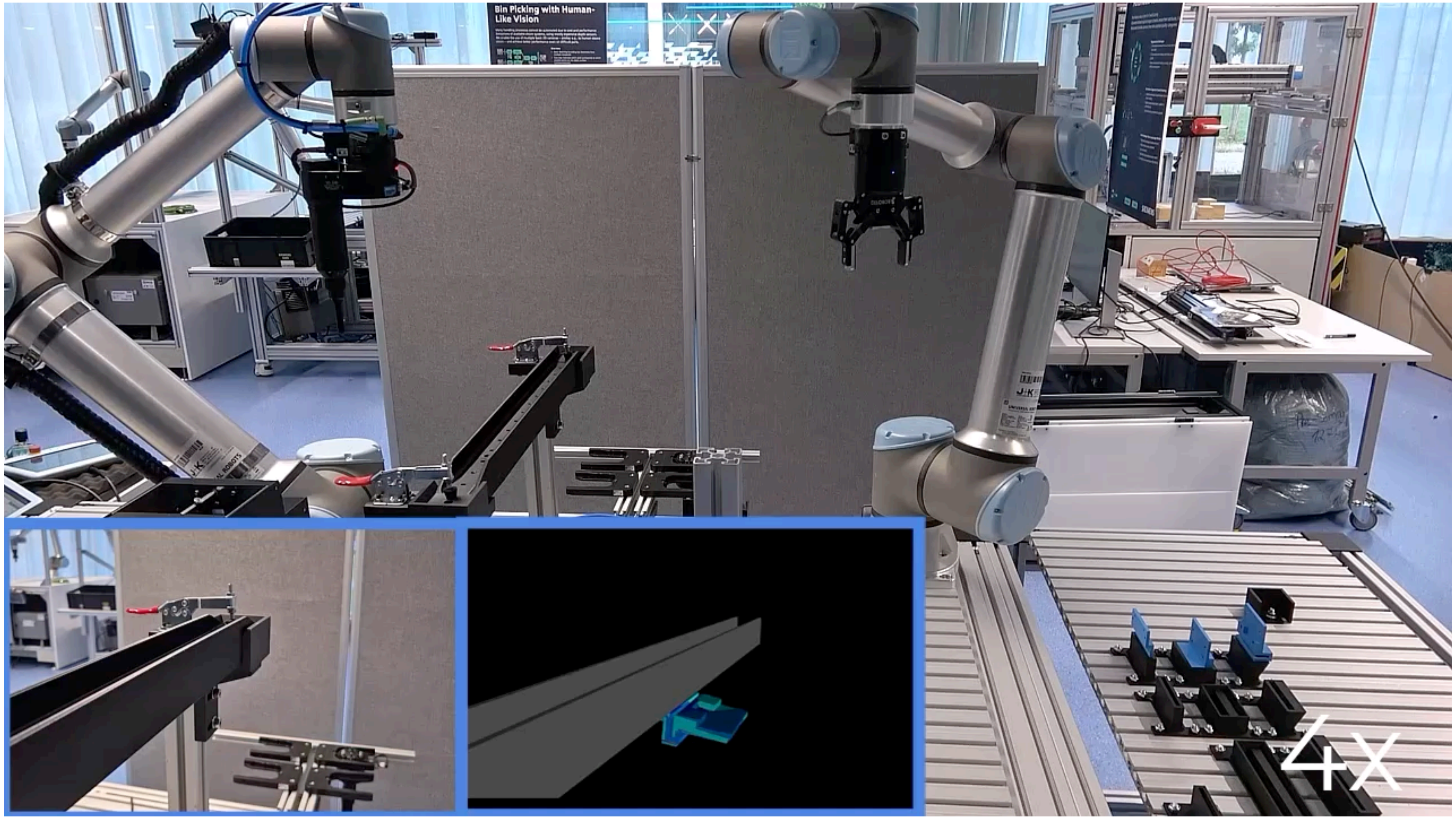


Hopping robot - move with minimal effort from start to end position

Homotopy initialized with start position everywhere. Optimizer finds creative solution.

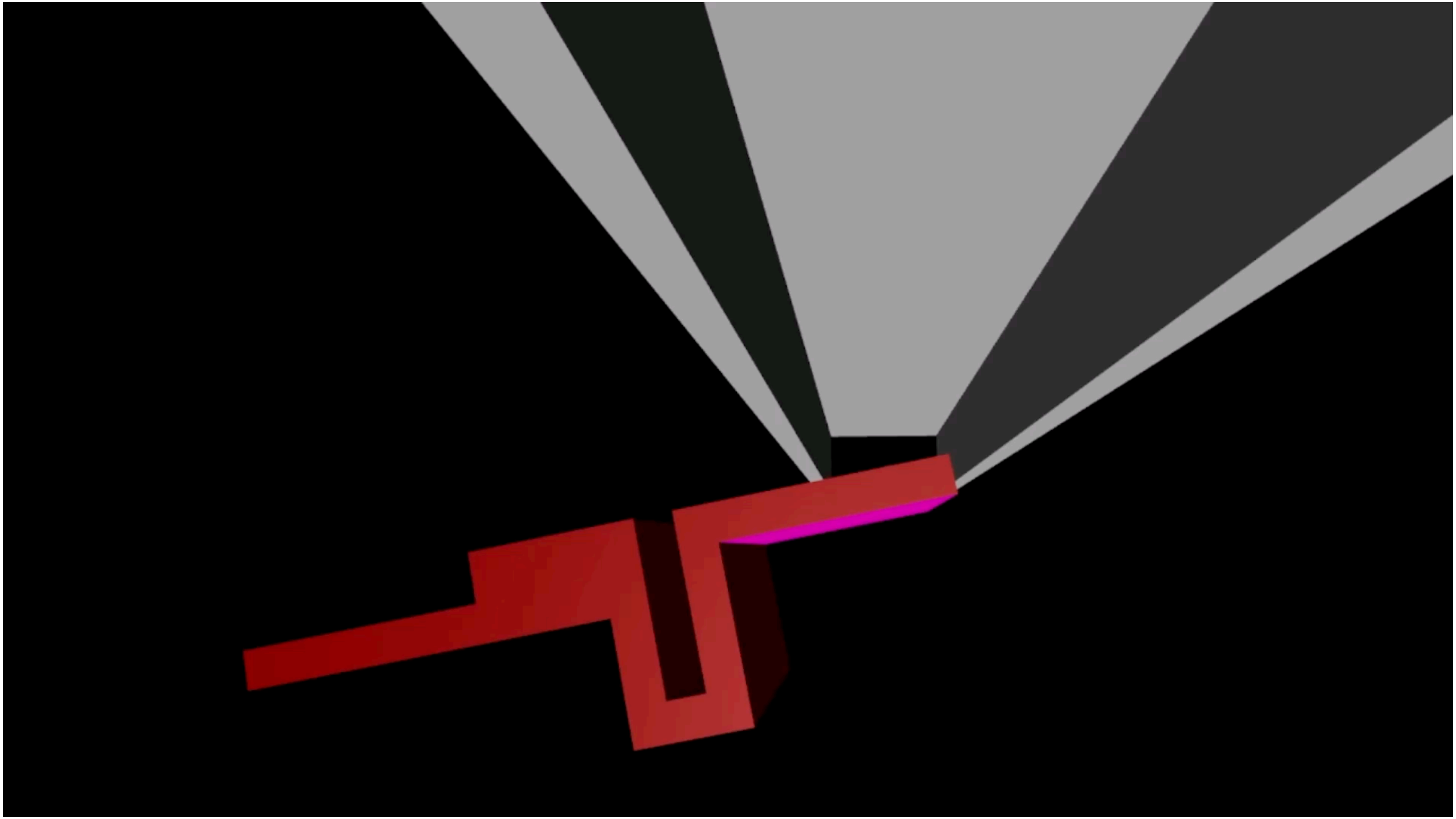


Today's Focus: Assembly Robots at Siemens Research in Munich (NSD3)

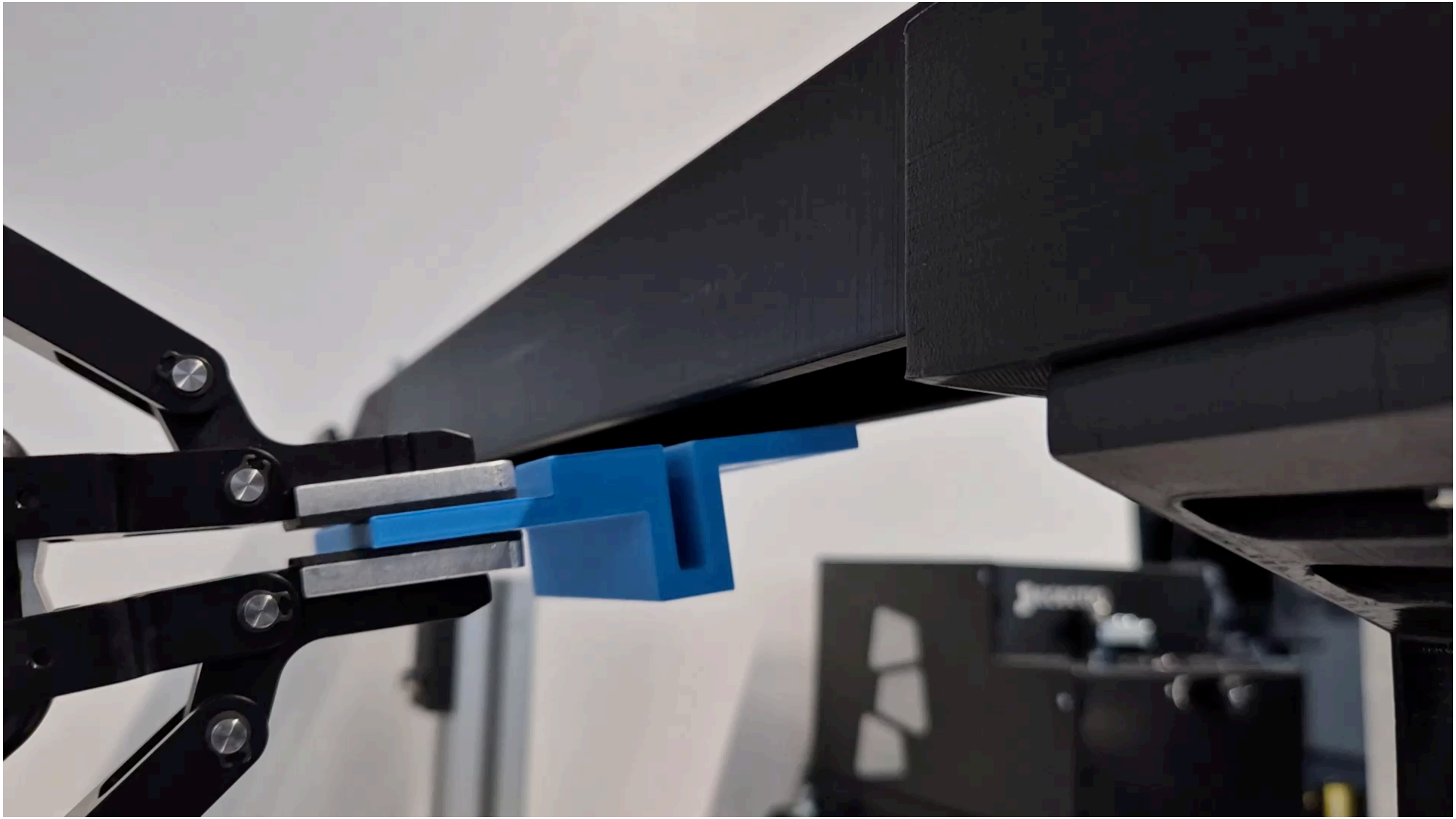


Christian Dietz (MSc Mathematics) industrial PhD student at University of Freiburg, supervised by Armin Nurkanovic and MD

Dream: just specify start and end position…but linear interpolation does not work
(simulation)



Dream: just specify start and end position…but linear interpolation does not work
(experiment)



How to formulate and solve OCP for assembly robot at Siemens?

1. Divide colliding bodies each into rigidly connected convex polyhedra
2. Define Signed Distance Function (SDF) between polyhedra
3. Compute Contact Normal of SDF (unique if slightly smoothed)
4. Formulate Complementarity Lagrangian System Model (NSD3)
5. Discretize OCP with Time-Stepping Method (Implicit Euler, Fixed Steps)
6. Solve resulting Mathematical Program with Complementarity Constraints (MPCC) via Scholtes' Relaxation Method
7. Play open-loop control trajectory on real robot

How to formulate and solve OCP for assembly robot at Siemens?

1. Divide colliding bodies each into rigidly connected convex polyhedra
2. **Define Signed Distance Function (SDF) between polyhedra**
3. **Compute Contact Normal of SDF (unique if slightly smoothed)**
4. Formulate Complementarity Lagrangian System Model (NSD3)
5. Discretize OCP with Time-Stepping Method (Implicit Euler, Fixed Steps)
6. Solve resulting Mathematical Program with Complementarity Constraints (MPCC) via Scholtes' Relaxation Method
7. Play open-loop control trajectory on real robot

Optimization-based signed distance function (SDF) for polytopes

Halfspace representation of polytopes for $n_\mathrm{w} \in \{2,3\}$:

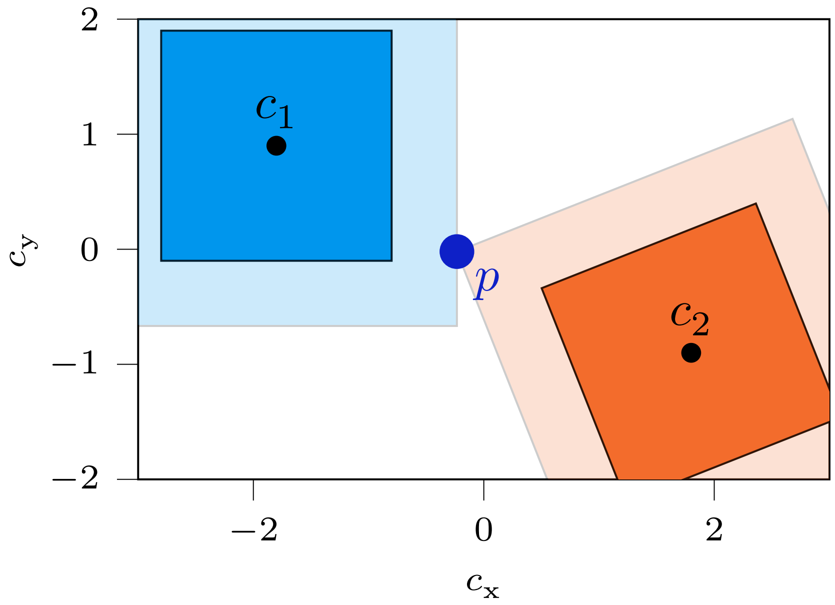
$$\mathcal{P}_1 = \{p \in \mathbb{R}^{n_\mathrm{w}} \mid G_1 p \le h_1\},\ \mathcal{P}_2 = \{p \in \mathbb{R}^{n_\mathrm{w}} \mid G_2 p \le h_2\}.$$

Associating degrees of freedom:

- ρ_i center of mass of i-th polytope
- ξ_i orientation of i-th polytope
- System configuration: $q = (\rho_1, \xi_2, \rho_2, \xi_2)$
- $R(\xi_i)$ - rotation matrices

Calculating the SDF as growth distance:

$$\begin{aligned}\Phi_0(q) = \min_{p,\alpha} \quad & \alpha \\ \text{s.t.} \quad & G_1 R(\xi_1)^\top (p - \rho_1) \le (1+\alpha) h_1, \\ & G_2 R(\xi_2)^\top (p - \rho_2) \le (1+\alpha) h_2.\end{aligned}$$



Smoothing the signed distance function

The optimization-based SDF is given by a parametric linear program

$$\begin{aligned} \Phi_0(q) = \min_{z} &\quad c^\top z \\ \text{s.t.} &\quad A(q)z \leq b(q), \end{aligned}$$

with primal variables $z = (p, \alpha)$.
Perturbed KKT conditions as considered in interior-point methods with barrier parameter $\tau > 0$ are given by

$$\begin{aligned} 0 &= c + A(q)^\top \lambda, \\ y &= b(q) - A(q)z, \\ \lambda_i y_i &= \tau, \quad i = 1, \ldots, m, \\ \lambda &> \mathbf{0}, y > \mathbf{0}, \end{aligned}$$

λ are Lagrange multipliers and y are inequality constraint slacks.

Smoothing the signed distance function (1)

By writing the equality conditions compactly the perturbed KKT conditions are denoted by

$$F_\tau(\gamma; q) = \mathbf{0},$$
$$\lambda > \mathbf{0}, y > \mathbf{0},$$

with primal, dual and slack variables $\gamma = (z, \lambda, y)$.

Proposition

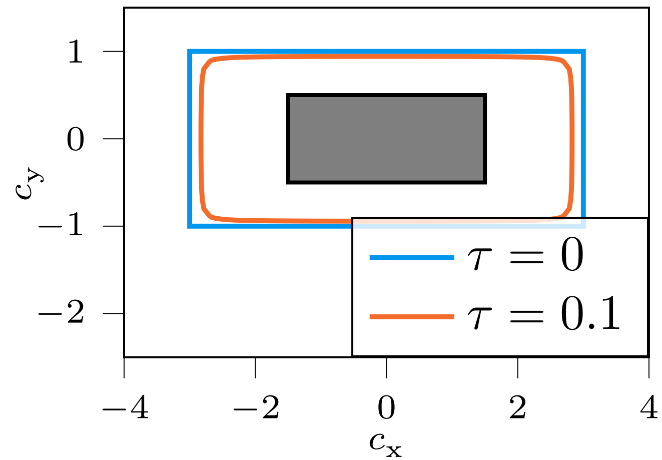
The solution $\gamma_\tau = (z_\tau, \lambda_\tau, y_\tau)$ of the perturbed optimality conditions exists and is unique.[1]

This implies that the distance function defined by

$$\Phi_\tau(q) = \{\alpha \mid F_\tau(\gamma_\tau; q) = \mathbf{0}, \lambda_\tau > \mathbf{0}, y_\tau > \mathbf{0}\},$$

is well-defined for $\tau > 0$.

Level lines $\Phi_\tau(q) = 1$, q point mass



[1] C. Dietz, S. Albrecht, A. Nurkanović, M. Diehl. *Smoothed Distance Functions for Direct Optimal Control of Contact-Rich Systems.* European Control Conference (ECC) 2025.

Contact normal approximation for the smooth SDF

Recap on definitions

The SDF is given by

$$\Phi_0(q) = \min_{z} \quad c^\top z \quad \text{s.t.} \quad A(q)z \leq b(q), \tag{1}$$

with inequality constraint slacks

$$y(z, q) = b(q) - A(q)z.$$

We additionally define

- $\bar{Z}(q)$ denotes the set of all primal optimal solutions to (1)
- $\bar{\Lambda}(q)$ denotes the set of all corresponding dual optimal solutions

- Modelling of contact-rich systems requires definition of a contact normal vector
- Normally the contact normal is chosen as the gradient of the SDF (results in third-order sensitivities in Newton-type optimization!)

Directional derivatives at an exact solution:[2]

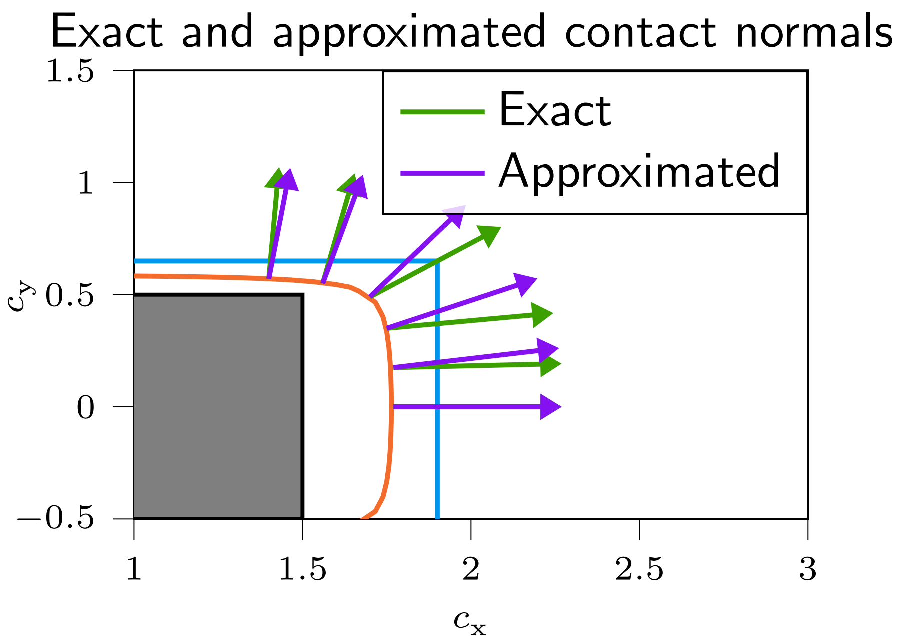
$$\partial_d \Phi_0(q) = \min_{z \in \bar{Z}(q)} \max_{\lambda \in \bar{\Lambda}(q)} -d^\top \nabla_q y(z, q)\lambda,$$

Proposed contact normal approximation:

$$n_\tau(q) = \frac{-\nabla_q y(z_\tau, q)\lambda_\tau}{\|\nabla_q y(z_\tau, q)\lambda_\tau\|_2}.$$

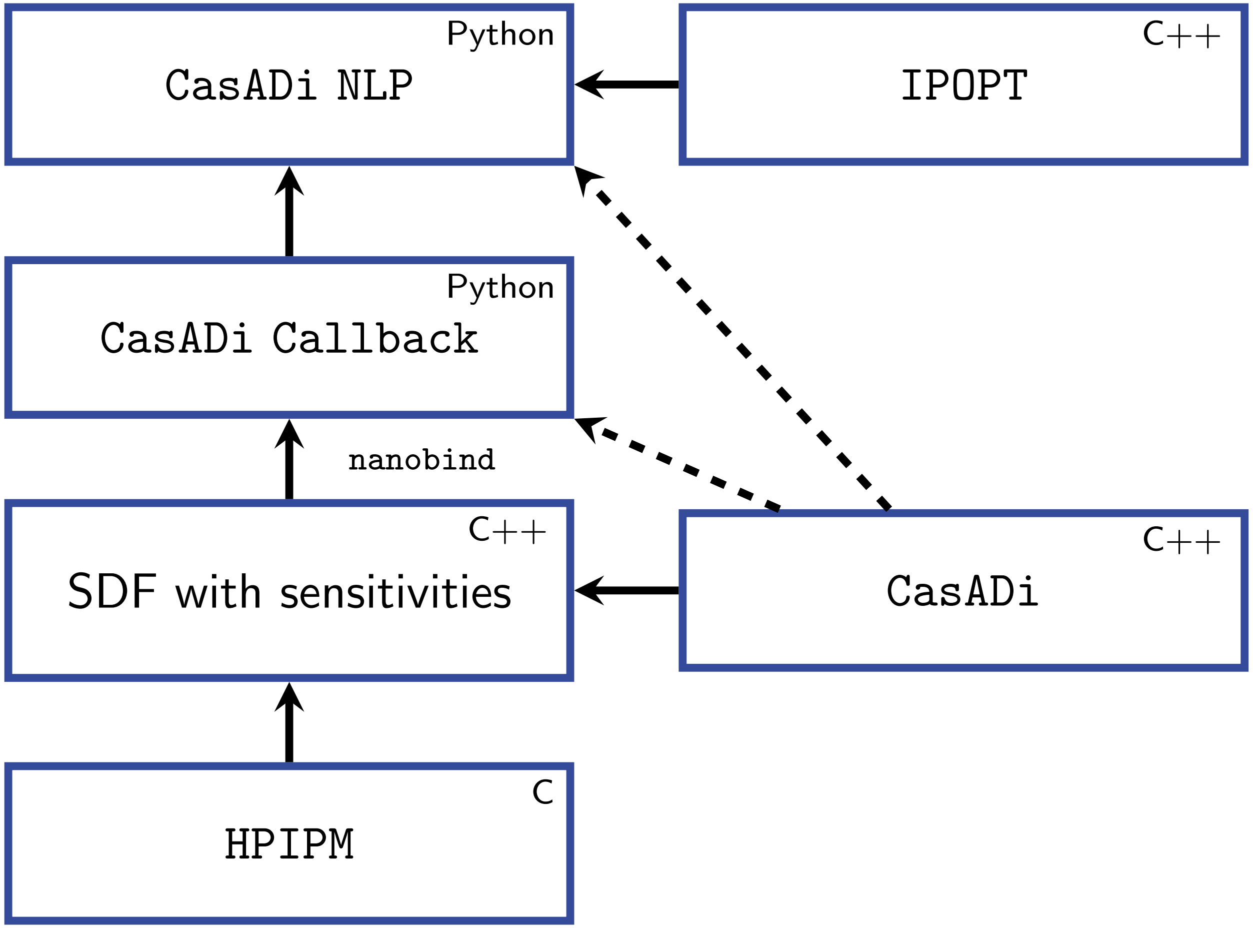
[2] W. Hogan. *Directional derivatives for extremal-value functions with applications to the comple*

Contact normal approximation for the smooth SDF



SDF implementation

- Numerical experiments use the `CasADi` toolbox through its `Python` interface and `IPOPT` as solver
- The SDF is specified through `CasADi`'s `Callback` class
- `HPIPM` is used to solve the distance problems up to barrier parameter $\tau > 0$
- A `C++` wrapper is used to efficiently manage `HPIPM` structures and parallel computing
- `C++` code is interfaced back to `Python` by using the `nanobind` library



How to formulate and solve OCP for assembly robot at Siemens?

1. Divide colliding bodies each into rigidly connected convex polyhedra
2. Define Signed Distance Function (SDF) between polyhedra
3. Compute Contact Normal of SDF (unique if slightly smoothed)
4. **Formulate Complementarity Lagrangian System Model (NSD3)**
5. **Discretize OCP with Time-Stepping Method (Implicit Euler, Fixed Steps)**
6. Solve resulting Mathematical Program with Complementarity Constraints (MPCC) via Scholtes' Relaxation Method
7. Play open-loop control trajectory on real robot

Robust contact-implicit trajectory optimization

Continuous-time contact-rich dynamical system:

$$
\begin{aligned}
&\dot{q} = \nu, \\
&M\dot{\nu} = u + \sum_{i=1}^{n_{\mathrm{d}}} n_{\tau,i}(q)\lambda_{\mathrm{n},i}, \\
&0 \leq \Phi_{\tau,i}(q) \;\perp\; \lambda_{\mathrm{n},i} \geq 0, \quad i = 1, \ldots, n_{\mathrm{d}}, \\
&\text{Multi impact law.}
\end{aligned}
$$

- $\nu \in \mathbb{R}^{n_{\mathrm{q}}}$ system velocity
- $M \in \mathbb{R}^{n_{\mathrm{q}} \times n_{\mathrm{q}}}$ inertia matrix
- $u \in \mathbb{R}^{n_{\mathrm{u}}}$ control input
- $n_{\mathrm{d}} \in \mathbb{N}$ object pairs with smooth SDF $\Phi_{\tau,i}$ and corresponding contact normals $n_{\tau,i}$

Implicit-Euler time-stepping discretization

Time-stepping discretization:

$$\begin{aligned}
&q_{k+1} = q_k + h\nu_{k+1},\\
&\nu_{k+1} = \nu_k + hM^{-1}(u_k + \sum_{i=1}^{n_\mathrm{d}} n_{\tau,i}(q_{k+1})\lambda_{\mathrm{n},k,i}),\\
&\Phi_{\tau,i}(q_{k+1})\lambda_{\mathrm{n},k,i} \leq \sigma,\ \ i = 1,\dots,n_\mathrm{d},\\
&0 \leq \Phi_{\tau,i}(q_{k+1}),\ 0 \leq \lambda_{\mathrm{n},k,i},\ \ i = 1,\dots,n_\mathrm{d},
\end{aligned}$$

with time-step $h > 0$ and using Scholtes' relaxation to relax complementarity constraints with $\sigma > 0$.

Compact notation for the discretized system:

$$\begin{aligned}
H_{\sigma,\tau}(x_k, x_{k+1}, \lambda_{\mathrm{n},k}, u_k) &= \mathbf{0},\\
G_{\sigma,\tau}(x_k, x_{k+1}, \lambda_{\mathrm{n},k}, u_k) &\leq \mathbf{0}.
\end{aligned}$$

How to formulate and solve OCP for assembly robot at Siemens?

1. Divide colliding bodies each into rigidly connected convex polyhedra
2. Define Signed Distance Function (SDF) between polyhedra
3. Compute Contact Normal of SDF (unique if slightly smoothed)
4. Formulate Complementarity Lagrangian System Model (NSD3)
5. Discretize OCP with Time-Stepping Method (Implicit Euler, Fixed Steps)
6. **Solve resulting Mathematical Program with Complementarity Constraints (MPCC) via Scholtes' Relaxation Method**
7. Play open-loop control trajectory on real robot

Numerical methods for MPCCs

MPEC

$$\min_{w\in\mathbb{R}^n} \quad f(w) \tag{3a}$$

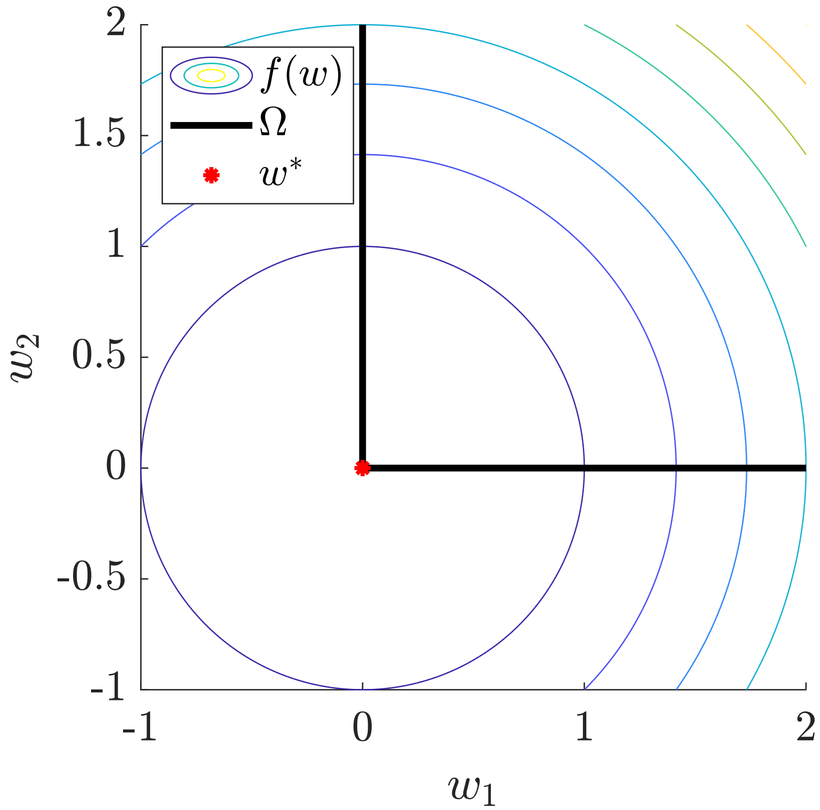
$$\text{s.t.} \quad g(w) = 0, \tag{3b}$$

$$h(w) \geq 0, \tag{3c}$$

$$0 \leq w_1 \perp w_2 \geq 0, \tag{3d}$$

$$w = (w_0, w_1, w_2) \in \mathbb{R}^n, \;\; w_0 \in \mathbb{R}^p, \;\; w_1, w_2 \in \mathbb{R}^m,$$

$$\Omega = \{x \in \mathbb{R}^n \mid g(w) = 0, h(w) \geq 0, \;\; 0 \leq w_1 \perp w_2 \geq 0\},$$



- Standard NLP methods solve the KKT conditions.
- MPECs violate constraint qualifications, and the KKT conditions may not be necessary.
- There are many stationary concepts for MPECs, and not all are useful.

Numerical methods for MPCCs

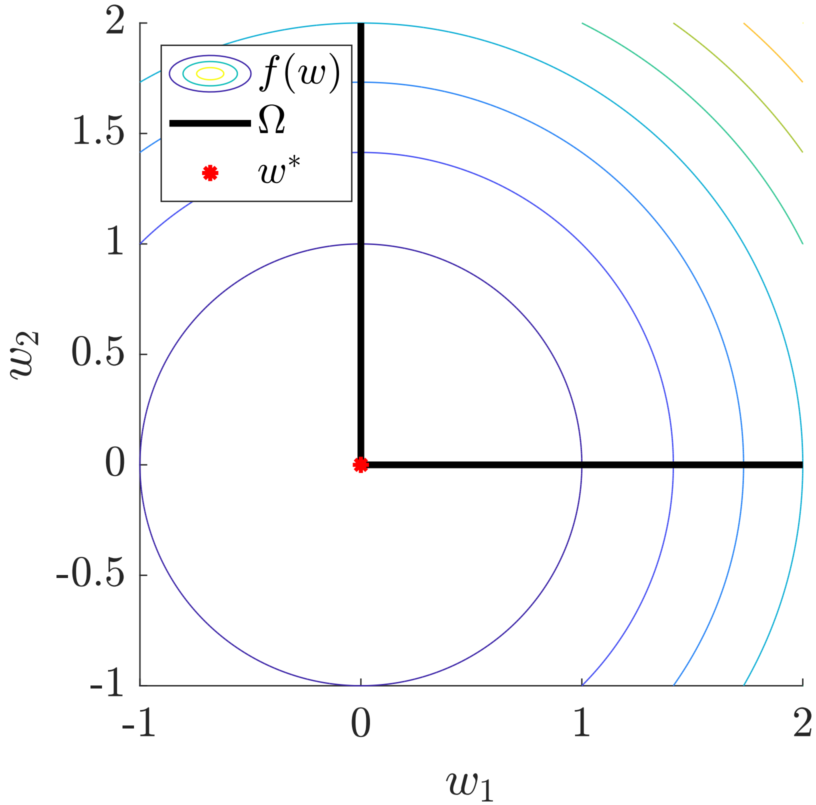
MPEC

$$\min_{w \in \mathbb{R}^n} \quad f(w) \tag{3a}$$

$$\text{s.t.} \quad g(w) = 0, \tag{3b}$$

$$h(w) \geq 0, \tag{3c}$$

$$0 \leq w_1 \perp w_2 \geq 0, \tag{3d}$$



$$w = (w_0, w_1, w_2) \in \mathbb{R}^n, \;\; w_0 \in \mathbb{R}^p, \;\; w_1, w_2 \in \mathbb{R}^m,$$

$$\Omega = \{x \in \mathbb{R}^n \mid g(w) = 0, h(w) \geq 0, \; 0 \leq w_1 \perp w_2 \geq 0\},$$

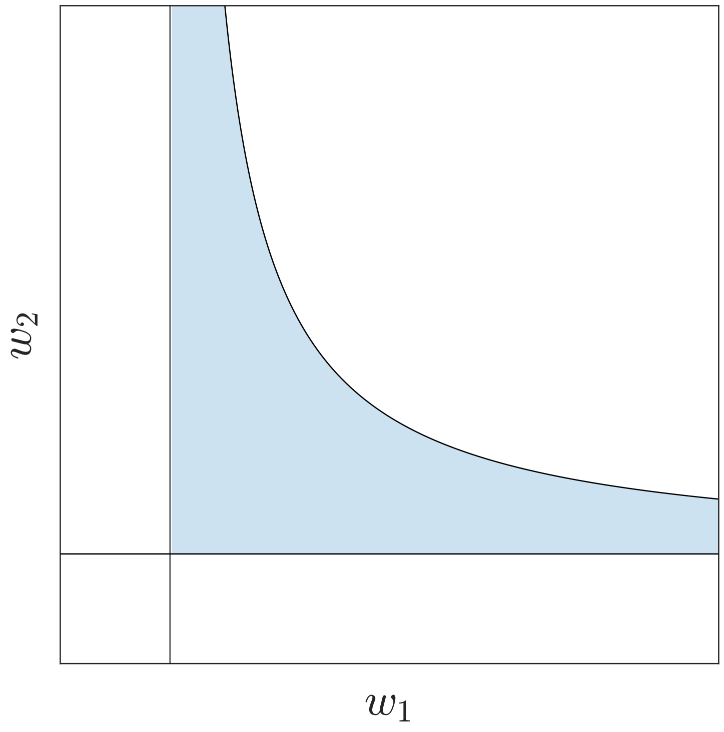
- Standard NLP methods solve the KKT conditions.
- MPECs violate constraint qualifications, and the KKT conditions may not be necessary.
- There are many stationary concepts for MPECs, and not all are useful.
- **Workaround/main idea**: solve a (finite) sequence of more regular problems.

Scholtes' global relaxation method

The easiest to implement and the most efficient regularization method [Scholtes, 2001].

Reg(σ^k)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^n} \quad & f(w) \\
\text{s.t.} \quad & g(w) = 0, \\
& h(w) \geq 0, \\
& w_1, w_2 \geq 0, \\
& w_{1,i} w_{2,i} \leq \sigma^k, \;\; i = 1, \dots, m.
\end{aligned}
$$

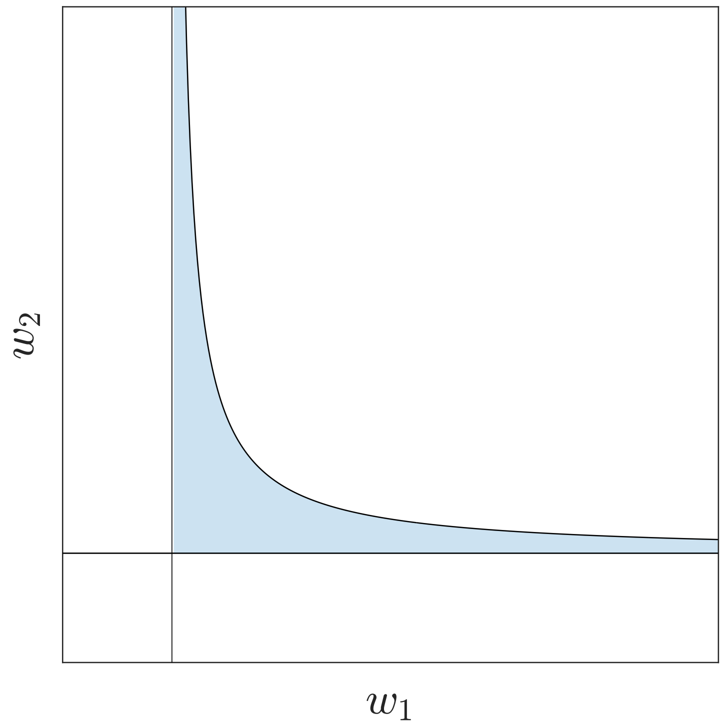


Scholtes' global relaxation method

The easiest to implement and the most efficient regularization method [Scholtes, 2001].

Reg(σ^k)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^n} \quad & f(w) \\
\text{s.t.} \quad & g(w) = 0, \\
& h(w) \geq 0, \\
& w_1, w_2 \geq 0, \\
& w_{1,i} w_{2,i} \leq \sigma^k, \;\; i = 1, \ldots, m.
\end{aligned}
$$

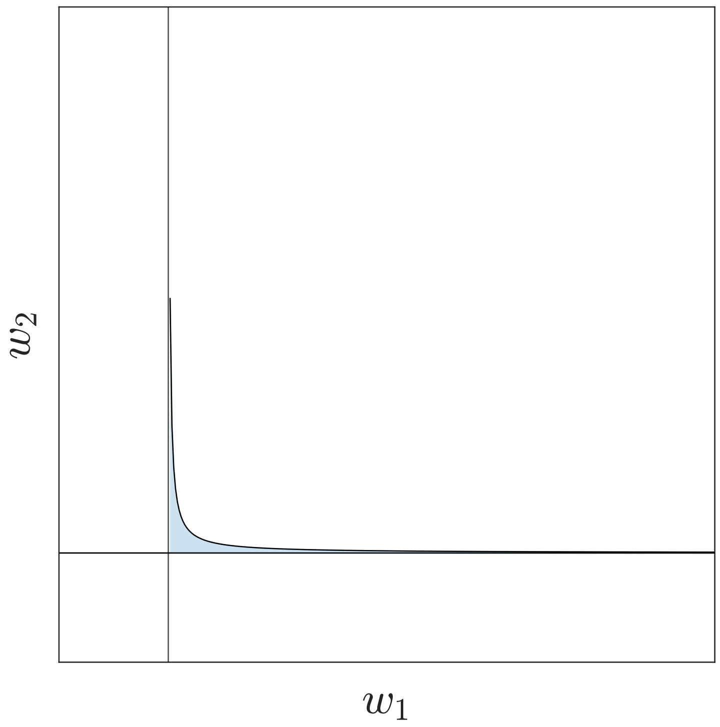


Scholtes' global relaxation method

The easiest to implement and the most efficient regularization method [Scholtes, 2001].

Reg(σ^k)

$$
\begin{aligned}
\min_{w\in\mathbb{R}^n} \quad & f(w) \\
\text{s.t.} \quad & g(w) = 0, \\
& h(w) \geq 0, \\
& w_1, w_2 \geq 0, \\
& w_{1,i} w_{2,i} \leq \sigma^k, \; i = 1, \dots, m.
\end{aligned}
$$

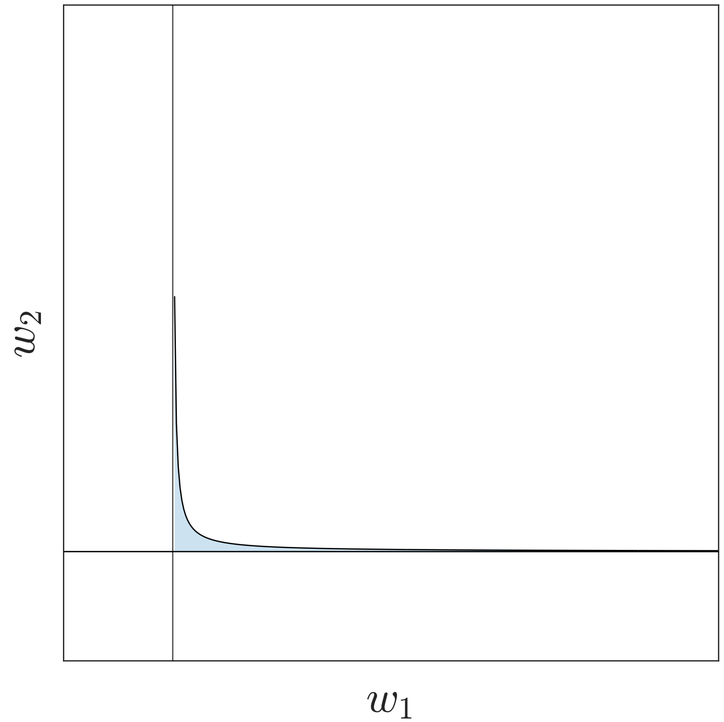


Scholtes' global relaxation method

The easiest to implement and the most efficient regularization method [Scholtes, 2001].

Reg(σ^k)

$$
\begin{aligned}
\min_{w \in \mathbb{R}^n} \quad & f(w) \\
\text{s.t.} \quad & g(w) = 0, \\
& h(w) \geq 0, \\
& w_1, w_2 \geq 0, \\
& w_{1,i} w_{2,i} \leq \sigma^k, \ i = 1, \dots, m.
\end{aligned}
$$

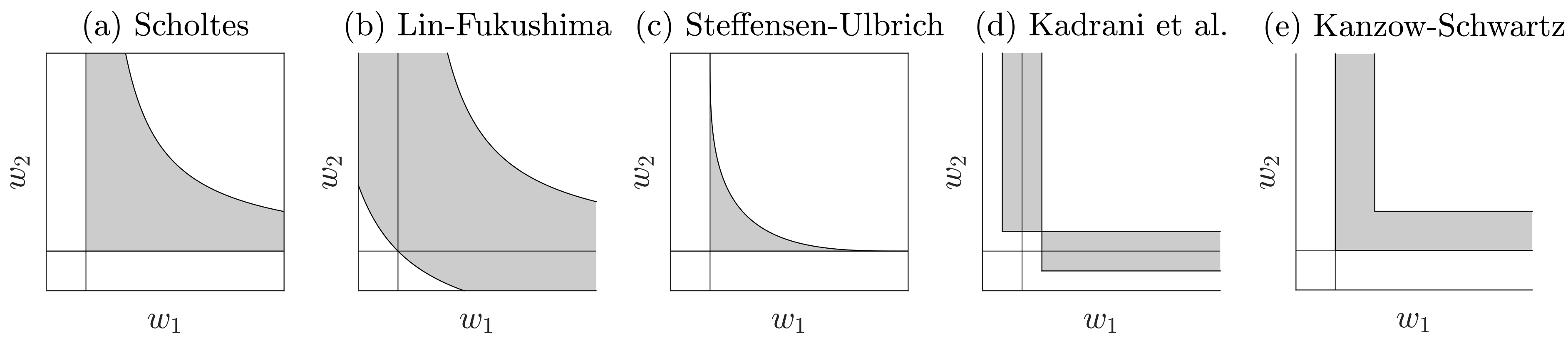


Theorem ([Scholtes, 2001, Hoheisel et al., 2013])

Let $\{\sigma^k\} \downarrow 0$ and let w^k be a stationary point of Reg(σ^k) with $w^k \to w^*$ such that MPEC-MFCQ holds at w^*. Then w^* is a C-stationary point of the the MPEC (3).

Other regularization methods

There exist many elaborate ways to relax the L-shaped set. Convergence theory in [Hoheisel et al., 2013]



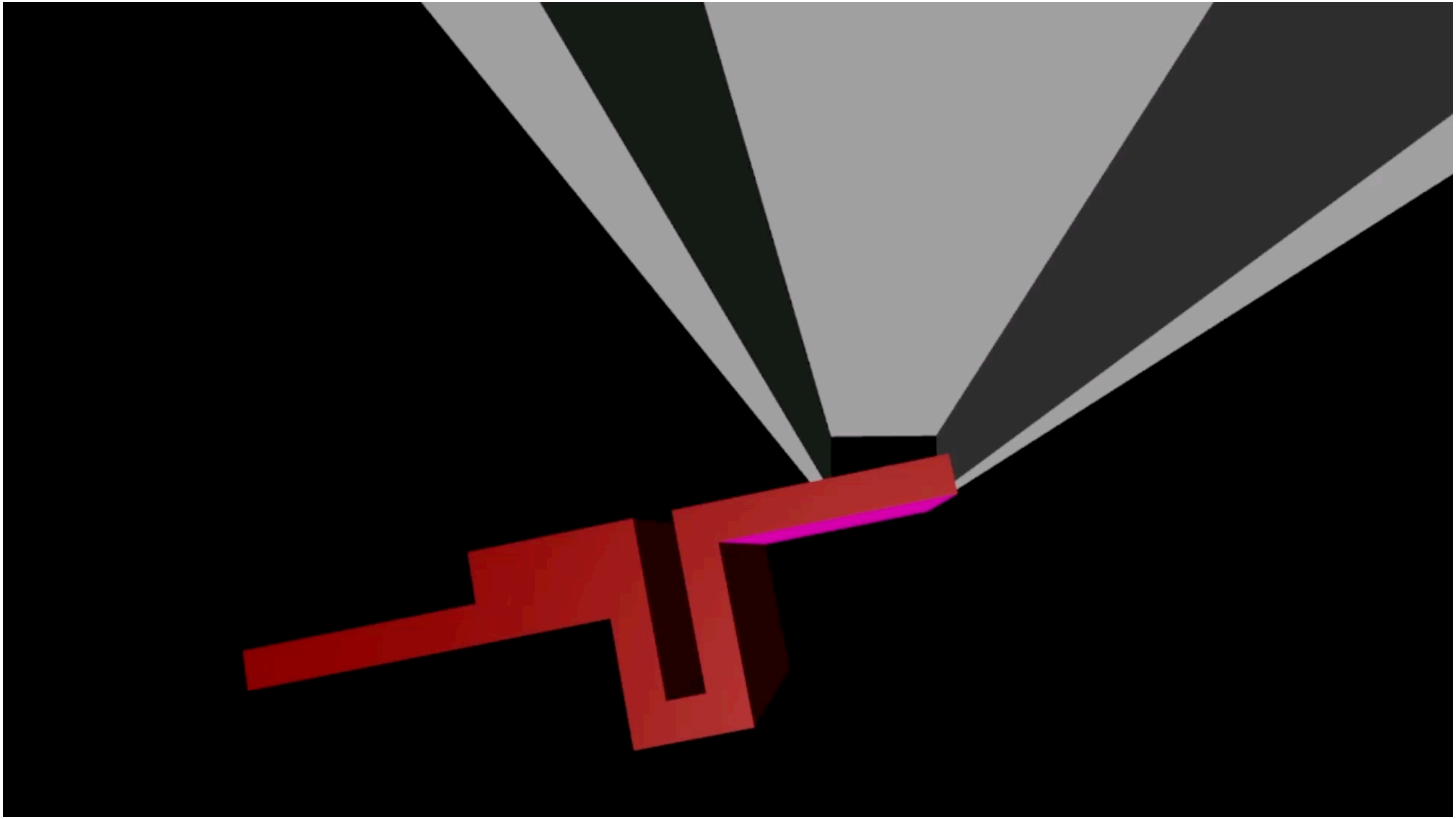
- They have better convergence properties than Scholtes' method if the NLP's are solved exactly.
- In practice, they perform better only on easier problems [Nurkanović et al., 2024].

:

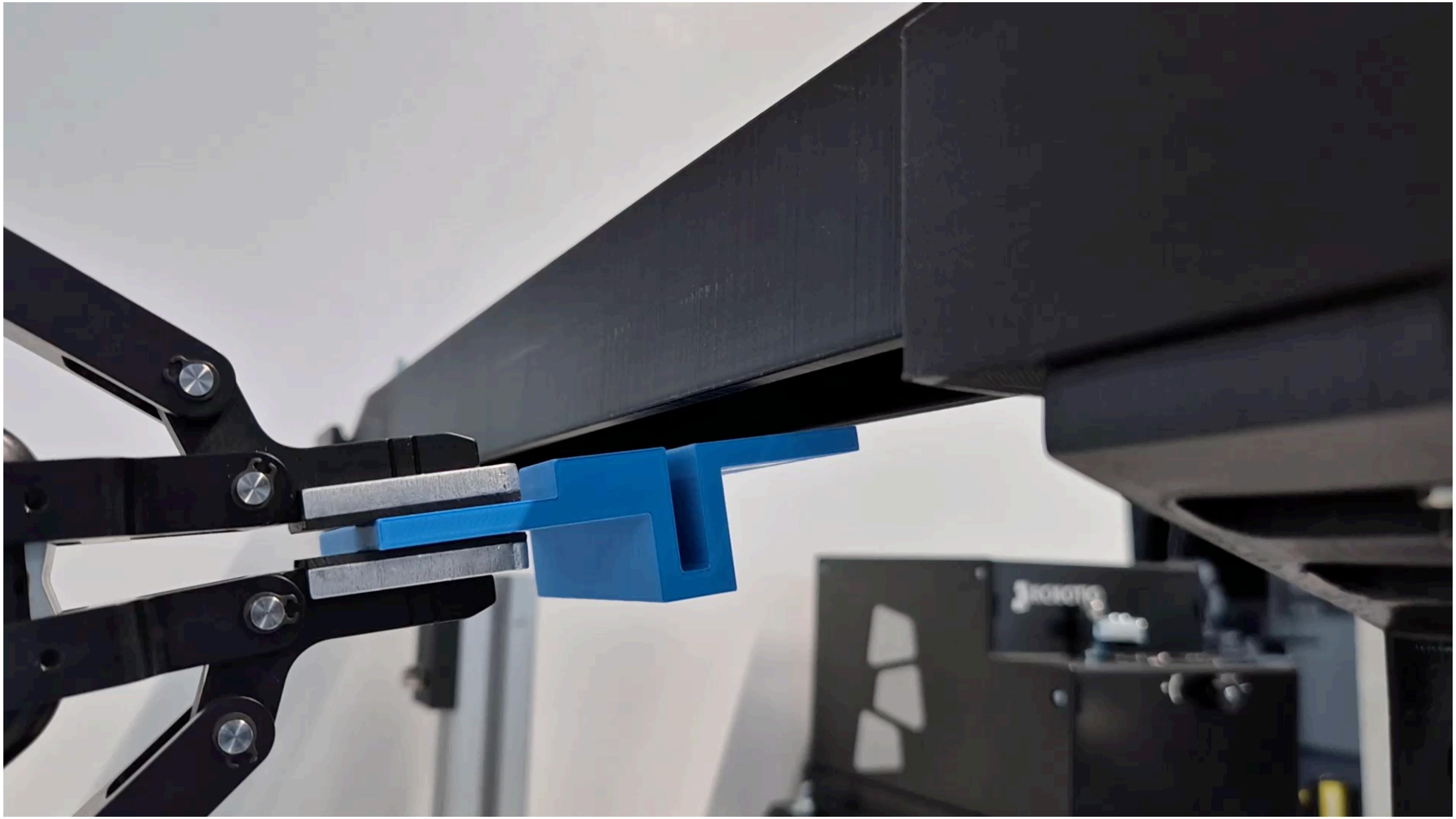
How to formulate and solve OCP for assembly robot at Siemens?

1. Divide colliding bodies each into rigidly connected convex polyhedra
2. Define Signed Distance Function (SDF) between polyhedra
3. Compute Contact Normal of SDF (unique if slightly smoothed)
4. Formulate Complementarity Lagrangian System Model (NSD3)
5. Discretize OCP with Time-Stepping Method (Implicit Euler, Fixed Steps)
6. Solve resulting Mathematical Program with Complementarity Constraints (MPCC) via Scholtes Relaxation Method
7. **Play open-loop control trajectory on real robot**

Solution of Optimal Control Problem (L2-Control Penalty)
(simulation)



Solution of Optimal Control Problem (L2-Control Penalty)
(experiment)

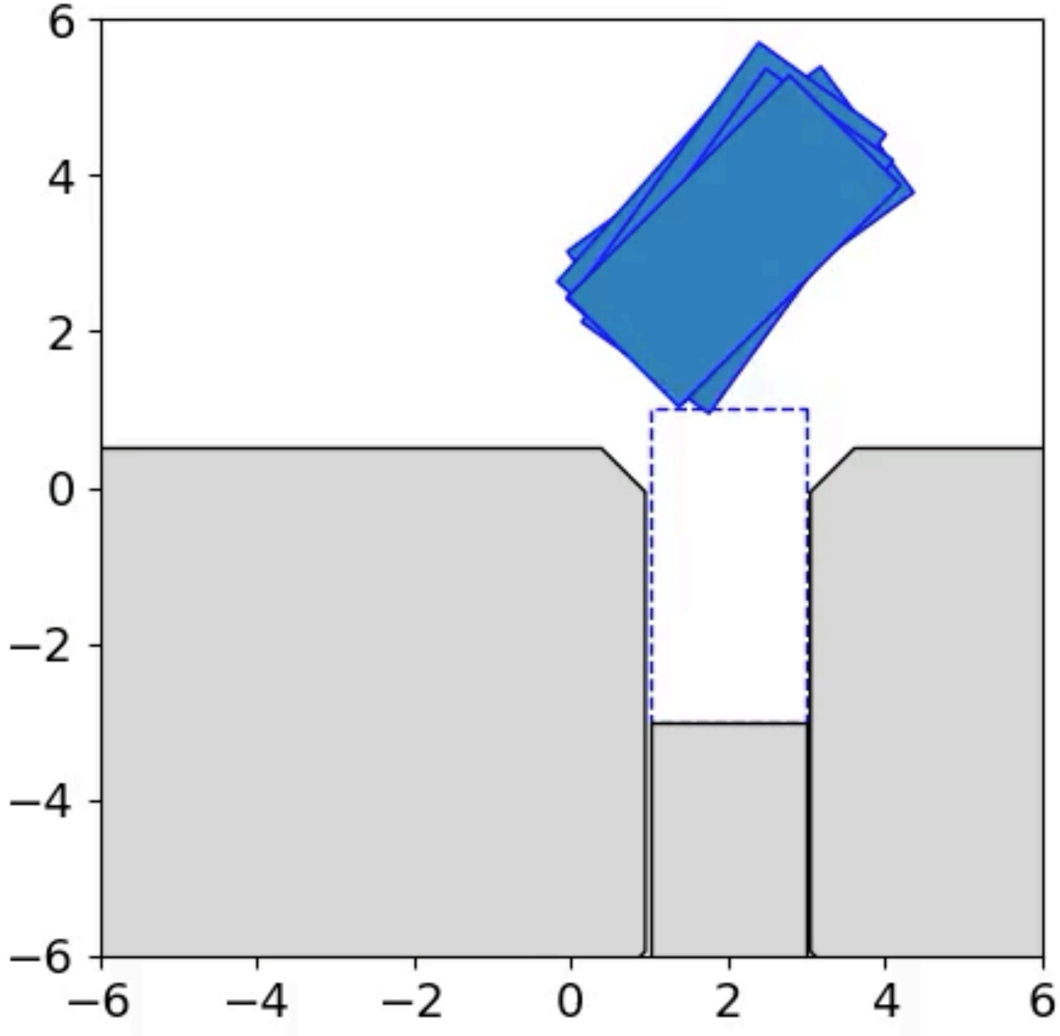


Solution of Optimal Control Problem (L2-Control Penalty)
(experiment)

How to formulate and solve OCP for assembly robot at Siemens?

1. Divide colliding bodies each into rigidly connected convex polyhedra
2. Define Signed Distance Function (SDF) between polyhedra
3. Compute Contact Normal of SDF (unique if slightly smoothed)
4. Formulate Complementarity Lagrangian System Model (NSD3)
5. Discretize OCP with Time-Stepping Method (Implicit Euler, Fixed Steps)
6. Solve resulting Mathematical Program with Complementarity Constraints (MPCC) via Scholtes Relaxation Method
7. ~~**Play open-loop control trajectory on real robot**~~
8. **Robustify by optimizing an ensemble of perturbed trajectories**
9. **Include Real Robot's Internal Impedance Control Law into Model**
10. **Play robust open-loop control trajectory on real robot**

Simulation of robust trajectory for peg in hole

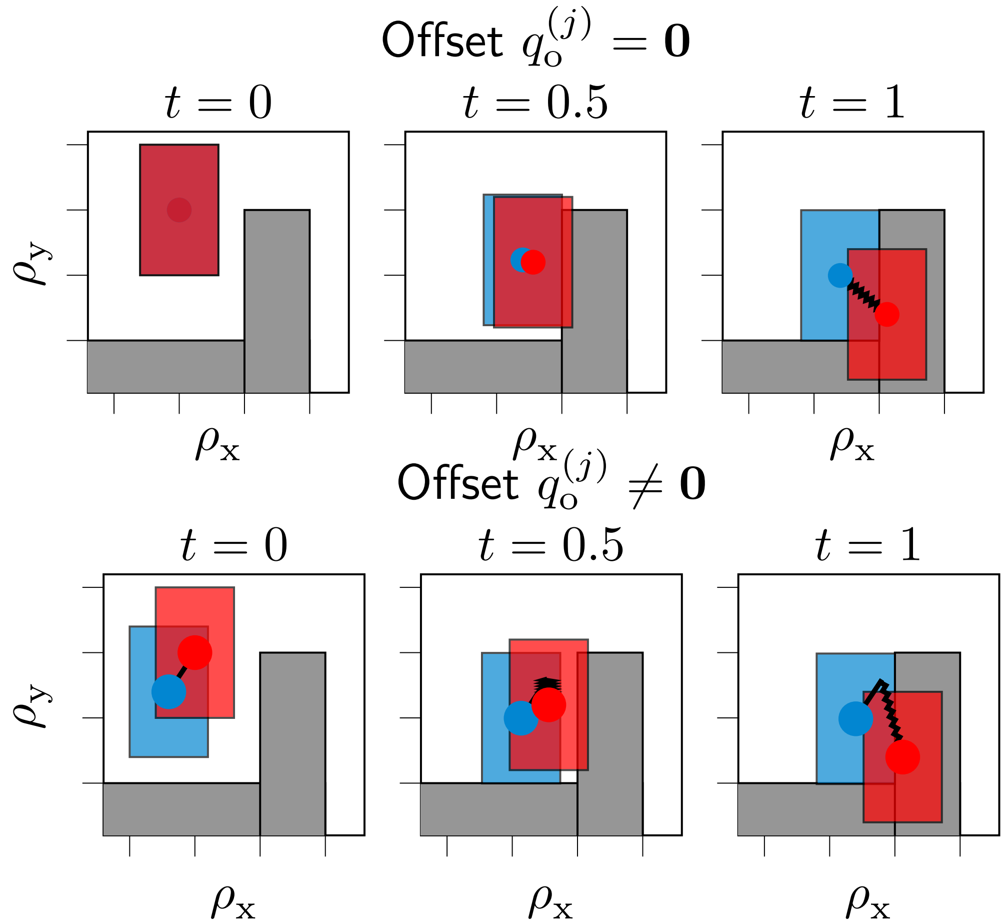


Impedance law for reliable motion execution on real systems

- To achieve closed-loop execution on a real system, we utilize an impedance law as control strategy
- The goal of the planning algorithm is to determine a desired trajectory which results in robust assembly motions if it is tracked by the impedance controller
- For a given desired trajectory $x_\mathrm{d} = (q_\mathrm{d}, \nu_\mathrm{d})$, a trajectory $x_\mathrm{e}^{(j)} = (q_\mathrm{e}^{(j)}, \nu_\mathrm{e}^{(j)})$ in the ensemble is controlled by the impedance force

$$u_j = D(\nu_\mathrm{d} - \nu_\mathrm{e}^{(j)}) + K((q_\mathrm{d} \oplus q_\mathrm{o}^{(j)}) \ominus q_\mathrm{e}^{(j)}),$$

with gain matrices D, K and a fixed offset $q_\mathrm{o}^{(j)}$.



Blue rectangle represents $q_\mathrm{e}^{(j)}$, red rectangle q_d.

Discretization and cost function for robust motion generation

Contact-rich system with quaternion dynamics:

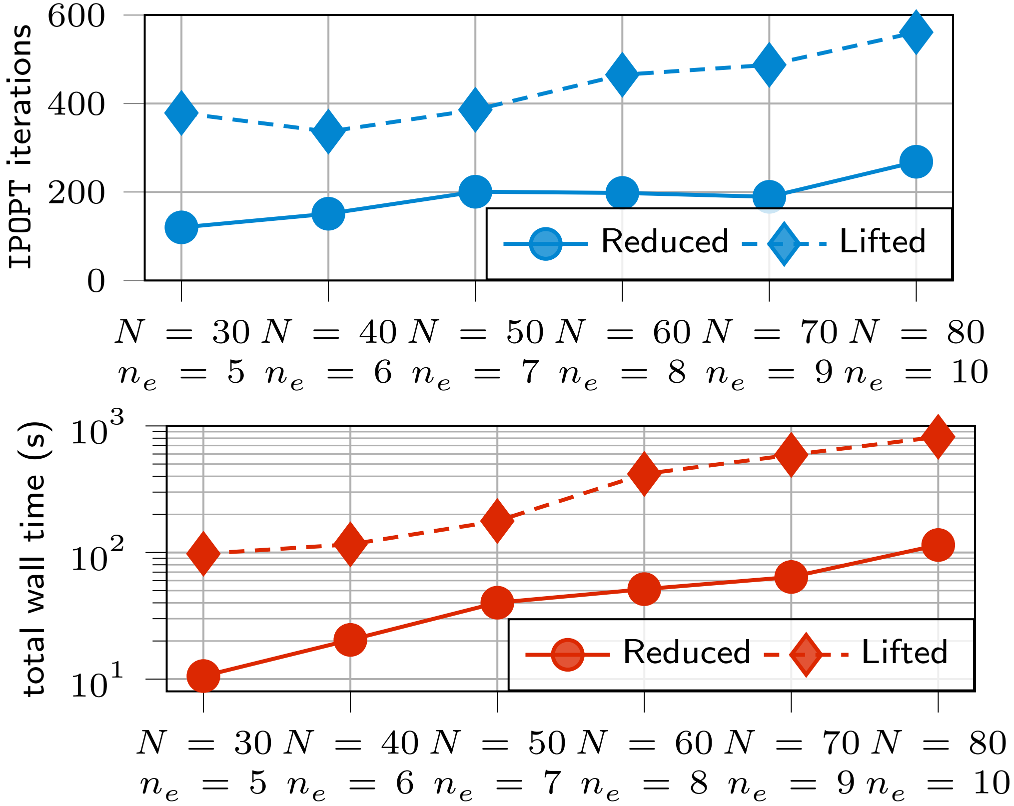
$$
\begin{aligned}
&\dot{q}_\mathrm{d} = Q(q_\mathrm{d})\nu_\mathrm{d},\\
&\text{For } j = 1,\dots,n_\mathrm{s}:\\
&\dot{q}_\mathrm{e}^{(j)} = Q(q_\mathrm{e}^{(j)})\nu_\mathrm{e}^{(j)},\\
&M\dot{\nu}_\mathrm{e}^{(j)} = u_j + \sum_{i=1}^{n_\mathrm{d}} Q(q_\mathrm{e}^{(j)})^\top n_{\tau,i}(q_\mathrm{e}^{(j)})\lambda_{\mathrm{n},i}^{(j)},\\
&\lambda_{\mathrm{n},i}^{(j)}\Phi_{\tau,i}(q_\mathrm{e}^{(j)}) \le \sigma,\ \ i = 1,\dots,n_\mathrm{d}\\
&0 \le \lambda_{\mathrm{n},i}^{(j)},\ \ 0 \le \Phi_{\tau,i}(q_\mathrm{e}^{(j)}),\ \ i = 1,\dots,n_\mathrm{d},\\
&u_j = D(\nu_\mathrm{d} - \nu_\mathrm{e}^{(j)}) + K((q_\mathrm{d} \oplus q_\mathrm{o}^{(j)}) \ominus q_\mathrm{e}^{(j)}),
\end{aligned}
$$

- Discretization through N_cnt intervals with N_sim simulation intervals per control interval
- Total simulation steps $N_\mathrm{tot} = N_\mathrm{cnt} N_\mathrm{sim}$
- On each simulation interval an implicit Euler time-stepping discretization is utilized
- On each control interval a constant $\nu_{\mathrm{d},k},\ \ k = 1,\dots,N_\mathrm{cnt}$ is used
- Cost function for terminal state $\bar{x} = (\bar{q}, \bar{\nu})$:

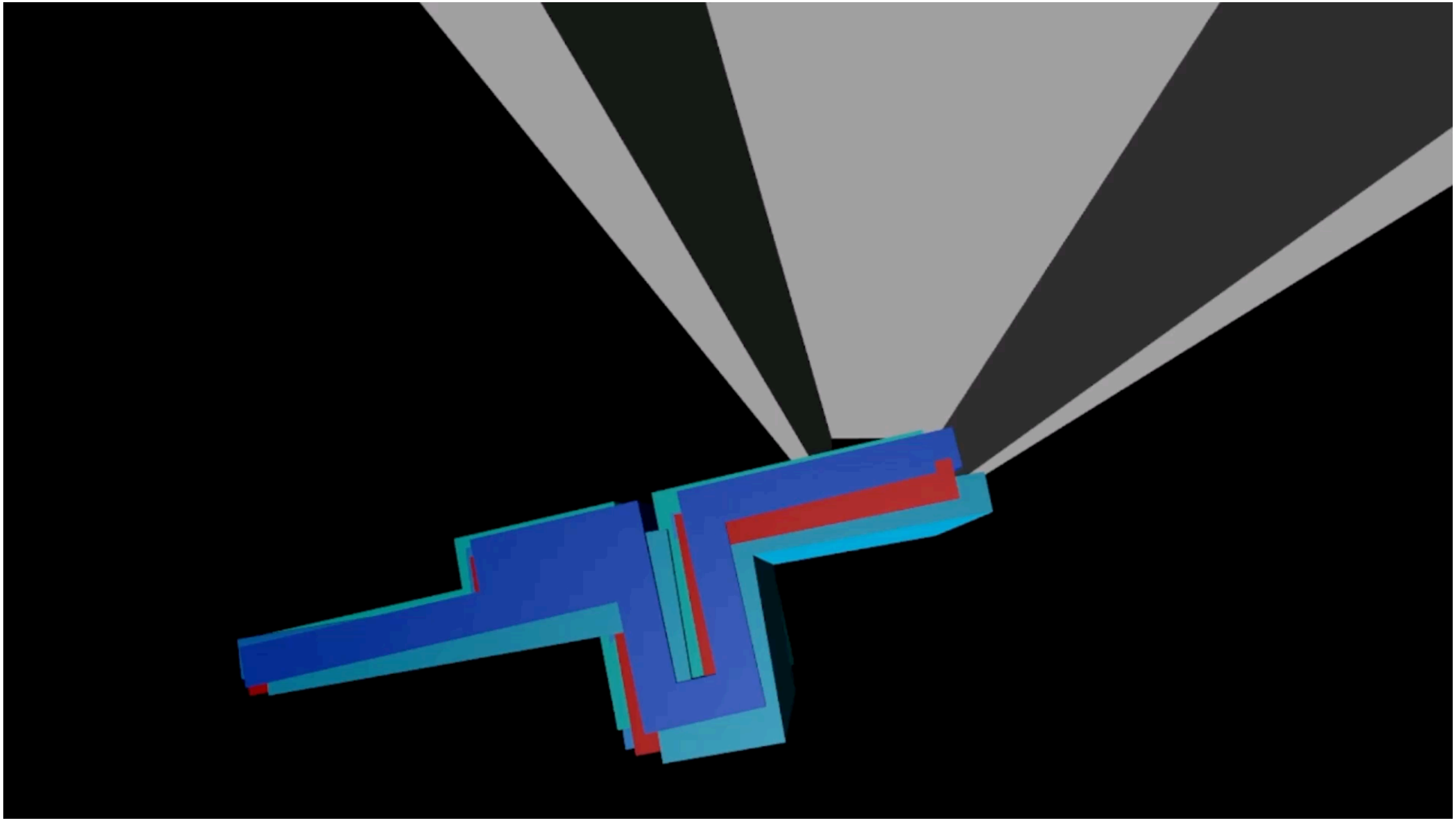
$$
\begin{aligned}
\mathrm{cost} = &\sum_{k=1}^{N_\mathrm{cnt}} 0.001\|\nu_{\mathrm{d},k,\mathrm{trs}}\|_2^2 + 0.01\|\nu_{\mathrm{d},k,\mathrm{ang}}\|_2^2\\
&+ 1\|\bar{\rho} - \rho_{\mathrm{d},N_\mathrm{tot}}\|_2^2 + 10(1 - (\bar{\xi}^\top \xi_{\mathrm{d},N_\mathrm{tot}})^2)\\
&+ \sum_{j=1}^{n_\mathrm{e}} 100\|\bar{\rho} - \rho_{\mathrm{e},N_\mathrm{tot}}^{(j)}\|_2^2 + 1000(1 - (\bar{\xi}^\top \xi_{\mathrm{e},N_\mathrm{tot}}^{(j)})^2)
\end{aligned}
$$

Computational performance comparison of reduced and lifted SDF implementations

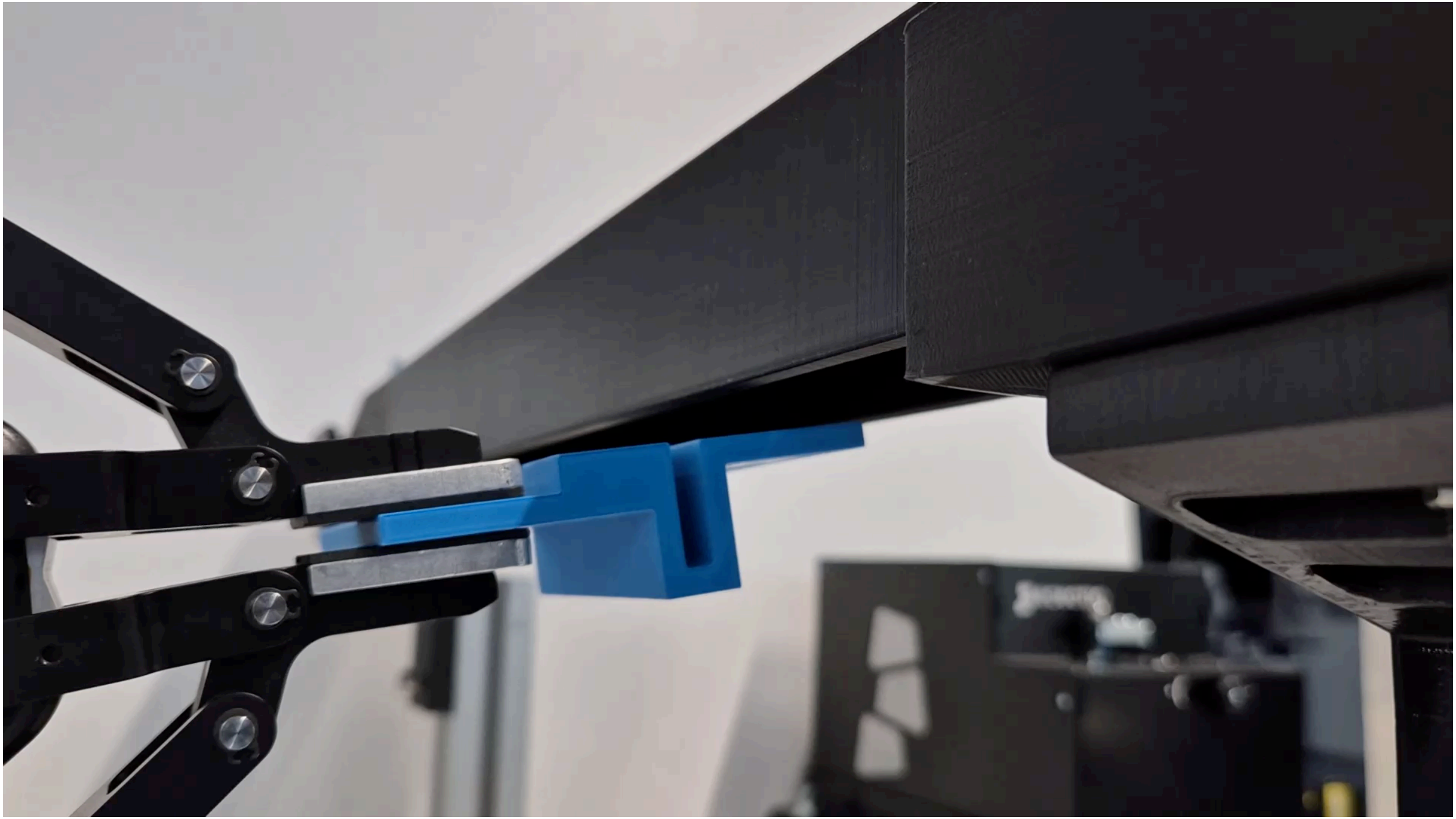
- The SDF $\Phi_{\tau,i}$ can be either evaluated as proposed by using HPIPM or by adding the perturbed KKT conditions directly in the optimal control problem (reduced or lifted implementation)
- We compare computational performance on a two-dimensional peg-in-hole problem for different trajectory lengths N and number of simultaneously simulated trajectories n_e
- Using the reduced modelling with external SDF evaluation results in less IPOPT iterations and less total wall time for all considered problem sizes



Robust Optimal Control Solution
(simulation)



Robust Optimal Control Solution (5 scenarios)
(experiment)



Conclusions

- Newton-type optimization can address seemingly combinatorial optimization problems in nonsmooth optimal control
- Mathematical Programs with Complementarity Constraints (MPCC) are a powerful tool for “disciplined nonsmooth programming”
- Derivatives remain a crucial optimization ingredient also when the nonconvexity of problems increases

Conclusions (2)

The great watershed in optimization isn’t between convexity and nonconvexity, but between **computer functions that do - or do not - provide derivatives.**