Numerical Optimal Control

Moritz Diehl

Systems Control and Optimization Laboratory, University of Freiburg, Germany

Summer School on Robust Model Predictive Control with CasADi, University of Freiburg
September 15-19, 2025

(slides jointly developed with Armin Nurkanović)

universität freiburg

Continuous-Time Optimal Control Problems (OCP)

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

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

(More general optimal control problems)

Many features left out here for simplicity of presentation:

- multiple dynamic stages
- differential algebraic equations (DAE) instead of ODE
- explicit time dependence
- constant design parameters
- multipoint constraints $r(x(t_0), x(t_1), \ldots, x(t_{\text{end}})) = 0$

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

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

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

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

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

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

Three levels of difficulty:

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

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

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

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

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

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

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$ ($\rightarrow$ convex optimization)

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

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

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

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$ $\quad$ (→ convex optimization)

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

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

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

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

Three levels of difficulty:

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

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

(c) Nonsmooth and Mixed-Integer Dynamics

Three Levels of Difficulty in Continuous-Time OCP

Continuous-Time OCP

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

Three levels of difficulty:

(a) Linear ODE: $f(x,u) = Ax + Bu$ ($\rightarrow$ convex optimization)

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

(c) Nonsmooth and Mixed-Integer Dynamics

In this school, we focus on 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)

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

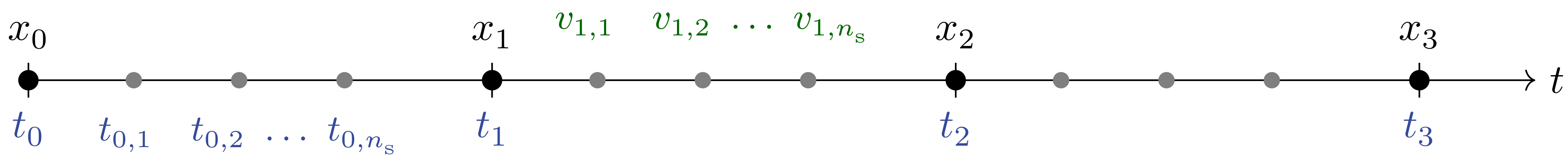
Discretization: N Runge-Kutta steps of each n_s stages

$$x_{0,0} = \bar{x}_0, \qquad \Delta t = \frac{T}{N}$$

$$\begin{aligned} v_{k,j} &= f(x_{k,j}, u_k) \\ x_{k,j} &= x_{k,0} + \Delta t \textstyle\sum_{n=1}^{n_s} a_{jn} v_{k,n} \\ x_{k+1,0} &= x_{k,0} + \Delta t \textstyle\sum_{n=1}^{n_s} b_n v_{k,n} \end{aligned}$$

$$j = 1, \ldots, n_s, \quad k = 0, \ldots, N-1$$

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

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

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

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

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

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

- Direct methods "first discretize, then optimize"

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

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

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

- 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

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

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

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

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

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"

Discrete time OCP (an NLP)

$$\min_{\mathbf{x},\mathbf{z},\mathbf{u}} \sum_{k=0}^{N-1} L_\mathrm{d}(x_k,z_k,u_k) + E(x_N)$$

$$\begin{aligned} \text{s.t.}\quad x_0 &= \bar{x}_0 \\ x_{n+1} &= \phi_f(x_n,z_n,u_n) \\ 0 &= \phi_\mathrm{int}(x_n,z_n,u_n) \\ 0 &\geq \phi_h(x_n,z_n,u_n),\ n=0,\dots,N-1 \\ 0 &\geq r(x_N) \end{aligned}$$

Variables $\mathbf{x}=(x_0,\dots,x_N)$, $\mathbf{z}=(z_0,\dots,z_N)$ and $\mathbf{u}=(u_0,\dots,u_{N-1})$.
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)

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

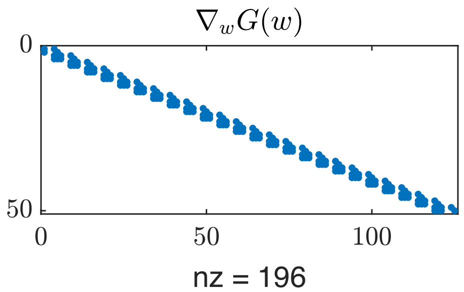
Variables $w = (\mathbf{x}, \mathbf{z}, \mathbf{u})$

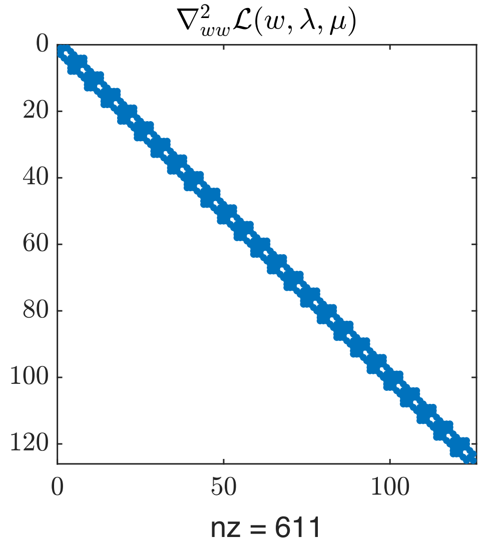
Nonlinear Program (NLP)

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

Large and sparse NLP

Sparse NLP resulting from direct transcription





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

Nonlinear Program (NLP)

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

Large and sparse NLP

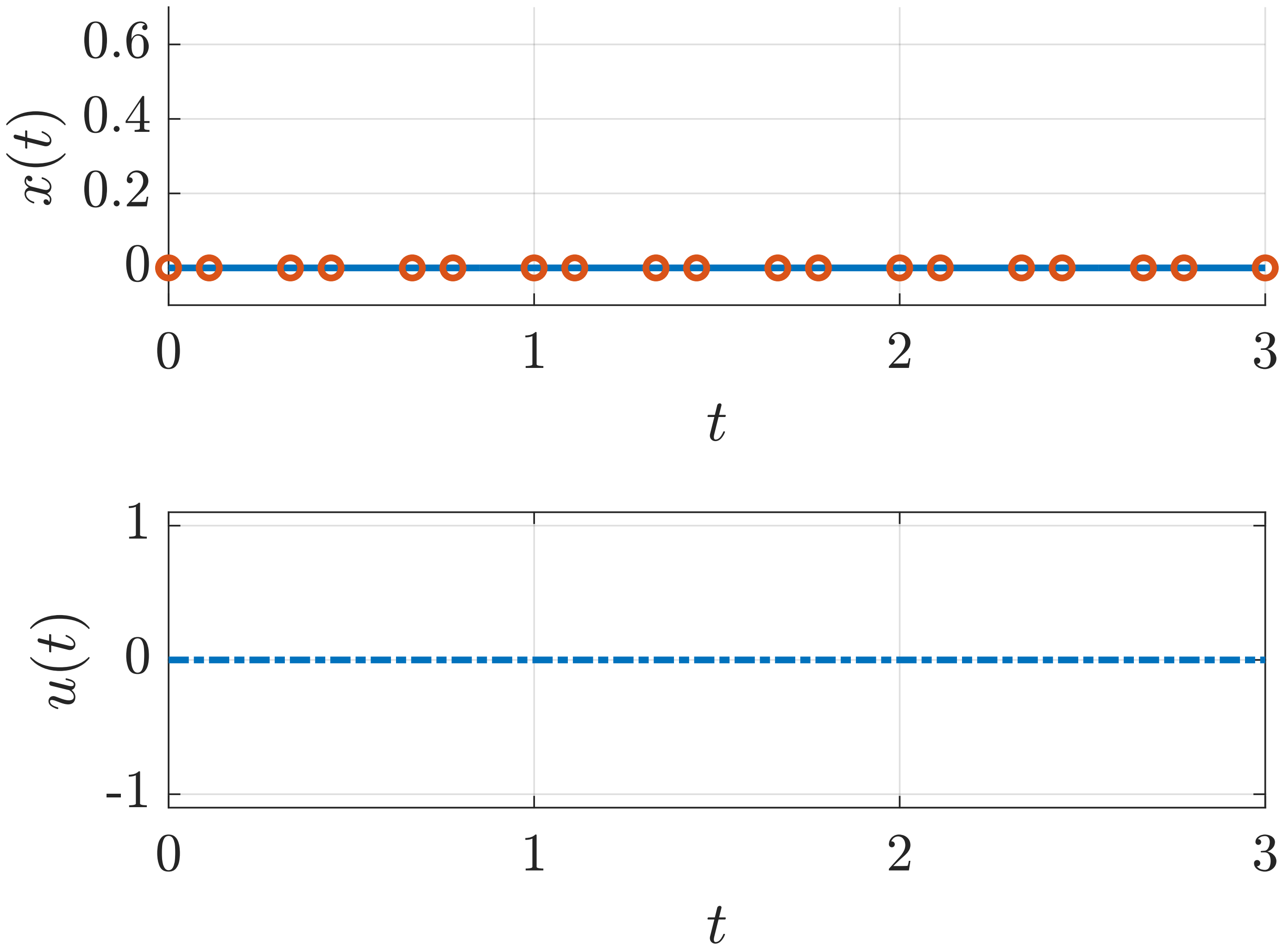
Illustrative example of direct collocation with Newton-type optimization

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

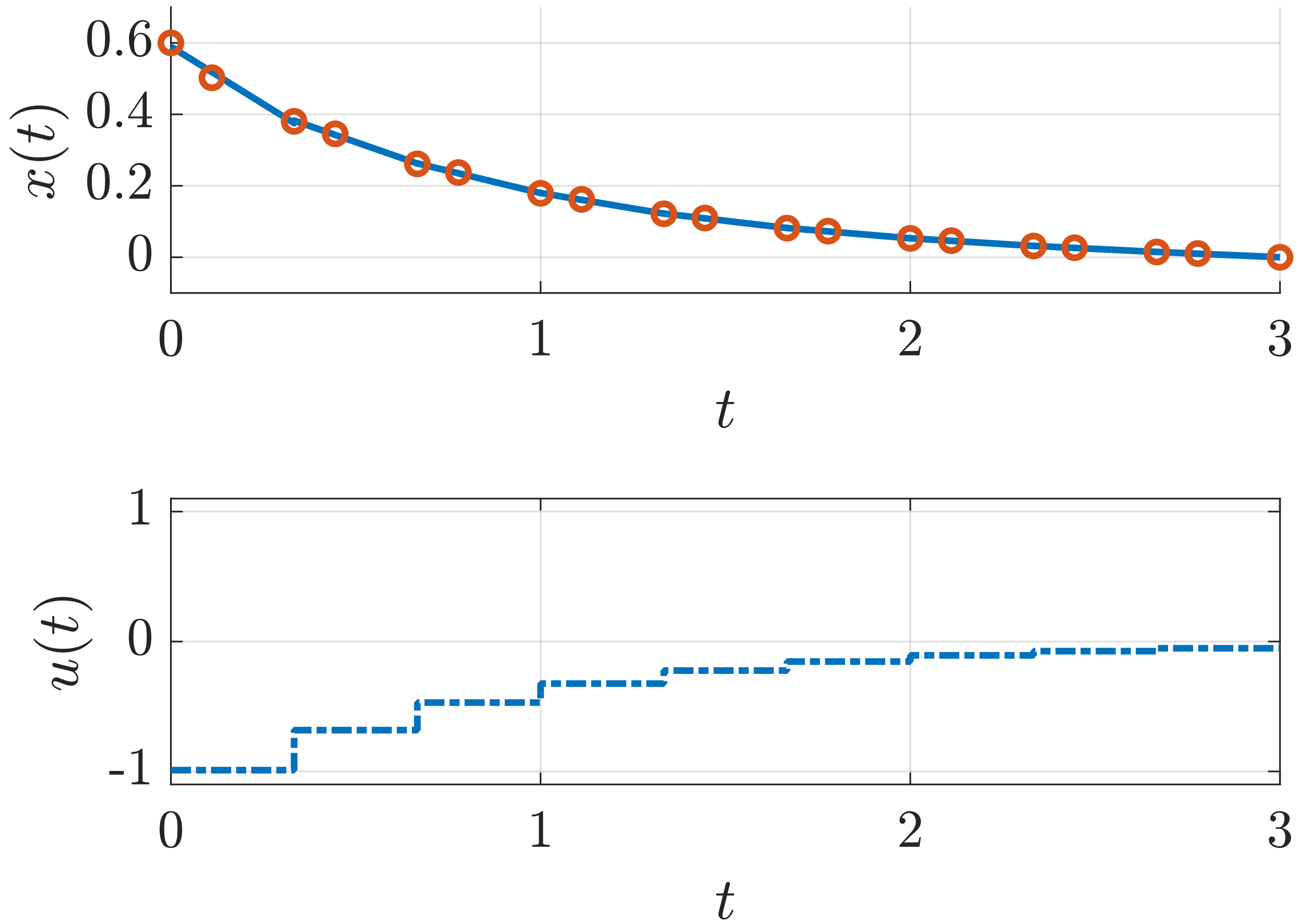
$$\begin{aligned}
\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_0^3 x(t)^2 + u(t)^2 \, dt \\
\text{subject to} \quad & \\
x(0) &= \bar{x}_0 && \text{(initial value, } \bar{x}_0 = 0.6\text{)} \\
\dot{x} &= (1+x)x + u, && \text{(ODE model)} \\
-1 \leq u(t) &\leq 1, \qquad t \in [0,3] && \text{(bounds)} \\
x(3) &= 0 && \text{(terminal constraint)}
\end{aligned}$$

- 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)

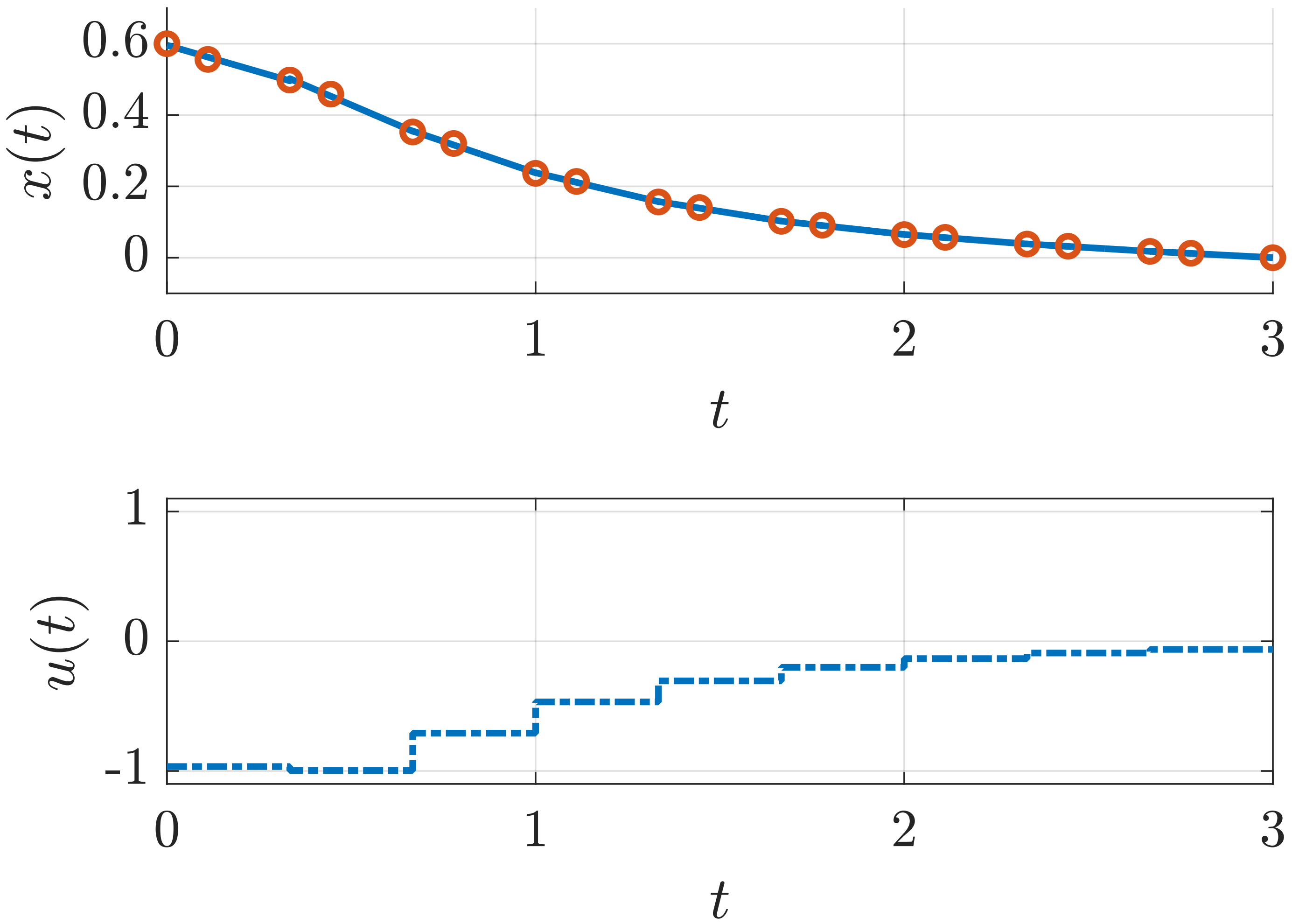
Illustrative example: Initialization



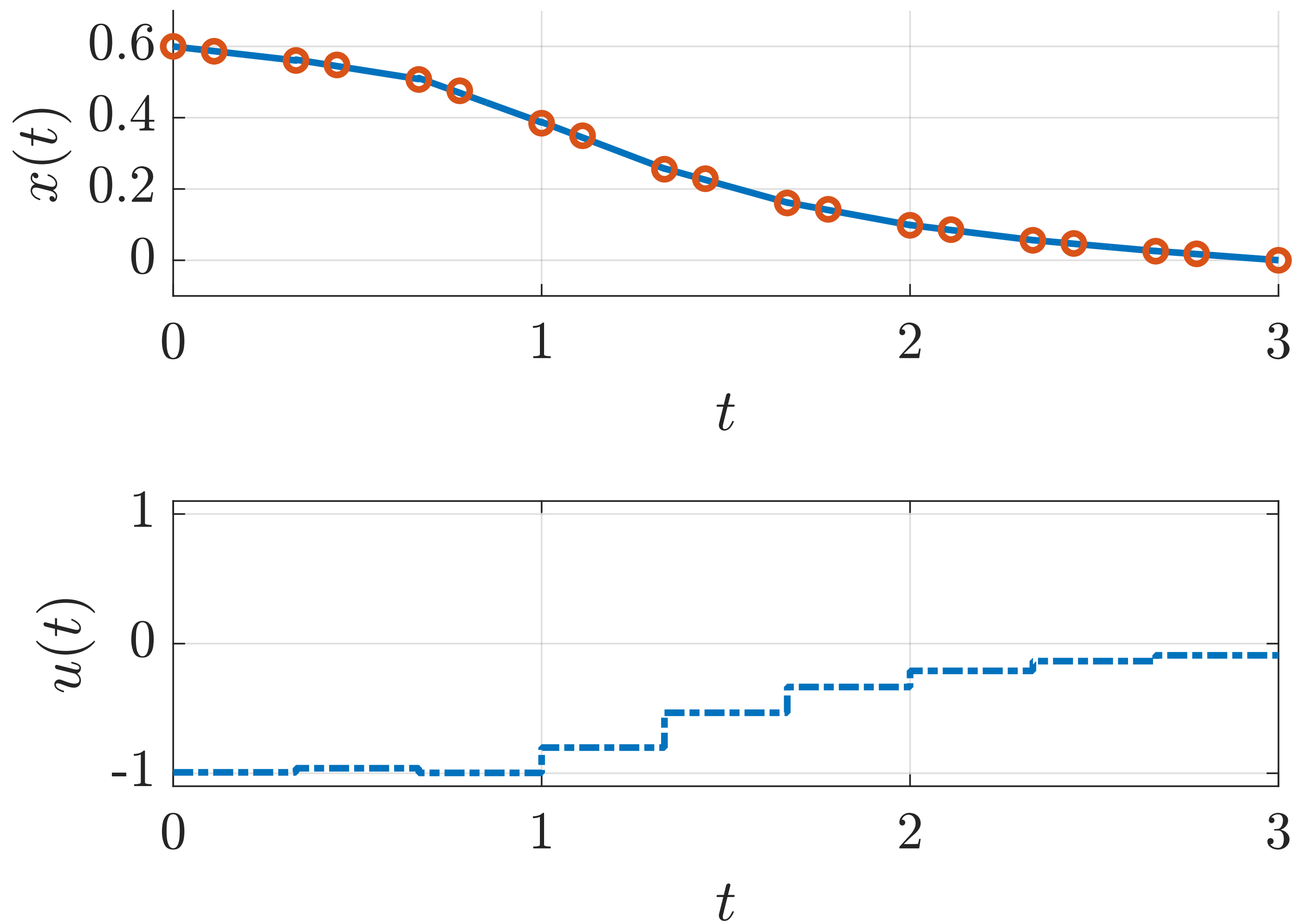
Illustrative example: First Iterate



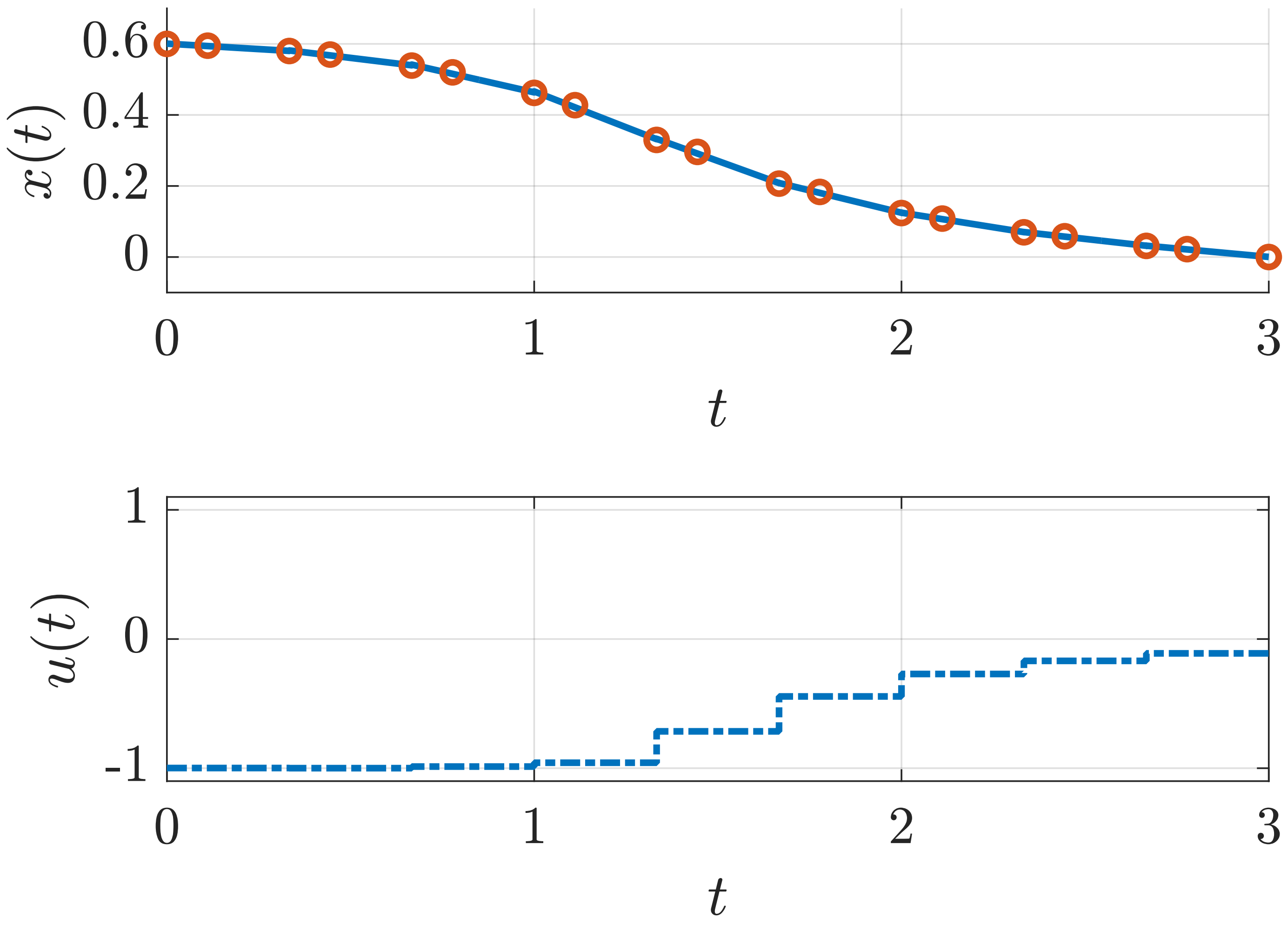
Illustrative example: Second Iterate



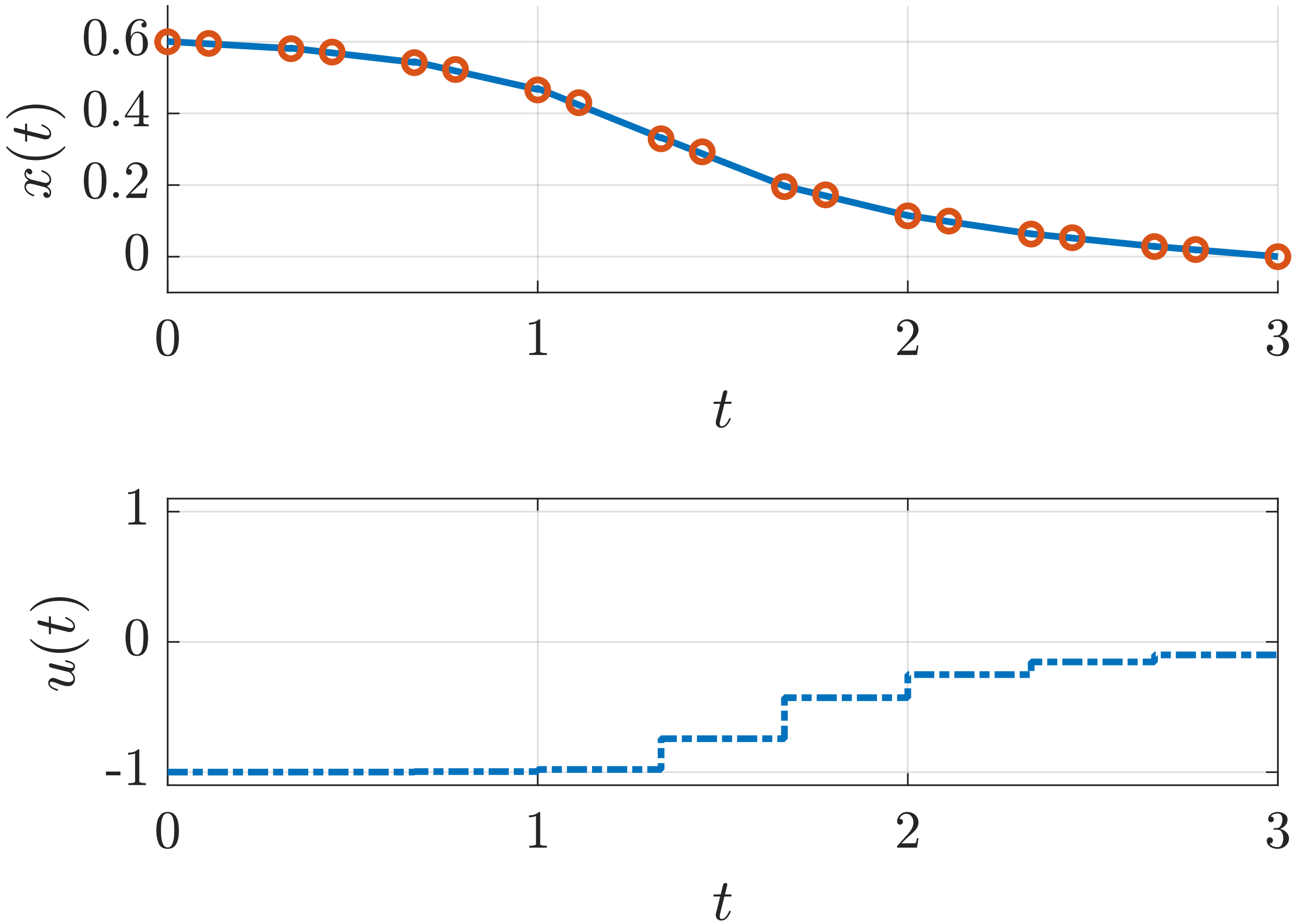
Illustrative example: Third Iterate



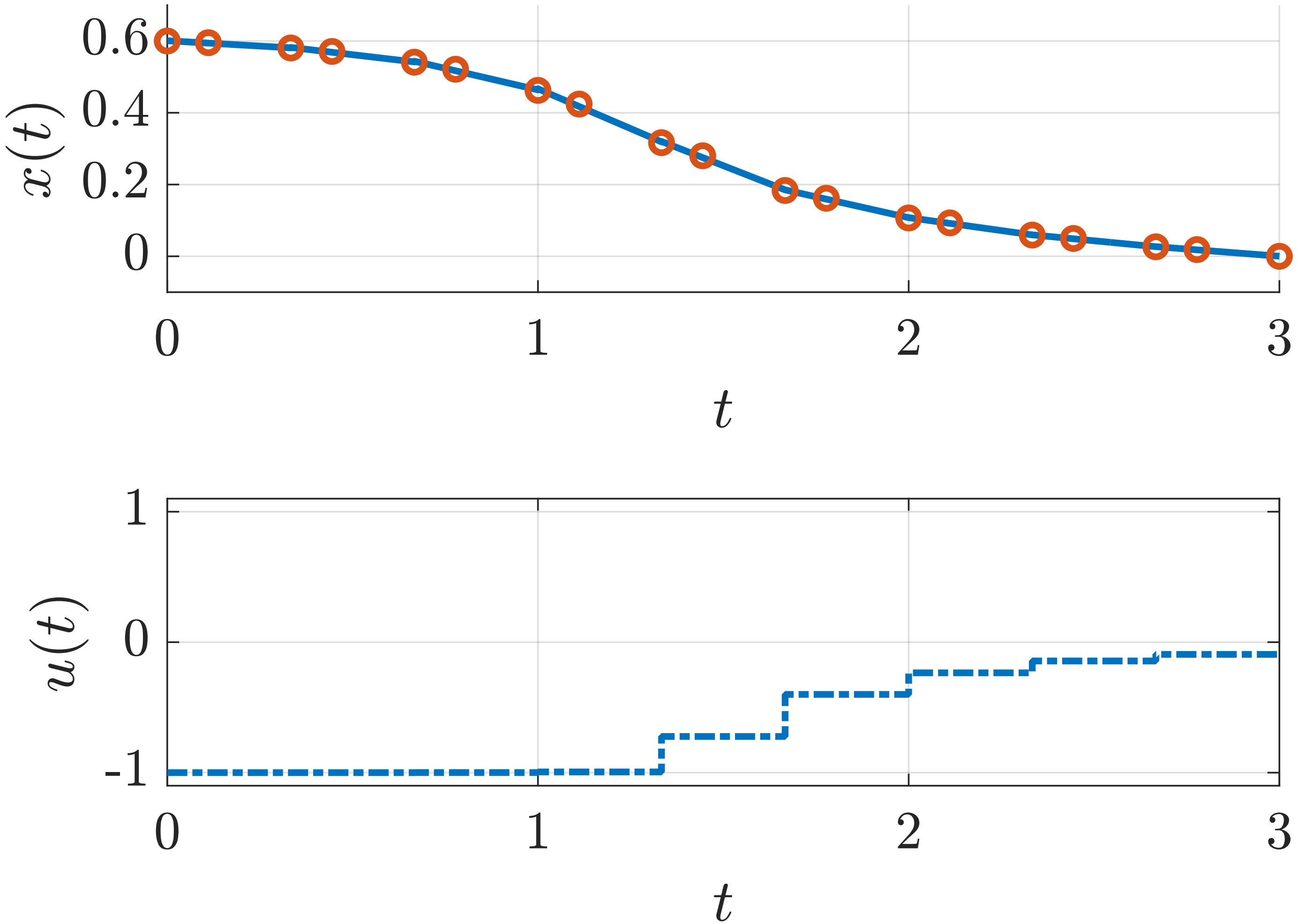
Illustrative example: Fourth Iterate



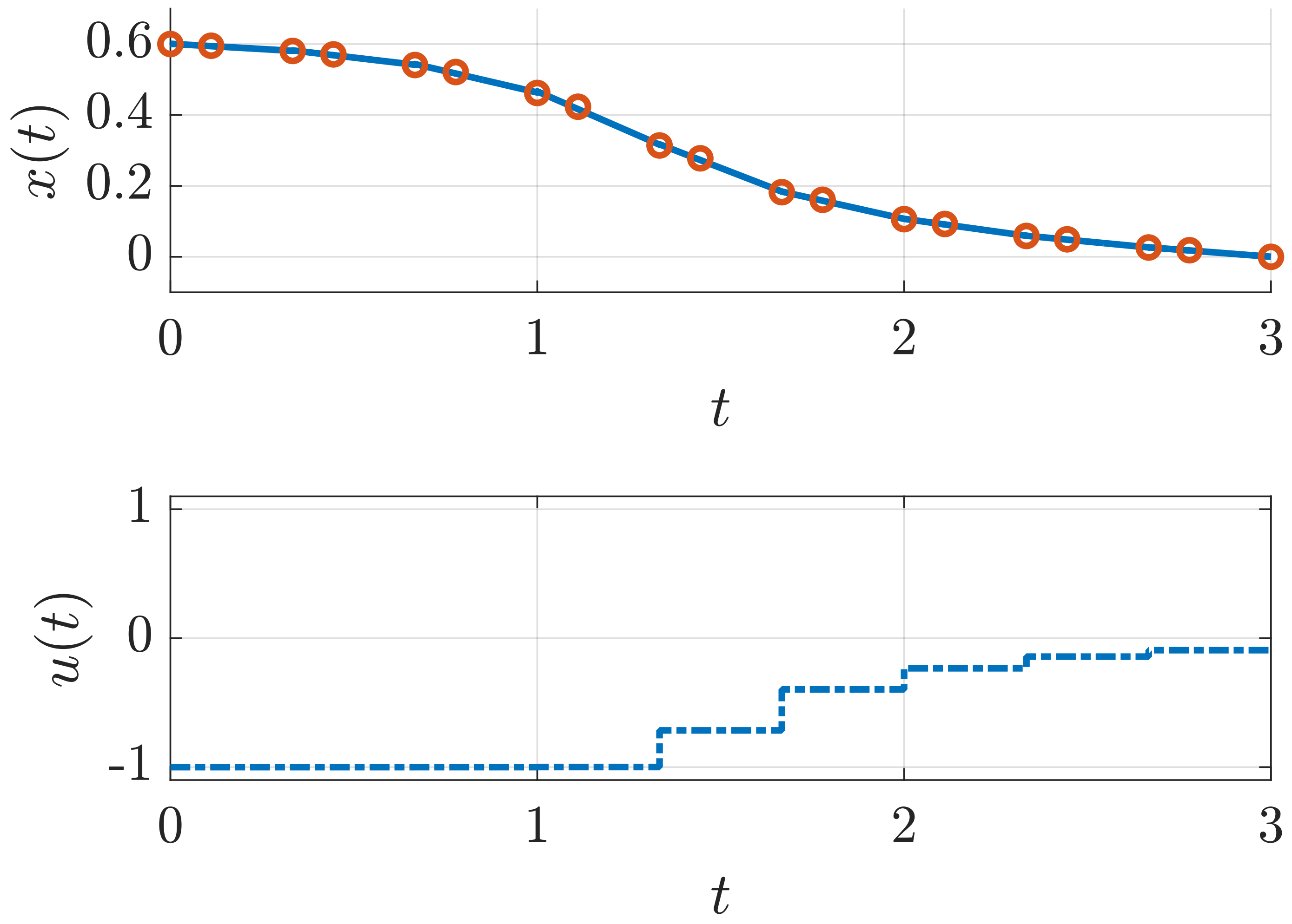
Illustrative example: Fifth Iterate



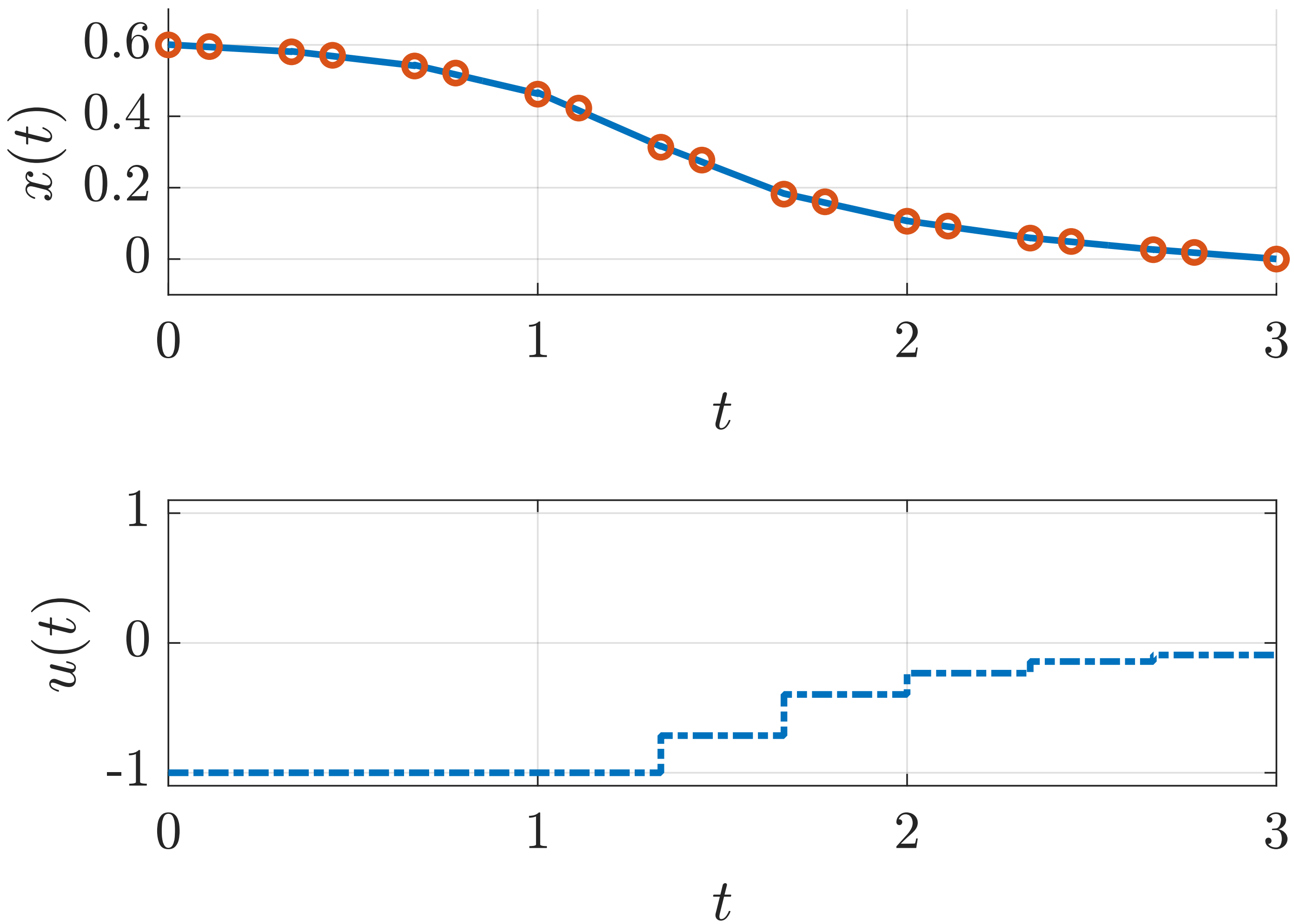
Illustrative example: Sixth Iterate



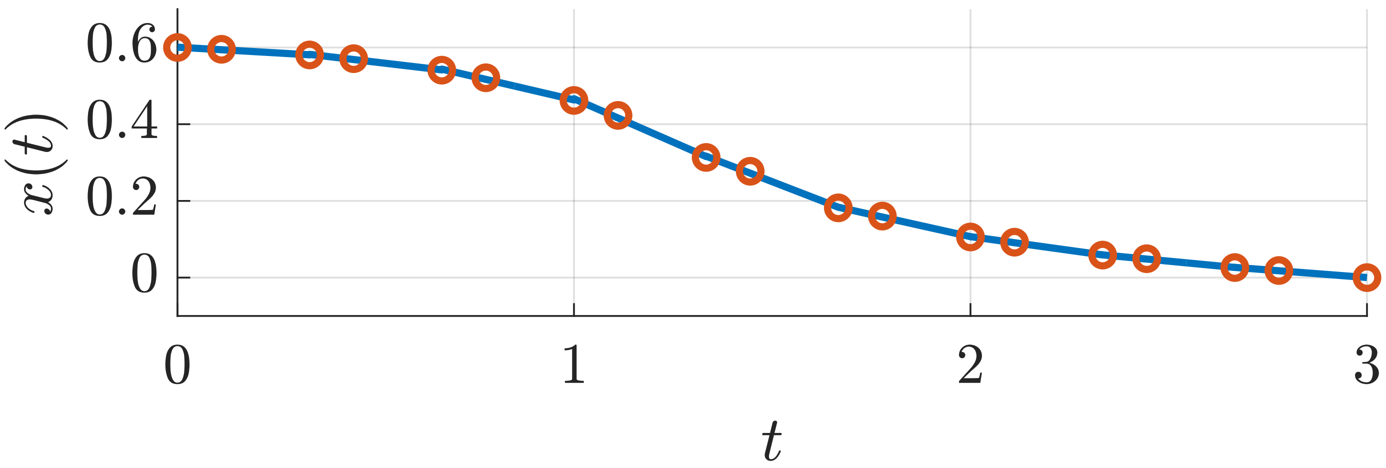
Illustrative example: Seventh Iterate

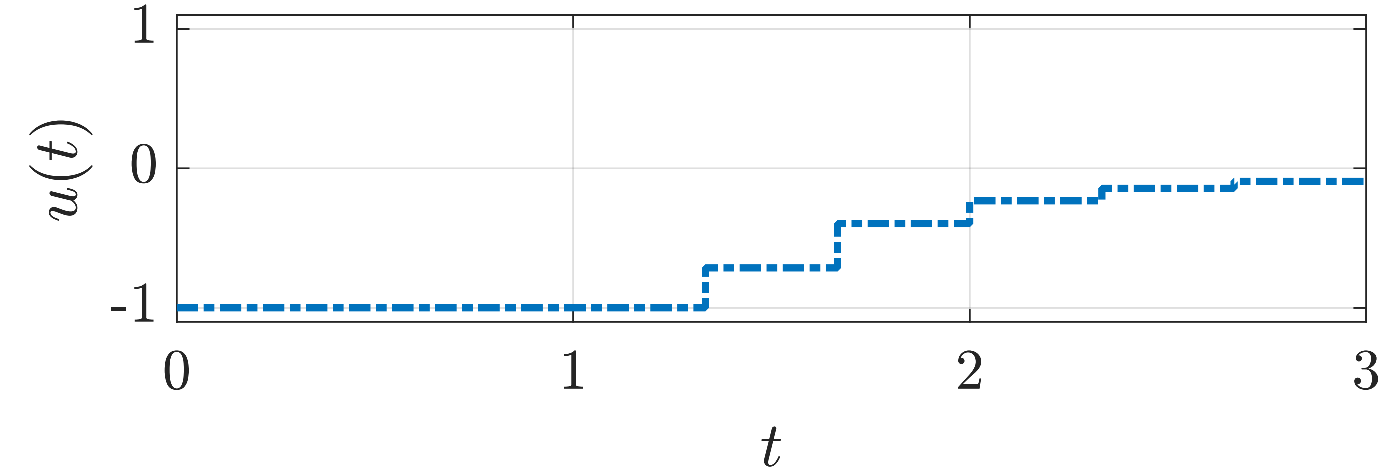


Illustrative example: Eighth Iterate



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:

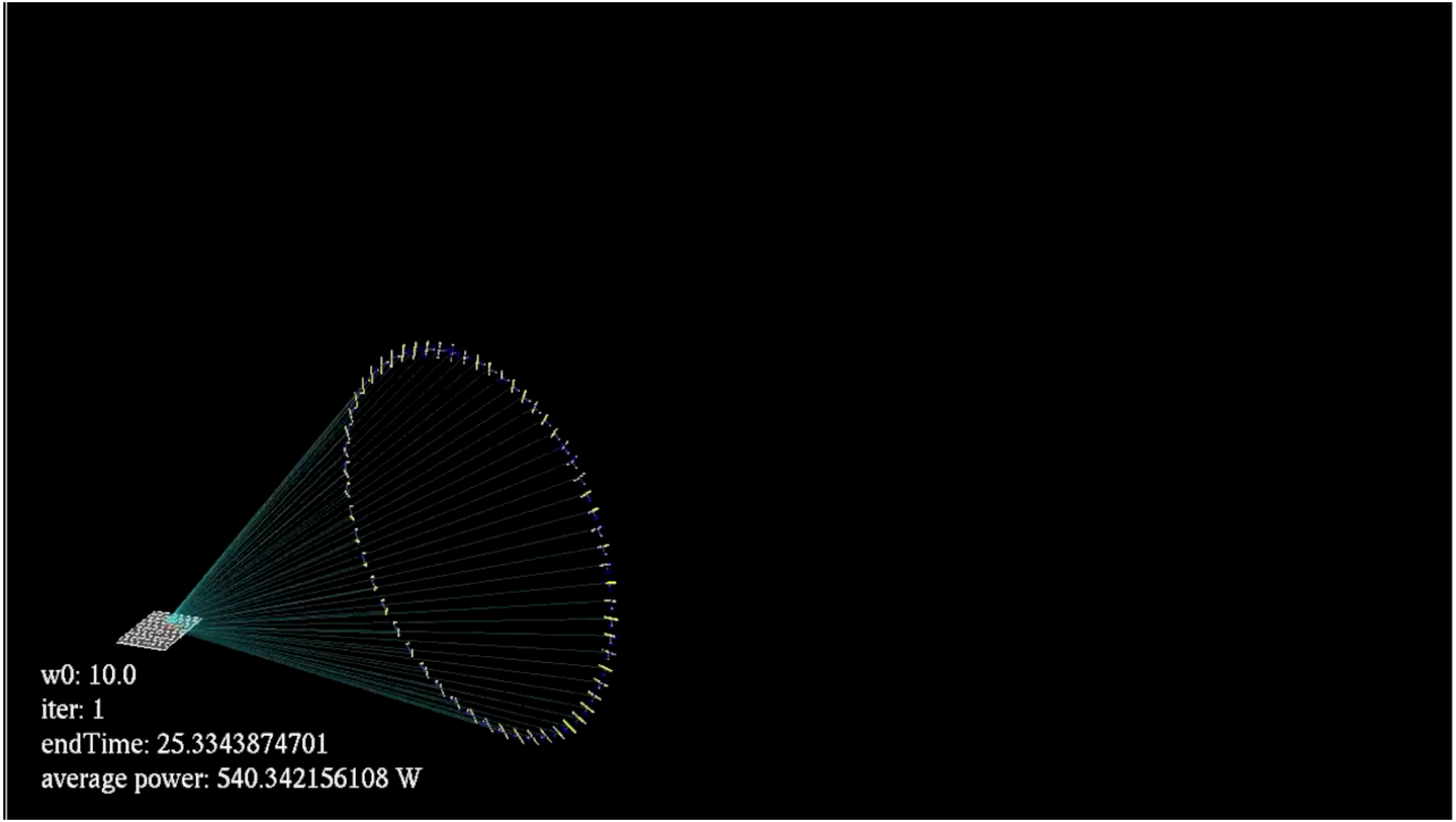
$$\vec{v} = \begin{bmatrix} \dot{x} - \dot{\delta} y \\ \dot{y} + \dot{\delta}(r_\mathrm{A} + x) \\ \dot{z} \end{bmatrix} - \vec{w}(x, y, z, \delta, t), \qquad \alpha = -\frac{\vec{E}_z^T \vec{v}}{\vec{E}_x^T \vec{v}}, \qquad \beta = \frac{\vec{E}_y^T \vec{v}}{\vec{E}_x^T \vec{v}}$$

Aero. forces/torques:

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

Newton-Type Optimization Iterations for Power Optimal Flight

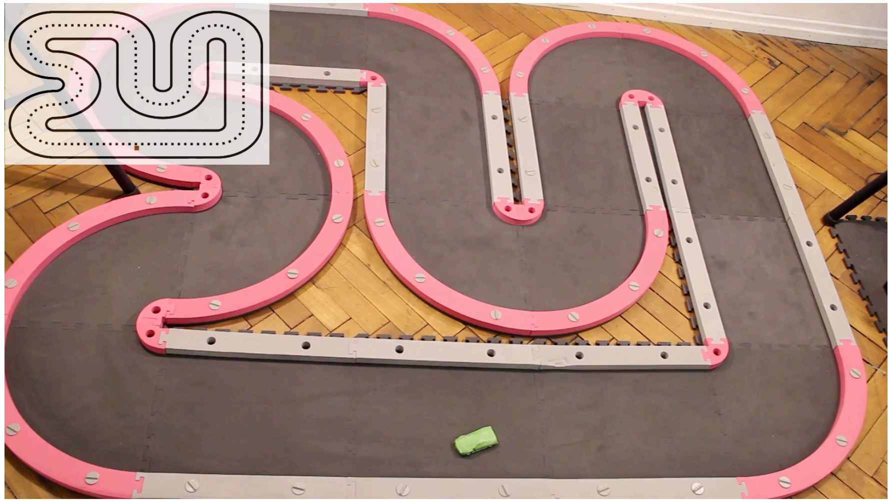
(video by Greg Horn, using CasADi and Ipopt as optimization engine)



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

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

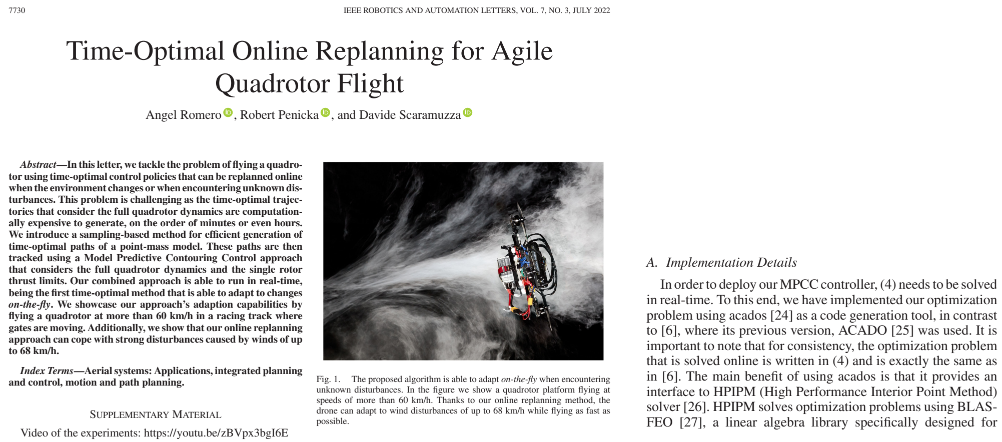
Example 1: Autonomous Driving (in Freiburg)



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

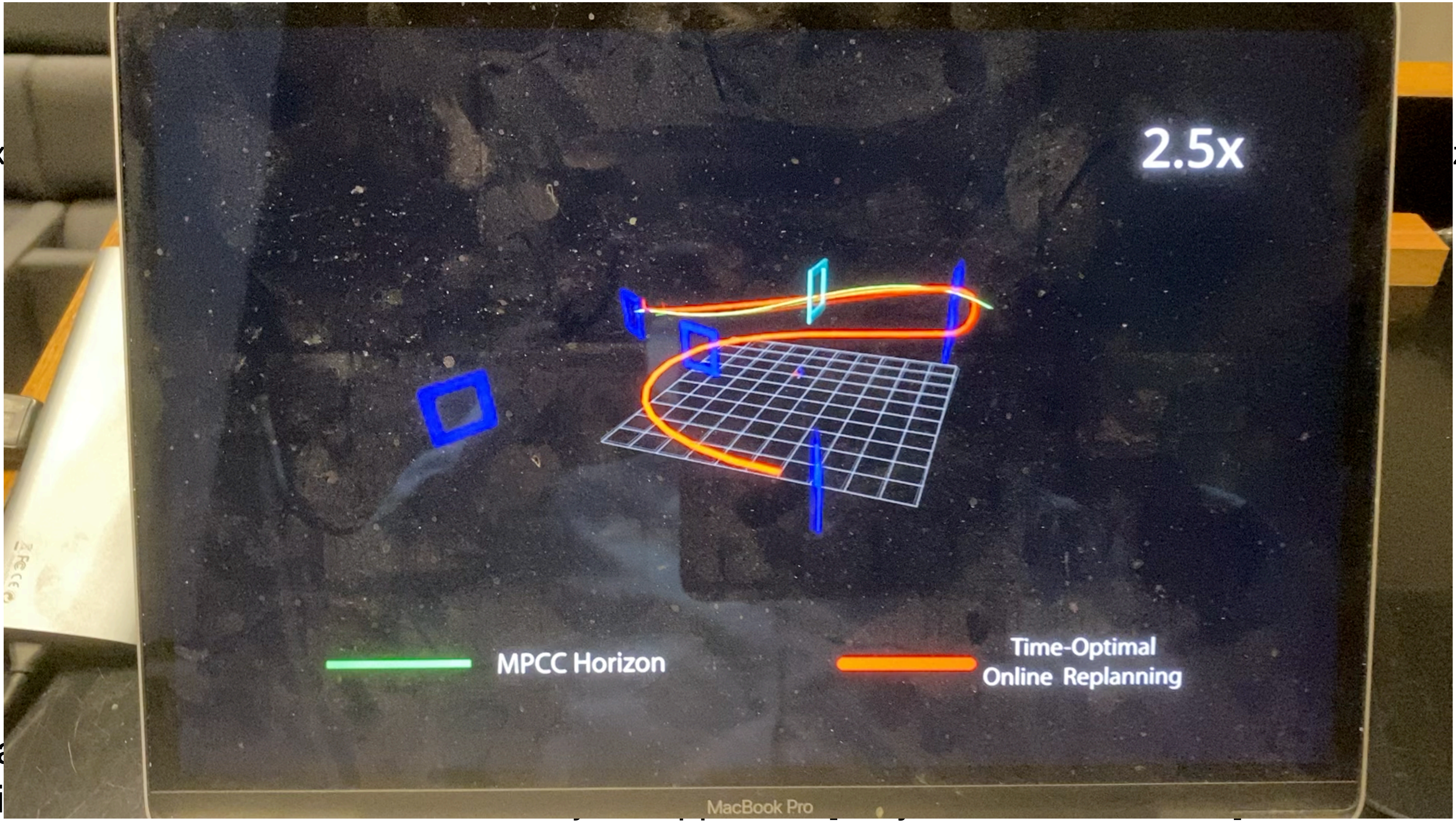
In order to deploy our MPCC controller, (4) needs to be solved in real-time. To this end, we have implemented our optimization problem using acados [24] as a code generation tool, in contrast to [6], where its previous version, ACADO [25] was used. It is important to note that for consistency, the optimization 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)

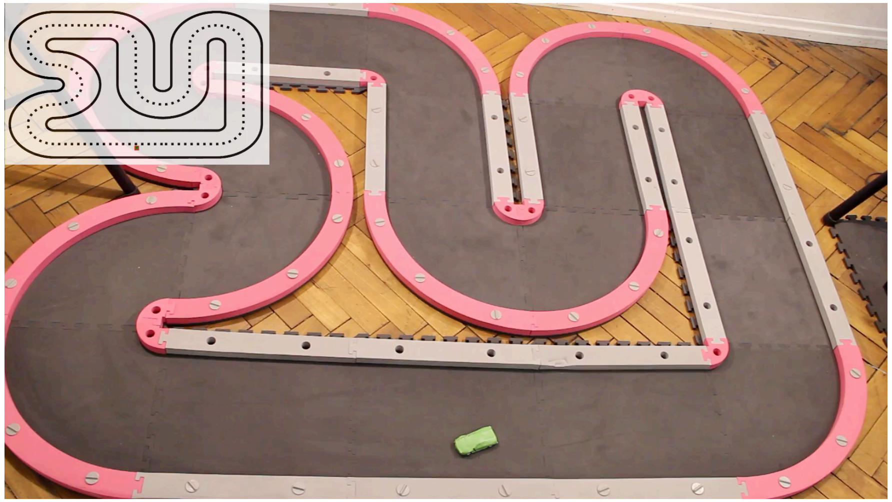
La

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.]

Direct Methods

- "first discretize, then optimize"
- transcribe infinite problem into finite **Nonlinear Programming Problem (NLP)**
- Pros and Cons:
  - \+ can use state-of-the-art methods for NLP solution
  - \+ can treat inequality constraints and multipoint constraints much easier
  - \- obtains only suboptimal / approximate solution
- nowadays most commonly used methods due to their easy applicability and robustness

Classification of Direct Optimal Control Methods

Direct methods transform continuous time problem into a nonlinear program (NLP):

- Direct Transcription: all internal integrator variables are kept exposed as NLP variables. Special cases: direct collocation and pseudospectral methods. (called "simultaneous approach", as simulation and optimization are tackled simultaneously by NLP solver)
- Direct Multiple Shooting: for every control interval, all internal integration steps are hidden to the NLP. Integration routine is complicated but differentiable function (also called "simultaneous approach")
- Direct Single Shooting: all state variables are eliminated by forward simulation, only the control parameters are kept as NLP variables. NLP objective and constraints are very long functions. (called "sequential approach", as simulation and optimization proceed sequentially)
- Flatness-based optimal control: in "flat" systems, the states and control inputs can be obtained from derivatives of a "flat output". One can then parameterize the flat output as superposition of smooth basis functions, and formulate an NLP in the space of the basis coefficients. Similar in performance to simultaneous approaches but limited to flat systems.

Classification of Direct Optimal Control Methods

Direct methods transform continuous time problem into a nonlinear program (NLP):

- Direct Transcription: all internal integrator variables are kept exposed as NLP variables. Special cases: direct collocation and pseudospectral methods. (called "simultaneous approach", as simulation and optimization are tackled simultaneously by NLP solver)
- Direct Multiple Shooting: for every control interval, all internal integration steps are hidden to the NLP. Integration routine is complicated but differentiable function (also called "simultaneous approach")
- Direct Single Shooting: all state variables are eliminated by forward simulation, only the control parameters are kept as NLP variables. NLP objective and constraints are very long functions. (called "sequential approach", as simulation and optimization proceed sequentially)
- Flatness-based optimal control: in "flat" systems, the states and control inputs can be obtained from derivatives of a "flat output". One can then parameterize the flat output as superposition of smooth basis functions, and formulate an NLP in the space of the basis coefficients. Similar in performance to simultaneous approaches but limited to flat systems.

Classification of Direct Optimal Control Methods

Direct methods transform continuous time problem into a nonlinear program (NLP):

- Direct Transcription: all internal integrator variables are kept exposed as NLP variables. Special cases: direct collocation and pseudospectral methods. (called "simultaneous approach", as simulation and optimization are tackled simultaneously by NLP solver)
- Direct Multiple Shooting: for every control interval, all internal integration steps are hidden to the NLP. Integration routine is complicated but differentiable function (also called "simultaneous approach")
- Direct Single Shooting: all state variables are eliminated by forward simulation, only the control parameters are kept as NLP variables. NLP objective and constraints are very long functions. (called "sequential approach", as simulation and optimization proceed sequentially)
- Flatness-based optimal control: in "flat" systems, the states and control inputs can be obtained from derivatives of a "flat output". One can then parameterize the flat output as superposition of smooth basis functions, and formulate an NLP in the space of the basis coefficients. Similar in performance to simultaneous approaches but limited to flat systems.

Direct Methods: Comparison of Sequential and Simultaneous Approach

We compare two direct methods:

- Direct Single Shooting (sequential simulation and optimization)
- Direct Multiple Shooting (simultaneous simulation and optimization)

Now change to Part II of this talk