The acados software

An Introduction & Research Spotlights

Jonathan Frey

Systems Control and Optimization Laboratory (syscop)

September 18, 2025

universität freiburg

Me: Jonathan Frey

🧑‍💻 PhD student in Freiburg with Moritz Diehl

Me: Jonathan Frey

- 🧑‍💻 PhD student in Freiburg with Moritz Diehl
- 🎓 Studied mathematics
  - Bachelor: TU Ilmeanu
  - Master: Uni Freiburg

universität freiburg

Me: Jonathan Frey

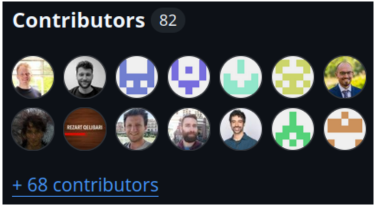
- PhD student in Freiburg with Moritz Diehl
- Studied mathematics
  - Bachelor: TU Ilmeanu
  - Master: Uni Freiburg
- Research
  - Fast MPC implementations acados
  - MPC + RL via `leap-c`
  - Advanced OCP discretizations

Talk structure

1. Overview on acados
2. Differentiable Nonlinear Model Predictive Control

acados – fast embedded solvers for nonlinear optimal control

An open-source software package mainly developed in Freiburg, Germany

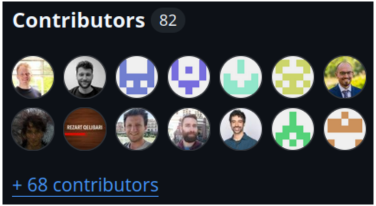


Efficiency, usability, modularity, state-of-the-art optimization algorithms

- ⚡ Written in C using high-performance linear algebra provided by BLASFEO
- ⚡ Fully exploits sparsity of optimal control structured NLPs

acados – fast embedded solvers for nonlinear optimal control

An open-source software package mainly developed in Freiburg, Germany



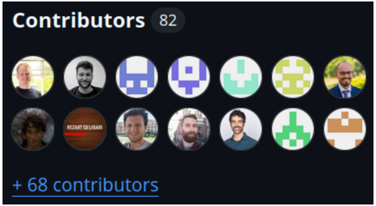
Efficiency, usability, modularity, state-of-the-art optimization algorithms

- Written in `C` using high-performance linear algebra provided by `BLASFEO`
- Fully exploits sparsity of optimal control structured NLPs
- Interfaces to `Python`, MATLAB, `Simulink`
- nonlinear & symbolic models via `CasADi`
- Flexible problem formulation: multi-phase & MHE

- Minimal dependencies $\Longrightarrow$ embeddable

acados – fast embedded solvers for nonlinear optimal control

An open-source software package mainly developed in Freiburg, Germany



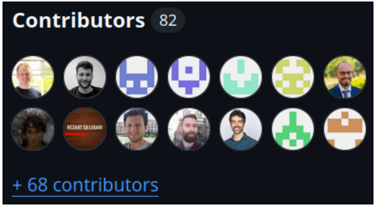
Efficiency, usability, modularity, state-of-the-art optimization algorithms

- ⚡ Written in `C` using high-performance linear algebra provided by `BLASFEO`
- ⚡ Fully exploits sparsity of optimal control structured NLPs
- 🕹️ Interfaces to `Python`, MATLAB, `Simulink`
- nonlinear & symbolic models via `CasADi`
- 🤸 Flexible problem formulation: multi-phase & MHE

- 🔌 Minimal dependencies $\Longrightarrow$ embeddable
- 🔄 Integrators for ODE & DAE: ERK & IRK, efficient sensitivity propagation
- 🧮 QP solvers: full & partial condensing via `HPIPM`
  `HPIPM`, `DAQP`, `qpOASES`, `qpDUNES`, `OSQP`

acados – fast embedded solvers for nonlinear optimal control

An open-source software package mainly developed in Freiburg, Germany

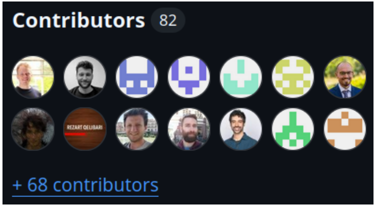


Efficiency, usability, modularity, state-of-the-art optimization algorithms

- Written in C using high-performance linear algebra provided by BLASFEO
- Fully exploits sparsity of optimal control structured NLPs
- Interfaces to Python, MATLAB, Simulink
- nonlinear & symbolic models via CasADi
- Flexible problem formulation: multi-phase & MHE
- Minimal dependencies $\Longrightarrow$ embeddable
- Integrators for ODE & DAE: ERK & IRK, efficient sensitivity propagation
- QP solvers: full & partial condensing via HPIPM HPIPM, DAQP, qpOASES, qpDUNES, OSQP
- NLP solvers: SQP, DDP, RTI, AS-RTI
- Robust & stochastic MPC via zoRO
- Exploit convex-over-nonlinear structures

acados – fast embedded solvers for nonlinear optimal control

An open-source software package mainly developed in Freiburg, Germany

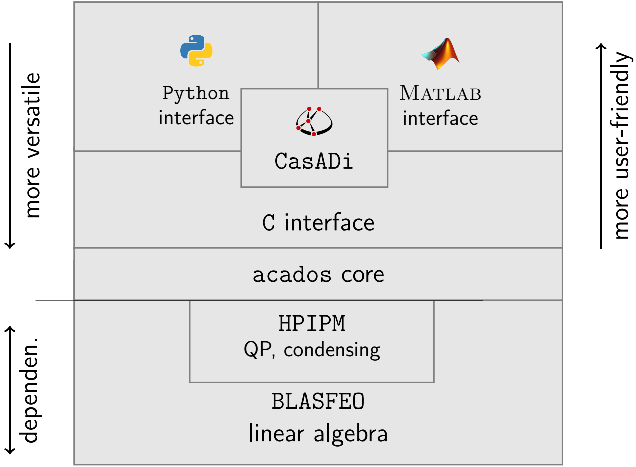


Efficiency, usability, modularity, state-of-the-art optimization algorithms

- ⚡ Written in C using high-performance linear algebra provided by `BLASFEO`
- ⚡ Fully exploits sparsity of optimal control structured NLPs
- 🕹️ Interfaces to `Python`, MATLAB, `Simulink`
- nonlinear & symbolic models via `CasADi`
- 🤸 Flexible problem formulation: multi-phase & MHE
- 🔌 Minimal dependencies $\Longrightarrow$ embeddable
- 🔄 Integrators for ODE & DAE: ERK & IRK, efficient sensitivity propagation
- 🧮 QP solvers: full & partial condensing via `HPIPM` `HPIPM`, `DAQP`, `qpOASES`, `qpDUNES`, `OSQP`
- 🎯 NLP solvers: SQP, DDP, RTI, AS-RTI
- 🦺 Robust & stochastic MPC via zoRO
- 🧠 Exploit convex-over-nonlinear structures

⭐ `github.com/acados/acados` 📚 `docs.acados.org` 🤝 `discourse.acados.org`

Structure of the acados software



The interplay between the acados dependencies, the 'core' C library and its interfaces.

- BLASFEO: Basic Linear Algebra for Embedded Optimization (Frison et al., 2018)
- HPIPM: High-Performance Interior Point Method (Frison & Diehl, 2020)

Intro

- Real-world control applications
  - fast dynamics,
  - nonlinear optimal control problem formulations,
  - strict hardware limitations

  require tailored high-performance algorithms.
- acados implements such algorithms
- Application projects include
  - Wind turbines
  - Drones
  - Race cars
  - Driving assistance systems
  - Electric drives
  - Vessels
  - ...

Intro – Model Predictive Control

Continuous-time optimal control problem (OCP):

$$
\begin{aligned}
\underset{x(\cdot),z(\cdot),u(\cdot)}{\text{minimize}} \quad & \int_{t=0}^{T} \ell(x(t), z(t), u(t))\,\mathrm{d}t + M(x(T)) \\
\text{subject to} \quad & x(0) = \bar{x}_0, \\
& 0 = f(\dot{x}(t), x(t), z(t), u(t)), \quad t \in [0, T], \\
& 0 \geq g(x(t), z(t), u(t)), \quad t \in [0, T].
\end{aligned} \tag{1}
$$

- State x, control u, (algebraic variables z)
- Cost l, M
- Dynamics f
- Constraints g

In MPC, instances of these problems are solved repeatedly, with current state $\bar{x}_0$.

OCP structured NLP handled in acados

$$
\begin{aligned}
\underset{\substack{x_0,\ldots,x_N,\\ u_0,\ldots,u_{N-1},\\ z_0,\ldots,z_{N-1},\\ s_0,\ldots,s_N}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} l_k(x_k, u_k, z_k) + M(x_N) + \sum_{k=0}^{N} \rho_k(s_k) && && \text{(2a)}\\
\text{subject to} \quad & \begin{bmatrix} x_{k+1} \\ z_k \end{bmatrix} = \phi_k(x_k, u_k), && k = 0, \ldots, N-1, && \text{(2b)}\\
& 0 \geq g_k(x_k, z_k, u_k) - J_{\mathrm{s},k} s_k && k = 0, \ldots, N-1, && \text{(2c)}\\
& 0 \geq g_N(x_N) - J_{\mathrm{s},N} s_N, && && \text{(2d)}\\
& 0 \leq s_k && k = 0, \ldots, N. && \text{(2e)}
\end{aligned}
$$

- ϕ_k – discrete time dynamics on $[t_k, t_{k+1}]$ – typically acados integrator from ODE or DAE
- l_k – approximation of Lagrange cost term ℓ on $[t_k, t_{k+1}]$
- efficient treatment of slack variables s_k, with linear-quadratic penalties $\rho_k(\cdot)$
- inequality constraints g_k
- general formulation: problem functions can vary stage wise

Ingredients of SQP-type methods and acados modules

SQP-type algorithm:

- NLP solver
- Linearization
- Regularization
- QP solution
- Globalization

Ingredients of SQP-type methods and acados modules

SQP-type algorithm:
NLP solver – Linearization – Regularization – QP solution – Globalization

acados module	Variants
OCP-NLP solver	SQP, RTI, AS-RTI [1], DDP[2], SQP_WITH_FEASIBLE_QP
Nonlinear functions	CasADi[8] generated, generic C functions
Dynamics	ERK, IRK, GNSF-IRK[7], Discrete dynamics
Hessian approximation	Exact, Gauss-Newton, Convex-over-nonlinear, custom
Regularization	Mirror, Project, Convexify [3]
Condensing	Full condensing, Partial condensing [4]
OCP QP	HPIPM[5], OSQP[9], qpDUNES, HPMPC
Dense QP	HPIPM, qpOASES, DAQP[6]
Globalization	Merit function, Funnel

[1](Frey et al., 2024), [2](Kiessling et al., 2024), [3](Verschueren et al., 2017) [4](Frison et al., 2016) [5](Frison & Diehl, 2020), [6](Arnstrom et al., 2022), [7](Frey et al., 2019) [8](Andersson et al., 2019), [9](Stellato et al., 2020)

Important Ressources: Documentation page

docs.acados.org/

acados

Search Ctrl + K

Home

Real-world examples

Citing

Installation

Related Projects

Interfaces

Interfaces Overview

Python Interface

MATLAB + Simulink and Octave Interface

User Guide

Problem Formulation

Troubleshooting

acados

Test Full Build Linux passing | Test Build and C tests with BLASFEO REFERENCE and OpenMP passing | build passing

Fast and embedded solvers for real-world applications of nonlinear optimal control.

Important links

🎦 Get inspired by real-world applications using acados 🚀

⭐ The `acados` **source code** is hosted on Github. Contributions via pull requests are welcome!

🤝 `acados` has a discourse-based forum.

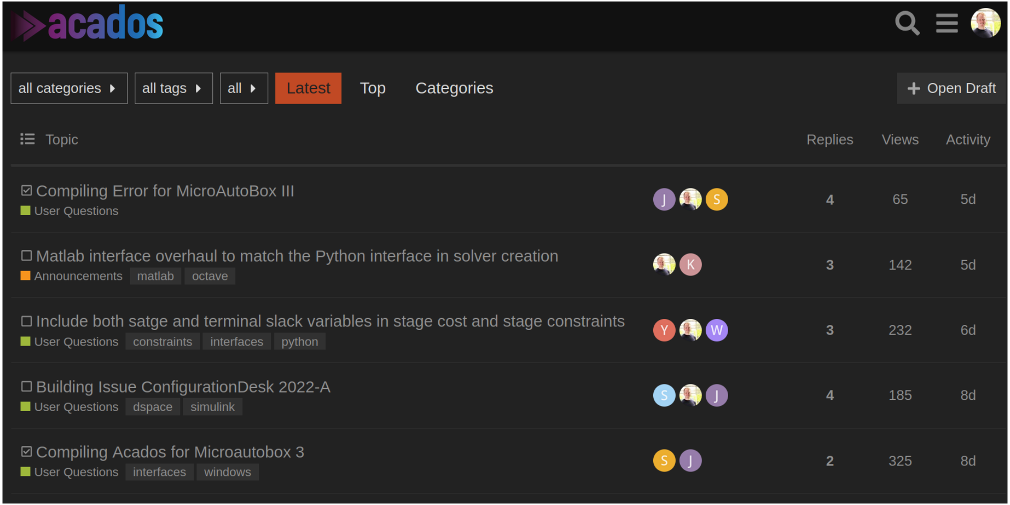
🏘 `acados` is mainly developed by the syscop group around Prof. Moritz Diehl, at the University of Freiburg.

About `acados`

`acados` is a modular and efficient software package for solving nonlinear programs (NLP) with an optimal control problem (OCP) structure. Such problems have to be solved repeatedly in **model predictive control**

Important Ressources: acados forum

https://discourse.acados.org/



Multi-Phase Optimal Control Problems for Efficient Nonlinear Model Predictive Control with acados

– Jonathan Frey, Katrin Baumgärtner, Gianluca Frison, Moritz Diehl

Optimal Control Applications and Methods, 2025

Multi-phase OCPs for efficient MPC with acados

Classic approach: for continuous-time, inifinite horizon problem

- Choose time horizon T, discretize with N stages
- Capture remaining infinite horizon in terminal cost

Multi-phase OCPs for efficient MPC with acados

Classic approach: for continuous-time, inifinite horizon problem

- Choose time horizon T, discretize with N stages
- Capture remaining infinite horizon in terminal cost

Multi-phase approach: allow more flexible treatment

- Conceptionally: OCP is initial stage u_0 and cost-to-go approximation
- Allows successively coarser formulation and models over the horizon

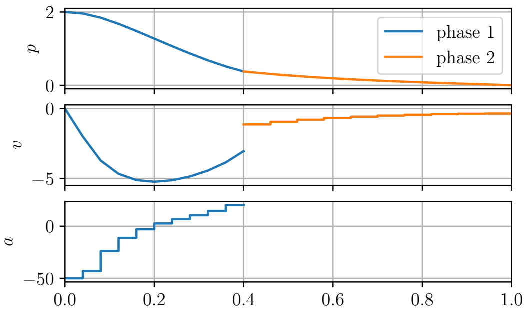
Multi-phase OCPs for efficient MPC with acados

Classic approach: for continuous-time, inifinite horizon problem

- Choose time horizon T, discretize with N stages
- Capture remaining infinite horizon in terminal cost

Multi-phase approach: allow more flexible treatment

- Conceptionally: OCP is initial stage u_0 and cost-to-go approximation
- Allows successively coarser formulation and models over the horizon
  - Phase 1: $x = [p, v], u = a$
  - Phase 2: $x = p, u = v$



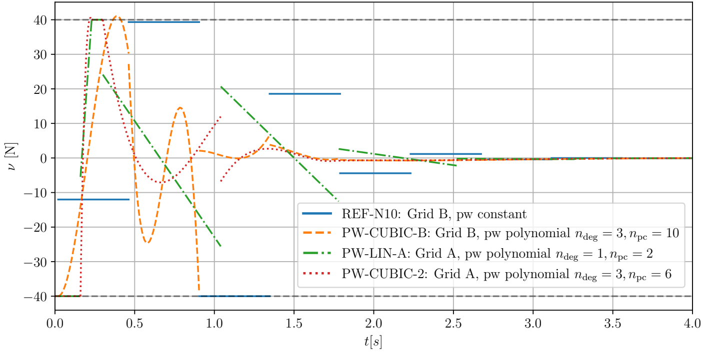
Multi-phase OCPs for efficient MPC with acados

Classic approach: for continuous-time, inifinite horizon problem

- Choose time horizon T, discretize with N stages
- Capture remaining infinite horizon in terminal cost

Multi-phase approach: allow more flexible treatment

- Conceptionally: OCP is initial stage u_0 and cost-to-go approximation
- Allows successively coarser formulation and models over the horizon
- Variety of control parameterizations, e.g. piecewise polynomial, closed-loop costing, ...



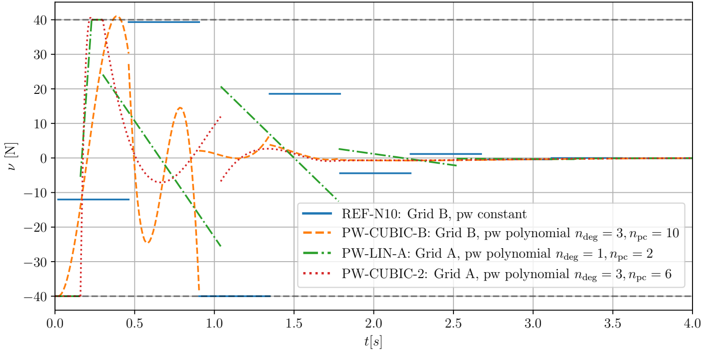
Multi-phase OCPs for efficient MPC with acados

Classic approach: for continuous-time, inifinite horizon problem

- Choose time horizon T, discretize with N stages
- Capture remaining infinite horizon in terminal cost

Multi-phase approach: allow more flexible treatment

- Conceptionally: OCP is initial stage u_0 and cost-to-go approximation
- Allows successively coarser formulation and models over the horizon
- Variety of control parameterizations, e.g. piecewise polynomial, closed-loop costing, ...



Summary (Frey et al., 2025): MOCP based NMPC controllers can trade off computation time and performance more effiently than standard OCPs.

Advanced-Step Real-Time Iterations with Four Levels – New Error Bounds and Fast Implementation in acados

– Jonathan Frey, Armin Nurkanović, Moritz Diehl

IEEE Control Systems Letters, 2024

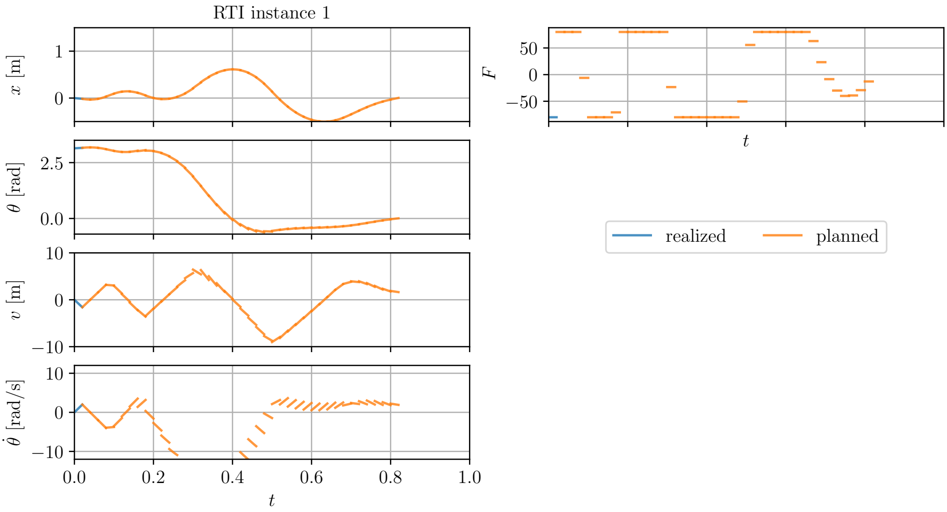
Real-time algorithms

- The Real-Time Iteration (RTI) performs only one SQP iteration in each sampling interval, (DIehl et al., 2001)
- Idea: give fast feedback and "converge over time" – examples follow

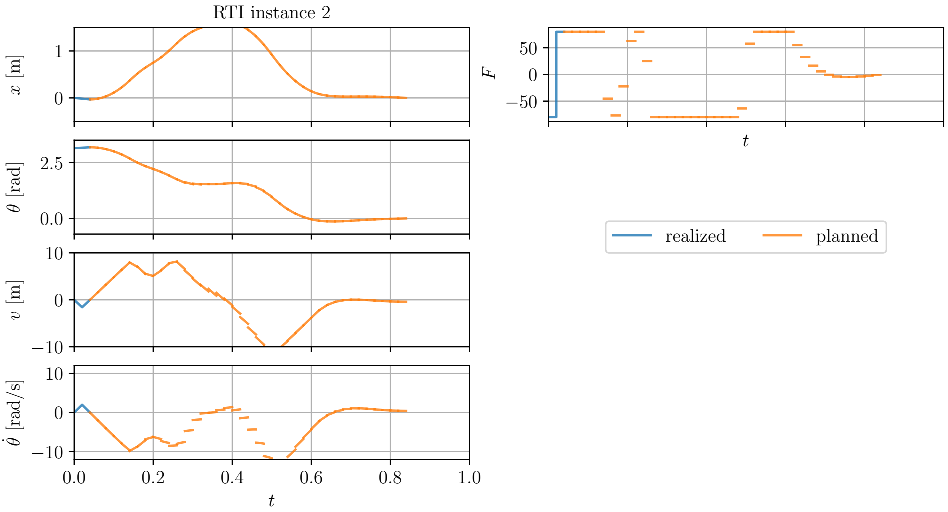
Real-time algorithms

- The Real-Time Iteration (RTI) performs only one SQP iteration in each sampling interval, (DIehl et al., 2001)
- Idea: give fast feedback and "converge over time" – examples follow
- Additionally: $\bar{x}_0$ enters only constraints linearly
  $\Longrightarrow$ allows to split SQP iteration into a feedback and a preparation phase.

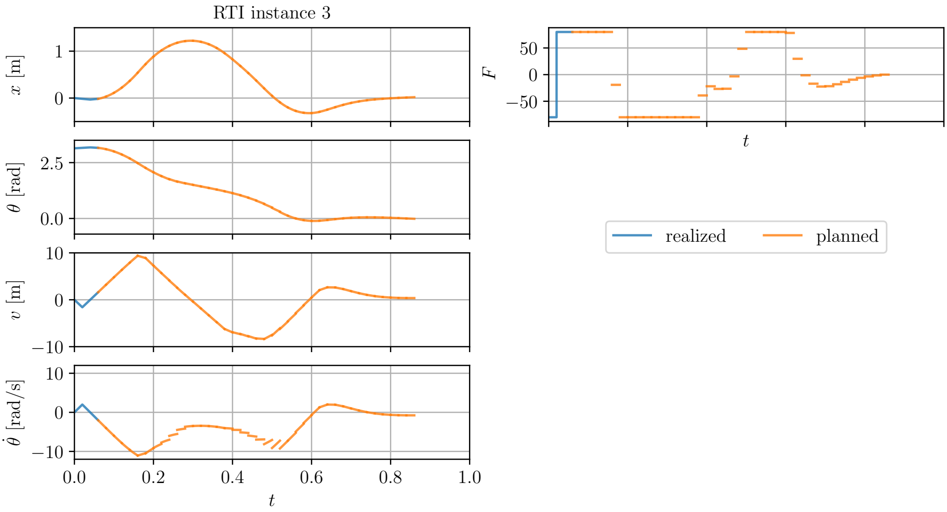
Convergence over time illustration – RTI



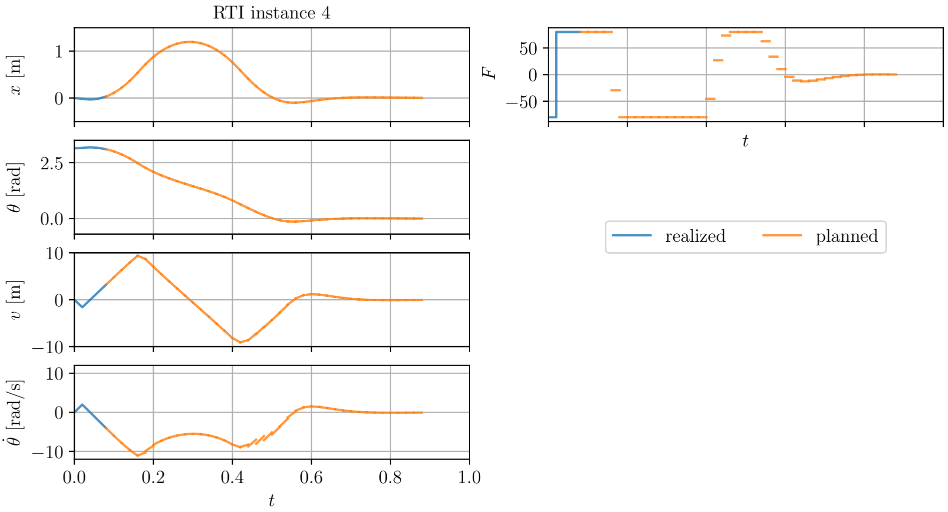
Convergence over time illustration – RTI



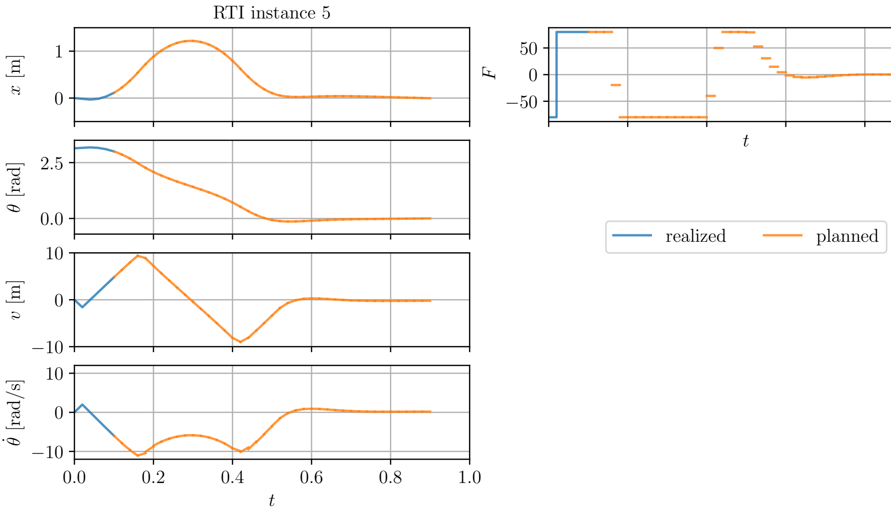
Convergence over time illustration – RTI



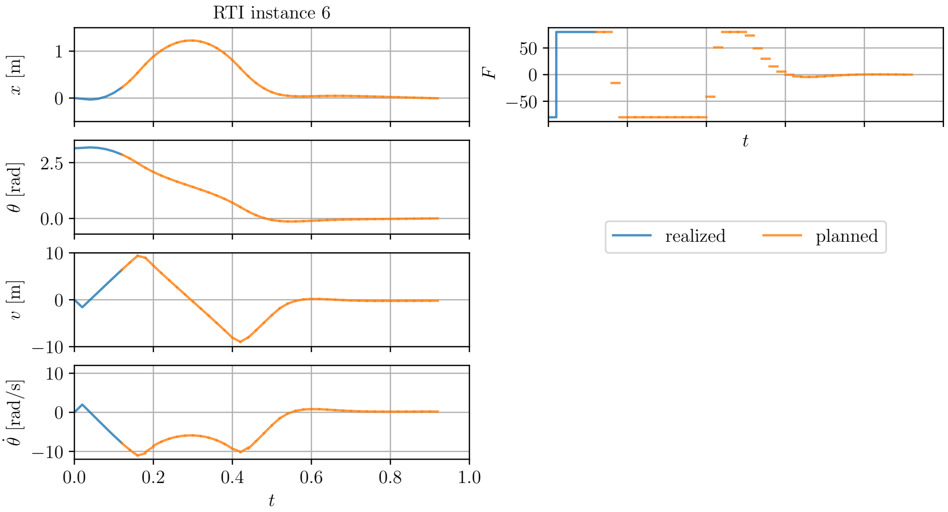
Convergence over time illustration – RTI



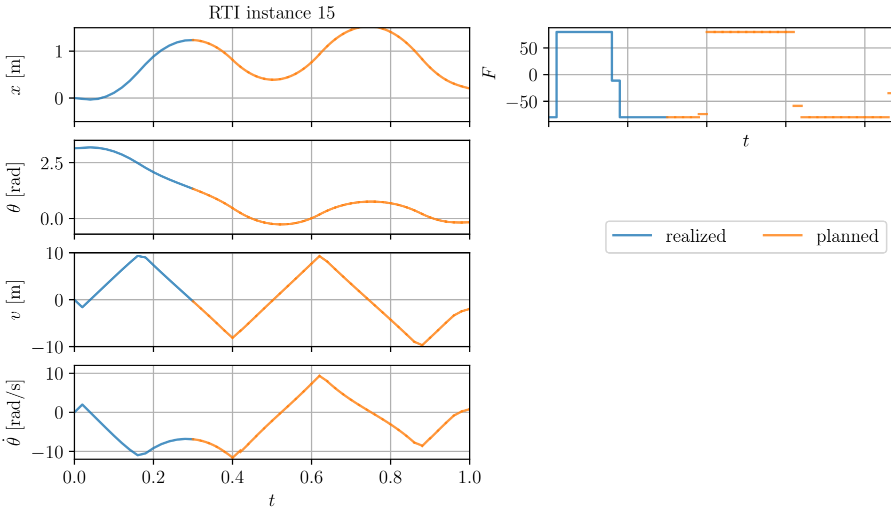
Convergence over time illustration – RTI



Convergence over time illustration – RTI



Convergence over time illustration – RTI



Real-time NMPC algorithms

$$\begin{aligned} \underset{w \in \mathbb{R}^{n_w}}{\text{minimize}} \quad & f(w) && \text{(3a)} \\ \text{subject to} \quad & 0 = g(w) + Mx, && \text{(3b)} \\ & 0 \leq h(w), && \text{(3c)} \end{aligned}$$

$$\begin{aligned} \underset{\Delta w}{\text{minimize}} \quad & (a^{k,j})^\top \Delta w + \tfrac{1}{2} \Delta w^\top A^{k,j} \Delta w && \text{(4a)} \\ \text{subject to} \quad & g^{k,j} + Mx^k + G^{k,j} \Delta w = 0, && \text{(4b)} \\ & h^{k,j} + H^{k,j} \Delta w \geq 0. && \text{(4c)} \end{aligned}$$

Real-time NMPC algorithms

$$\underset{w \in \mathbb{R}^{n_w}}{\text{minimize}} \quad f(w) \tag{3a}$$
$$\text{subject to} \quad 0 = g(w) + Mx, \tag{3b}$$
$$0 \leq h(w), \tag{3c}$$

$$\underset{\Delta w}{\text{minimize}} \quad (a^{k,j})^\top \Delta w + \tfrac{1}{2} \Delta w^\top A^{k,j} \Delta w \tag{4a}$$
$$\text{subject to} \quad g^{k,j} + Mx^k + G^{k,j} \Delta w = 0, \tag{4b}$$
$$h^{k,j} + H^{k,j} \Delta w \geq 0. \tag{4c}$$

AS-RTI steps

(S1) At time $t = t^k$: Predict the initial state x_{pred}^{k+1} at t^{k+1}

(S2) At $t \in \left[t^k, t^{k+1}\right)$: Starting with z^k, iterate on (3) with $x = x_{\text{pred}}^{k+1}$ to obtain z_{lin}^k – "*the inner iterations*". Use MLI variant (next slide)

Real-time NMPC algorithms

$$\begin{aligned} \underset{w \in \mathbb{R}^{n_w}}{\text{minimize}} \quad & f(w) && (3a) \\ \text{subject to} \quad & 0 = g(w) + Mx, && (3b) \\ & 0 \leq h(w), && (3c) \end{aligned}$$

$$\begin{aligned} \underset{\Delta w}{\text{minimize}} \quad & (a^{k,j})^\top \Delta w + \tfrac{1}{2} \Delta w^\top A^{k,j} \Delta w && (4a) \\ \text{subject to} \quad & g^{k,j} + Mx^k + G^{k,j} \Delta w = 0, && (4b) \\ & h^{k,j} + H^{k,j} \Delta w \geq 0. && (4c) \end{aligned}$$

AS-RTI steps

(S1) At time $t = t^k$: Predict the initial state x^{k+1}_{pred} at t^{k+1}

(S2) At $t \in \left[t^k, t^{k+1}\right)$: Starting with z^k, iterate on (3) with $x = x^{k+1}_{\text{pred}}$ to obtain z^k_{lin} – "*the inner iterations*". Use MLI variant (next slide)

(S3) At $t \in \left[t^k, t^{k+1}\right)$: Construct QP (4) on the linearization point z^k_{lin}.

(S4) At time t^{k+1}, solve (4) with $x = x^{k+1}$. – "feedback phase"

Real-time NMPC algorithms

$$\begin{aligned} \underset{w \in \mathbb{R}^{n_w}}{\text{minimize}} \quad & f(w) && \text{(3a)} \\ \text{subject to} \quad & 0 = g(w) + Mx, && \text{(3b)} \\ & 0 \leq h(w), && \text{(3c)} \end{aligned}$$

$$\begin{aligned} \underset{\Delta w}{\text{minimize}} \quad & (a^{k,j})^\top \Delta w + \tfrac{1}{2} \Delta w^\top A^{k,j} \Delta w && \text{(4a)} \\ \text{subject to} \quad & g^{k,j} + Mx^k + G^{k,j} \Delta w = 0, && \text{(4b)} \\ & h^{k,j} + H^{k,j} \Delta w \geq 0. && \text{(4c)} \end{aligned}$$

AS-RTI steps

(S1) At time $t = t^k$: Predict the initial state x_{pred}^{k+1} at t^{k+1}

(S2) At $t \in [t^k, t^{k+1})$: Starting with z^k, iterate on (3) with $x = x_{\text{pred}}^{k+1}$ to obtain z_{lin}^k – "*the inner iterations*". Use MLI variant (next slide)

(S3) At $t \in [t^k, t^{k+1})$: Construct QP (4) on the linearization point z_{lin}^k.

(S4) At time t^{k+1}, solve (4) with $x = x^{k+1}$. – "feedback phase"

Remarks

- RTI: 2 simplifies to setting $z_{\text{lin}}^k = z^k$ or shifted variant
- Advanced-step controller (ASC): z_{lin}^k is a local minimizer of (3) with $x = x_{\text{pred}}^{k+1}$
- Denote AS-RTI with level X iteration as *AS-RTI-X*;

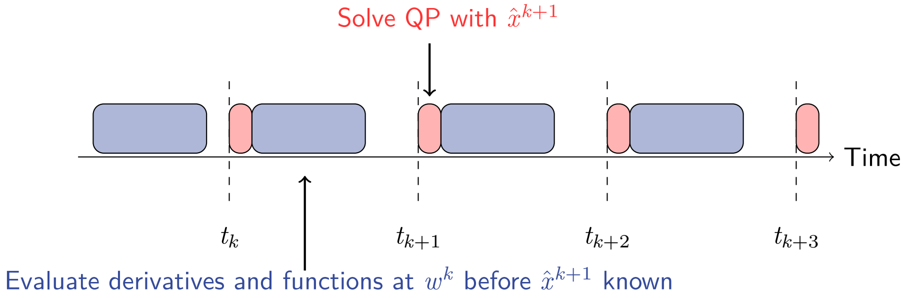
Schematic overview of the real-time iterations for NMPC

Preparation phase:

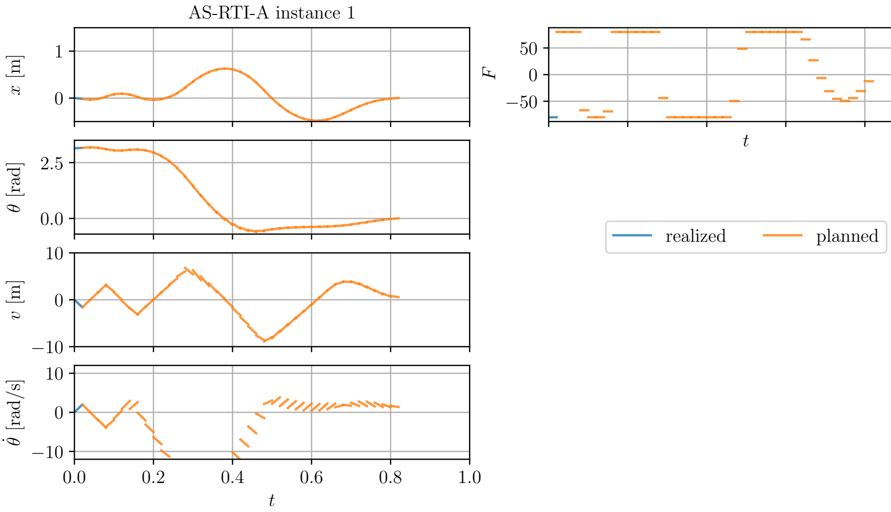
- at $t \in [t^k, t^{k+1})$: eval. derivatives at w^k, construct QP

Feedback phase:

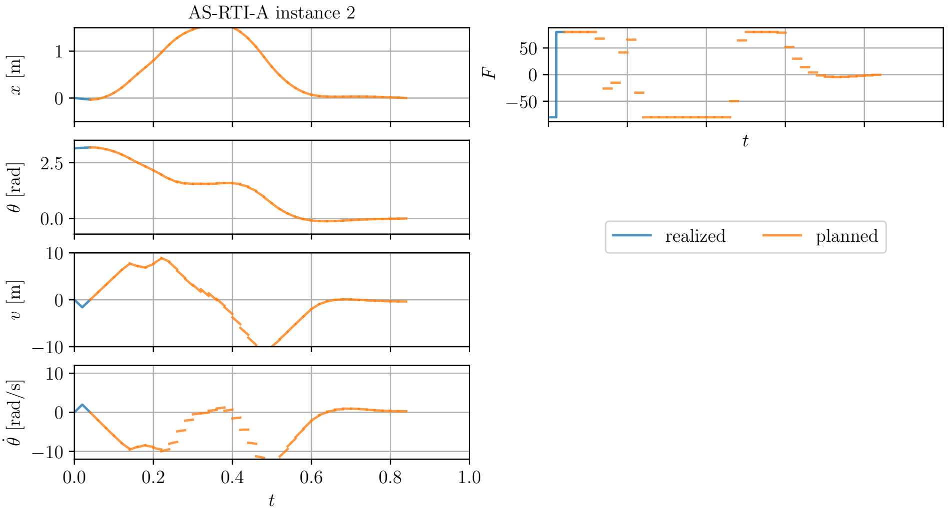
- at t^{k+1}, solve QP with $x = \hat{x}^{k+1}$ for $w^{k+1} = w^k + \Delta w^k$
- at $t^{k+1} + \delta t_{\mathrm{qp}}$ pass $u_0(\hat{x}^{k+1}) = \Pi w^{k+1}$ to the plant



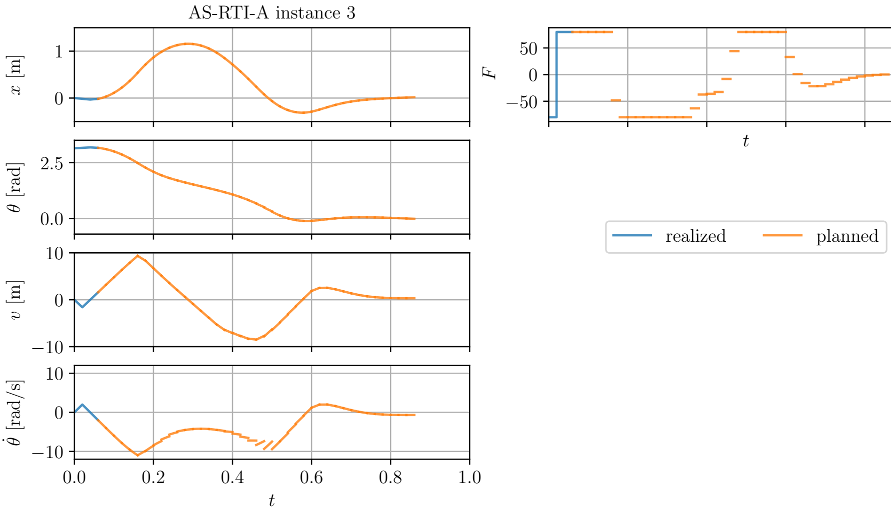
Convergence over time illustration – AS-RTI-A



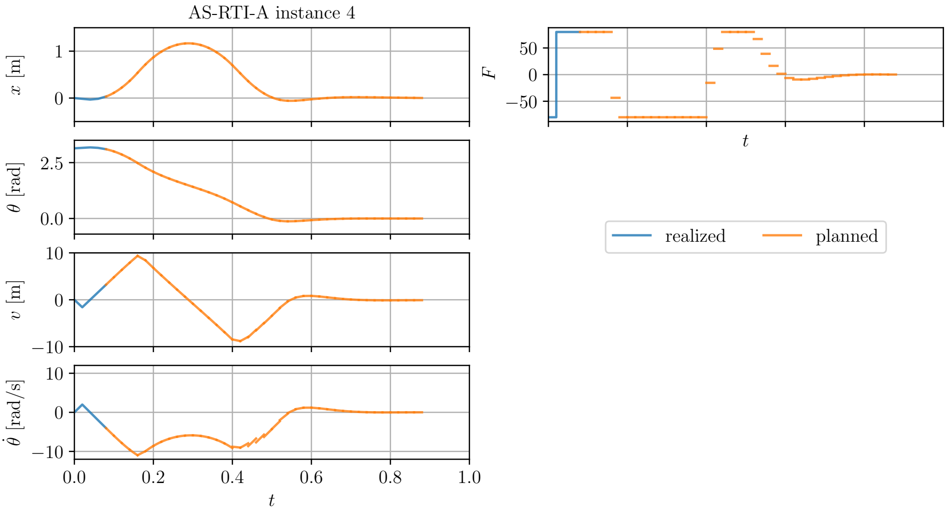
Convergence over time illustration – AS-RTI-A



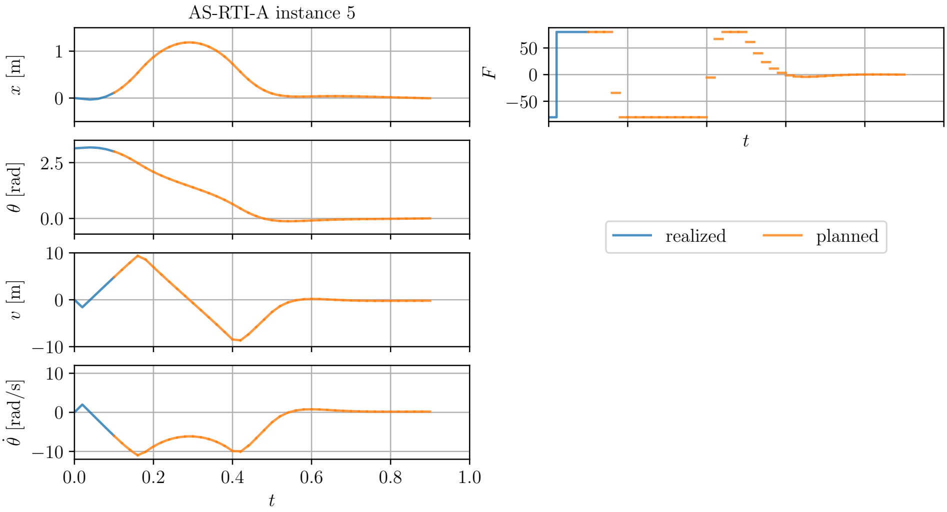
Convergence over time illustration – AS-RTI-A



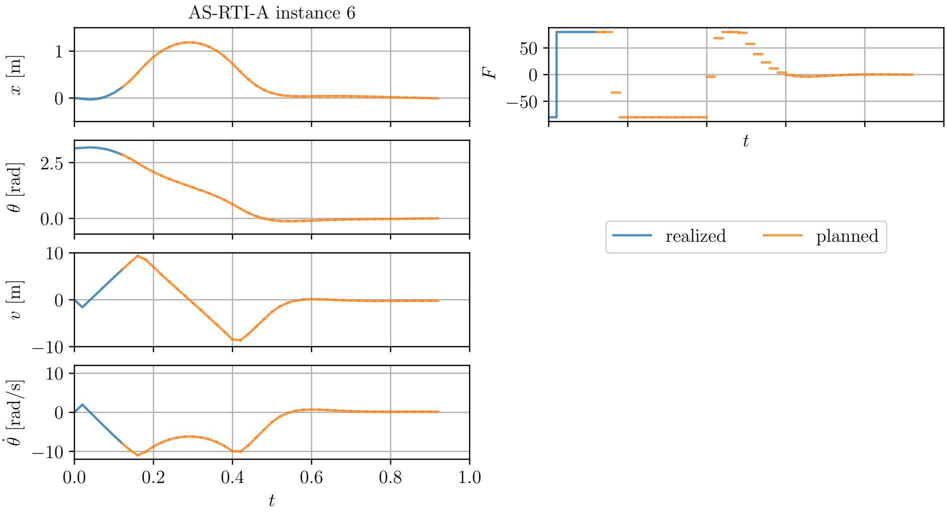
Convergence over time illustration – AS-RTI-A



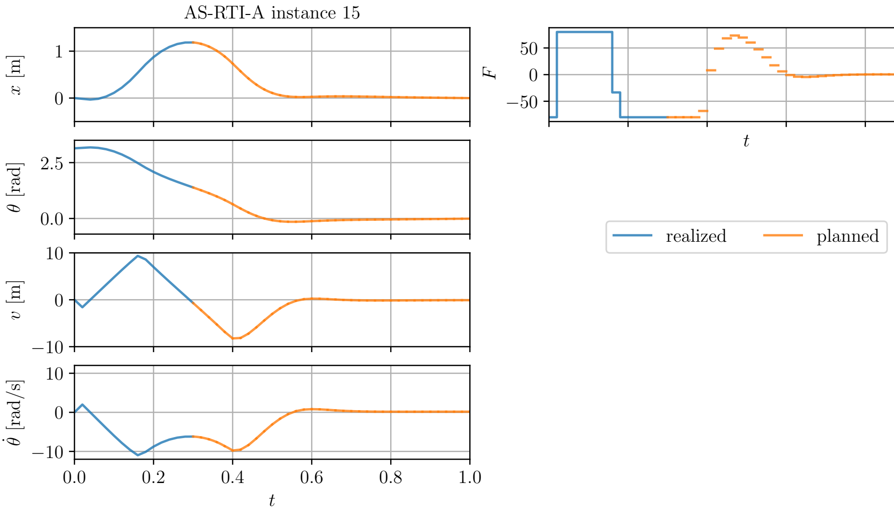
Convergence over time illustration – AS-RTI-A



Convergence over time illustration – AS-RTI-A



Convergence over time illustration – AS-RTI-A



Differentiable Nonlinear Model Predictive Control

– Jonathan Frey, Katrin Baumgärtner, Gianluca Frison, Dirk Reinhardt, Jasper Hoffmann, Leonard Fichtner, Sebastien Gros, Moritz Diehl

https://arxiv.org/abs/2505.01353

Solution sensitivities – Intro

Motivation

- Embedding optimization solvers in neural networks requires solution sensitivities
- Learning-enhanced MPC schemes, MPC-RL

Solution sensitivities – Intro

Motivation

- Embedding optimization solvers in neural networks requires solution sensitivities
- Learning-enhanced MPC schemes, MPC-RL

Related works

- "Differentiable MPC" – flaws in nonlinear case, implementation fails with constraints, (Amos et al., 2018)

Solution sensitivities – Intro

Motivation

- Embedding optimization solvers in neural networks requires solution sensitivities
- Learning-enhanced MPC schemes, MPC-RL

Related works

- "Differentiable MPC" – flaws in nonlinear case, implementation fails with constraints, (Amos et al., 2018)
- `cvxpylayers`, `cvxpygen`, limitation to convex problems, no OCP structure exploitation, (Agrawal et al., 2019; Schaller & Boyd, 2025)

Solution sensitivities – Intro

Motivation

- Embedding optimization solvers in neural networks requires solution sensitivities
- Learning-enhanced MPC schemes, MPC-RL

Related works

- "Differentiable MPC" – flaws in nonlinear case, implementation fails with constraints, (Amos et al., 2018)
- `cvxpylayers`, `cvxpygen`, limitation to convex problems, no OCP structure exploitation, (Agrawal et al., 2019; Schaller & Boyd, 2025)

Approach

🔮 Implicit function theorem on smoothed interior-point KKT system

⚡ Efficient Riccati factorization based on `HPIPM`

🔁 Adjoint sensitivities for efficient backward pass

KKT conditions & Smoothing

$$
\begin{aligned}
z^{\mathrm{sol}}(\theta) := \underset{z\in\mathbb{R}^{n_z}}{\arg\min} \quad & f(z;\theta) \\
\text{subject to} \quad & g(z;\theta) = 0, \\
& h(z;\theta) \leq 0
\end{aligned}
$$

KKT conditions & Smoothing

$$
\begin{aligned}
z^{\mathrm{sol}}(\theta) := \quad & \underset{z \in \mathbb{R}^{n_z}}{\arg\min} && f(z;\theta) \\
& \text{subject to} && g(z;\theta) = 0, \\
& && h(z;\theta) \leq 0
\end{aligned}
$$

Wanted: $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ 🤠💰

Simple dense NLP example

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & (x - \theta^2)^2 \\ \text{subject to} \quad & -1 \leq x \leq 1, \end{aligned}$$

Nondifferentiable solution map

$$x^\star(\theta) = \begin{cases} \theta^2, \text{ if } \theta \in [-1, 1] \\ 1, \text{ otherwise} \end{cases}$$

Derivative

$$\partial_\theta x^\star(\theta) = \begin{cases} 2 \cdot \theta, \text{ if } \theta \in (-1, 1) \\ 0, \text{ if } |\theta| > 1 \\ \text{not defined, for } \theta \in -1, 1 \end{cases}$$

Simple dense NLP example

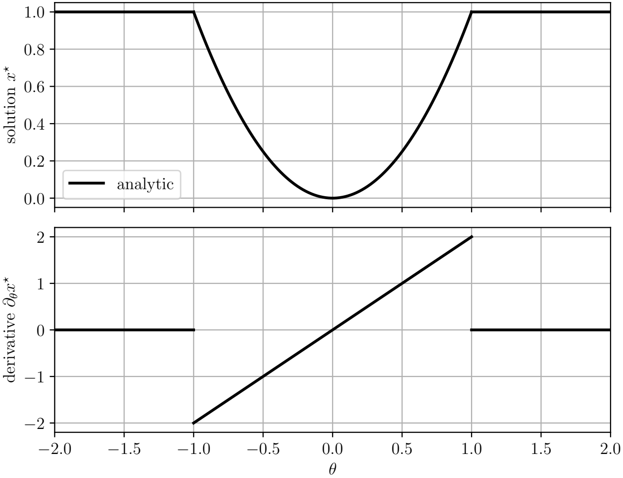
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & (x-\theta^2)^2 \\ \text{subject to} \quad & -1 \leq x \leq 1, \end{aligned}$$

Nondifferentiable solution map

$$x^\star(\theta) = \begin{cases} \theta^2, \text{ if } \theta \in [-1,1] \\ 1, \text{ otherwise} \end{cases}$$

Derivative

$$\partial_\theta x^\star(\theta) = \begin{cases} 2 \cdot \theta, \text{ if } \theta \in (-1,1) \\ 0, \text{ if } |\theta| > 1 \\ \text{not defined, for } \theta \in -1, 1 \end{cases}$$



KKT conditions & Smoothing

$$z^{\mathrm{sol}}(\theta) := \underset{z \in \mathbb{R}^{n_z}}{\arg\min} \quad f(z;\theta)$$
$$\text{subject to} \quad g(z;\theta) = 0,$$
$$h(z;\theta) \le 0$$

Wanted: $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ 🤠💰

KKT conditions & Smoothing

$$z^{\mathrm{sol}}(\theta) := \begin{array}{ll} \displaystyle\arg\min_{z \in \mathbb{R}^{n_z}} & f(z;\theta) \\ \text{subject to} & g(z;\theta) = 0, \\ & h(z;\theta) \leq 0 \end{array}$$

Wanted: $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ 🤠💰

Lagrangian function

$$\mathcal{L}(z, \lambda, \mu; \theta) = f(z;\theta) + \lambda^\top g(z;\theta) + \mu^\top h(z;\theta).$$

KKT conditions & Smoothing

$$z^{\text{sol}}(\theta) := \begin{aligned}[t] &\underset{z\in\mathbb{R}^{n_z}}{\arg\min} && f(z;\theta) \\ &\text{subject to} && g(z;\theta) = 0, \\ & && h(z;\theta) \leq 0 \end{aligned}$$

Wanted: $\frac{\partial z^{\text{sol}}}{\partial \theta}(\theta)$ 🤠💰

Lagrangian function

$$\mathcal{L}(z,\lambda,\mu;\theta) = f(z;\theta) + \lambda^\top g(z;\theta) + \mu^\top h(z;\theta).$$

KKT conditions

$$\begin{aligned} \nabla_z f(z;\theta) + \nabla_z g(z;\theta)\lambda + \nabla_z h(z;\theta)\mu &= 0, \\ g(z;\theta) &= 0, \\ h(z;\theta) &\leq 0, \\ \mu &\geq 0, \\ \mu_i h_i(z;\theta) &= 0,\ i = 1,\ldots,n_h. \end{aligned}$$

KKT conditions & Smoothing

$$z^{\text{sol}}(\theta) := \underset{z \in \mathbb{R}^{n_z}}{\arg\min} \quad f(z;\theta) \quad \text{subject to} \quad g(z;\theta) = 0, \quad h(z;\theta) \leq 0$$

Wanted: $\frac{\partial z^{\text{sol}}}{\partial \theta}(\theta)$ 🤠💰

Lagrangian function

$$\mathcal{L}(z, \lambda, \mu; \theta) = f(z;\theta) + \lambda^\top g(z;\theta) + \mu^\top h(z;\theta).$$

Interior-point smoothed KKT conditions

$$\begin{aligned}
\nabla_z f(z;\theta) + \nabla_z g(z;\theta)\lambda + \nabla_z h(z;\theta)\mu &= 0,\\
g(z;\theta) &= 0,\\
h(z;\theta) &\leq 0,\\
\mu &\geq 0,\\
\mu_i h_i(z;\theta) &= \tau, \; i = 1, \dots, n_h.
\end{aligned}$$

Interior-point methods (IPM) solve this for $\tau \to 0$,
e.g. `IPOPT`, `HPIPM`, `FORCES`, `Clarabel`, `fmincon`, ...

Simple dense NLP example

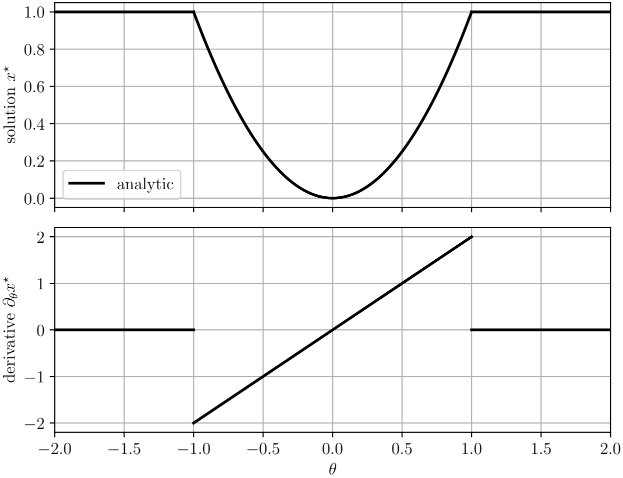
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & (x-\theta^2)^2 \\ \text{subject to} \quad & -1 \le x \le 1, \end{aligned}$$

Nondifferentiable solution map

$$x^\star(\theta) = \begin{cases} \theta^2, \text{ if } \theta \in [-1, 1] \\ 1, \text{ otherwise} \end{cases}$$

Derivative

$$\partial_\theta x^\star(\theta) = \begin{cases} 2 \cdot \theta, \text{ if } \theta \in (-1, 1) \\ 0, \text{ if } |\theta| > 1 \\ \text{not defined, for } \theta \in -1, 1 \end{cases}$$



Simple dense NLP example

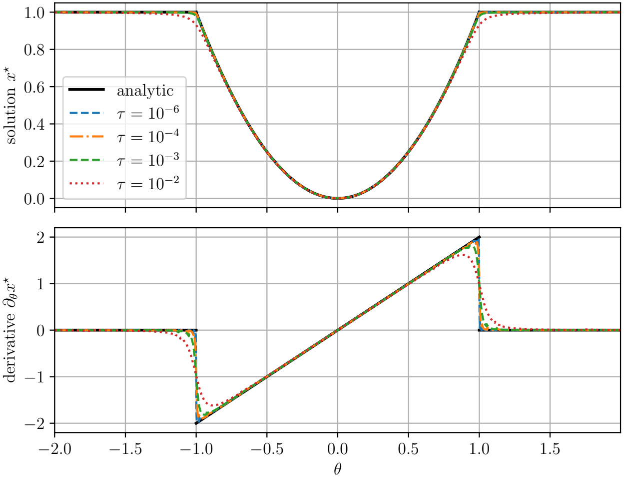
$$\begin{aligned} \underset{x}{\text{minimize}} \quad & (x - \theta^2)^2 \\ \text{subject to} \quad & -1 \leq x \leq 1, \end{aligned}$$

Nondifferentiable solution map

$$x^\star(\theta) = \begin{cases} \theta^2, & \text{if } \theta \in [-1, 1] \\ 1, & \text{otherwise} \end{cases}$$

Derivative

$$\partial_\theta x^\star(\theta) = \begin{cases} 2 \cdot \theta, & \text{if } \theta \in (-1, 1) \\ 0, & \text{if } |\theta| > 1 \\ \text{not defined}, & \text{for } \theta \in -1, 1 \end{cases}$$



Code `acados/examples/acados_python/solution_sensitivities_convex_example/non_ocp_example.py`

Assumptions

- Problem functions f, g, h, twice differentiable in z, once in θ.
- $(z^\star, \lambda^\star, \mu^\star)$ KKT point of the NLP with LICQ, SOSC and strict complementarity for $\bar{\theta}$

Theory: Solution map & IP Smoothing

Assumptions

- Problem functions f, g, h, twice differentiable in z, once in θ.
- $(z^\star, \lambda^\star, \mu^\star)$ KKT point of the NLP with LICQ, SOSC and strict complementarity for $\bar{\theta}$

Theoretical results

- In a neighborhood of $\bar{\theta}$, there exists a differentiable function $z^{\mathrm{sol}}(\theta)$ with $z^{\mathrm{sol}}(\bar{\theta}) = z^\star$ that corresponds to a locally unique solution.

Assumptions

- Problem functions f, g, h, twice differentiable in z, once in θ.
- $(z^\star, \lambda^\star, \mu^\star)$ KKT point of the NLP with LICQ, SOSC and strict complementarity for $\bar{\theta}$

Theoretical results

- In a neighborhood of $\bar{\theta}$, there exists a differentiable function $z^{\mathrm{sol}}(\theta)$ with $z^{\mathrm{sol}}(\bar{\theta}) = z^\star$ that corresponds to a locally unique solution.

For small positive values of τ

- The solution of the smoothed IP KKT system $z^{\mathrm{sol}}_{\mathrm{IPM}}(\tau; \bar{\theta})$ is a continuously differentiable function with $\lim_{\tau \to 0^+} z^{\mathrm{sol}}_{\mathrm{IPM}}(\tau, \bar{\theta}) = z^{\mathrm{sol}}(\bar{\theta})$ and $\left\| z^{\mathrm{sol}}_{\mathrm{IPM}}(\tau; \bar{\theta}) - z^\star \right\| \in \mathcal{O}(\tau)$
- In a neighborhood of $\bar{\theta}$, there exists a differentiable function $v(\tau; \theta) = (z(\tau; \theta), \lambda(\tau; \theta), \mu(\tau; \theta))$ that corresponds to a locally unique solution of the smoothed interior-point KKT system and $v(0; \bar{\theta}) := \lim_{\tau \to 0^+} v(\tau; \bar{\theta}) = (z^\star, \lambda^\star, \mu^\star)$ holds.

SQP and IPM

Setting: solve NLP with `acados` SQP

⚠️ SQP solves QP in Δ space of primal variables

SQP and IPM

Setting: solve NLP with `acados` SQP

⚠️ SQP solves QP in Δ space of primal variables

Theorem: Denote QP solution map at NLP solution $\Delta z_{\mathrm{QP}}^{\mathrm{sol}}(\theta, v^\star)$. For exact Hessian QP, the solution maps $z^{\mathrm{sol}}(\theta)$ and $z^\star + \Delta z_{\mathrm{QP}}^{\mathrm{sol}}(\theta, v^\star)$, and their sensitivities, $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ and $\frac{\partial \Delta z_{\mathrm{QP}}^{\mathrm{sol}}}{\partial \theta}(\theta, v^\star)$ coincide.

SQP and IPM

Setting: solve NLP with `acados` SQP

⚠️ SQP solves QP in Δ space of primal variables

Theorem: Denote QP solution map at NLP solution $\Delta z_{\mathrm{QP}}^{\mathrm{sol}}(\theta, v^\star)$. For exact Hessian QP, the solution maps $z^{\mathrm{sol}}(\theta)$ and $z^\star + \Delta z_{\mathrm{QP}}^{\mathrm{sol}}(\theta, v^\star)$, and their sensitivities, $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ and $\frac{\partial \Delta z_{\mathrm{QP}}^{\mathrm{sol}}}{\partial \theta}(\theta, v^\star)$ coincide.

Blending SQP with IP QP solver (`HPIPM`): Shrink τ in QP solver to $\tau_{\min} > 0$ instead of 0.

SQP and IPM

Setting: solve NLP with `acados` SQP

⚠️ SQP solves QP in Δ space of primal variables

Theorem: Denote QP solution map at NLP solution $\Delta z^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v^\star)$. For exact Hessian QP, the solution maps $z^{\mathrm{sol}}(\theta)$ and $z^\star + \Delta z^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v^\star)$, and their sensitivities, $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ and $\frac{\partial \Delta z^{\mathrm{sol}}_{\mathrm{QP}}}{\partial \theta}(\theta, v^\star)$ coincide.

Blending SQP with IP QP solver (`HPIPM`): Shrink τ in QP solver to $\tau_{\min} > 0$ instead of 0.

⚠️ Not an SQP method for $\tau_{\min} > 0$

✅ Convergence to IP-smoothed KKT solution

SQP and IPM

Setting: solve NLP with `acados` SQP

⚠️ SQP solves QP in Δ space of primal variables

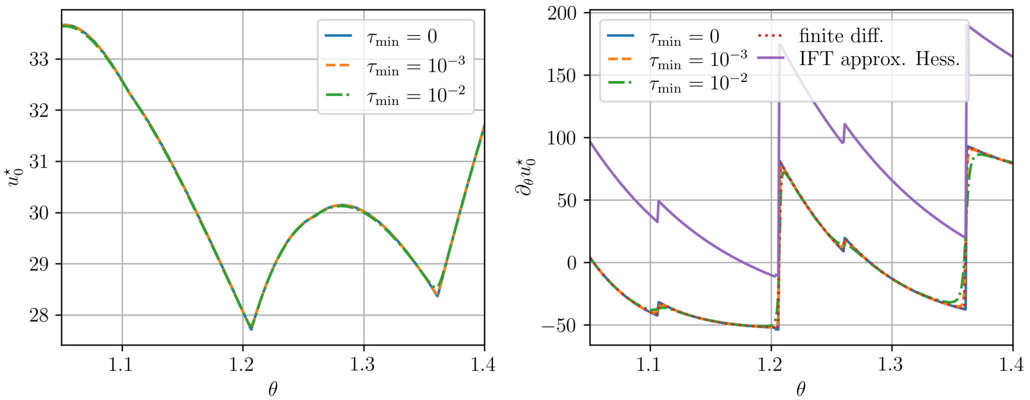
Theorem: Denote QP solution map at NLP solution $\Delta z_{\mathrm{QP}}^{\mathrm{sol}}(\theta, v^\star)$. For exact Hessian QP, the solution maps $z^{\mathrm{sol}}(\theta)$ and $z^\star + \Delta z_{\mathrm{QP}}^{\mathrm{sol}}(\theta, v^\star)$, and their sensitivities, $\frac{\partial z^{\mathrm{sol}}}{\partial \theta}(\theta)$ and $\frac{\partial \Delta z_{\mathrm{QP}}^{\mathrm{sol}}}{\partial \theta}(\theta, v^\star)$ coincide.

Blending SQP with IP QP solver (`HPIPM`): Shrink τ in QP solver to $\tau_{\min} > 0$ instead of 0.

⚠️ Not an SQP method for $\tau_{\min} > 0$

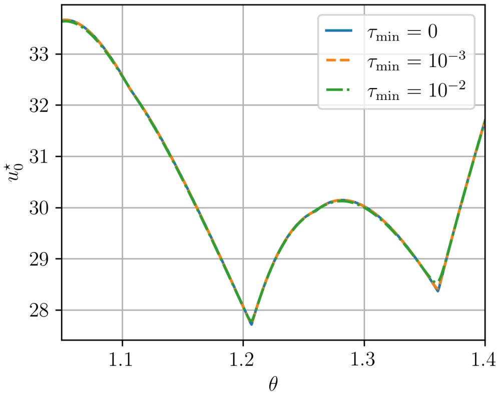
✅ Convergence to IP-smoothed KKT solution

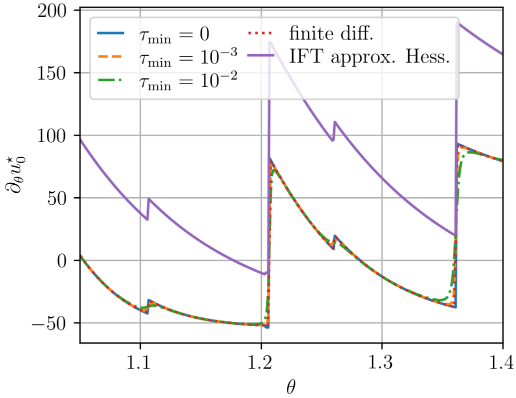
Highly-parametric optimal control example



- Pendulum on cart inspired
- θ in cost, dynamics, constraints
- θ mass of cart

Highly-parametric optimal control example





- Pendulum on cart inspired
- θ in cost, dynamics, constraints
- θ mass of cart

Wrong results!

⚠️ Gauss-Newton Hessian approx. in IFT

Hessian approximations & Two-solver approach

- Hessian approximations often beneficial in SQP
  - convergence
  - computational complexity
  - regularity
- Regularization needed when dealing with indefinite Hessians

⚠️ IFT requires **exact** Hessian ⚠️

Hessian approximations & Two-solver approach

- Hessian approximations often beneficial in SQP
  - convergence
  - computational complexity
  - regularity
- Regularization needed when dealing with indefinite Hessians

⚠️ IFT requires **exact** Hessian ⚠️

Two-solver approach

1. Nominal solver: can use approximate Hessian, regularization etc.
2. Sensitivity solver
   - load solution
   - evaluate exact Hessian
   - evaluate partial derivatives w.r.t. θ
   - solve linear system effiently with `HPIPM` Riccati

Hessian approximations & Two-solver approach

- Hessian approximations often beneficial in SQP
  - convergence
  - computational complexity
  - regularity
- Regularization needed when dealing with indefinite Hessians

⚠️ IFT requires **exact** Hessian ⚠️

Two-solver approach

1. Nominal solver: can use approximate Hessian, regularization etc.
2. Sensitivity solver
   - load solution
   - evaluate exact Hessian
   - evaluate partial derivatives w.r.t. θ
   - solve linear system effiently with `HPIPM` Riccati

Hessian approximations & Two-solver approach

- Hessian approximations often beneficial in SQP
  - convergence
  - computational complexity
  - regularity
- Regularization needed when dealing with indefinite Hessians

⚠️ IFT requires **exact** Hessian ⚠️

Two-solver approach

1. Nominal solver: can use approximate Hessian, regularization etc.
2. Sensitivity solver
   - load solution
   - evaluate exact Hessian
   - evaluate partial derivatives w.r.t. θ
   - solve linear system effiently with `HPIPM` Riccati

Benchmark: bounded LQR problems

Table: Timings in [ms] for solving $n_{\text{batch}} = 128$ instances with $N = 20$, $n_x = 8$, $n_u = 4$, $n_\theta = 248$. In parentheses are multiples of the acados runtime.

	Nominal solution			**Solution + adjoint sens.**		
$u_{\max}$	`acados`	`mpc.pytorch`	`cvxpygen`	`acados`	`mpc.pytorch`	`cvxpygen`
10^4	8.5	78 $(\times 9.2)$	262 $(\times 31)$	34.5	125 $(\times 3.6)$	658 $(\times 19)$
1.0	17.6	21024 $(\times 1200)$	6402 $(\times 360)$	42.0	21899 $(\times 520)$	6845 $(\times 160)$

Benchmark: details

$$
\begin{aligned}
\underset{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}}{\text{minimize}} \quad & \sum_{n=0}^{N-1} \begin{bmatrix} x_n \\ u_n \end{bmatrix}^\top H \begin{bmatrix} x_n \\ u_n \end{bmatrix} + x_N^\top H_x x_N && \text{(5a)}\\
\text{subject to} \quad & x_0 = \bar{x}_0, && \text{(5b)}\\
& x_{n+1} = A x_n + B u_n + b, \quad n = 0, \dots, N-1, && \text{(5c)}\\
& -u_{\max} \leq u_n \leq u_{\max}, \quad n = 0, \dots, N-1, && \text{(5d)}
\end{aligned}
$$

- $A = \mathbb{1} + 0.2 \cdot M$ and B, b and M sampled from standard normal distribution.
- $H = \mathbb{1}$ identity
- H_x submatrix with first n_x rows and columns of H.
- The problem data A, B, b, H is regarded as parameter θ, such that $n_\theta = n_x^2 + n_x n_u + n_x + (n_x + n_u)^2$.

Final slide

Summary

🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes

Final slide

Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation
- 🦉 In mature software acados

Final slide

Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation
- 🦉 In mature software acados
- ⏩ Adjoint solution sensitivities for efficient backward pass

Final slide

Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation
- 🦉 In mature software `acados`
- ⏩ Adjoint solution sensitivities for efficient backward pass
- 🕯️ Wrapped in `pytorch` layer in `leap-c`

Final slide

Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation
- 🦉 In mature software `acados`
- ⏩ Adjoint solution sensitivities for efficient backward pass
- 🕯️ Wrapped in `pytorch` layer in `leap-c`

Ongoing research

- 🌅 Incorporation in MPC-RL schemes and method comparison

Final slide

Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation
- 🦉 In mature software `acados`
- ⏩ Adjoint solution sensitivities for efficient backward pass
- 🕯️ Wrapped in `pytorch` layer in `leap-c`

Ongoing research

- 🌅 Incorporation in MPC-RL schemes and method comparison

Thanks for your attention! 🙏

I look forward to questions, discussions and collaborations!

References I

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019). Differentiable convex optimization layers. *Advances in neural information processing systems*, *32*.

Amos, B., Jimenez, I., Sacks, J., Boots, B., & Kolter, J. Z. (2018). Differentiable MPC for end-to-end planning and control. *Advances in neural information processing systems*, *31*.

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B., & Diehl, M. (2019). CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, *11*(1), 1–36. doi: 10.1007/s12532-018-0139-4

Arnstrom, D., Bemporad, A., & Axehill, D. (2022). A dual active-set solver for embedded quadratic programming using recursive LDL^T updates. *IEEE Transactions on Automatic Control*. doi: 10.1109/TAC.2022.3176430

References II

DIehl, M., Uslu, I., Findeisen, R., Schwarzkopf, S., Allgöwer, F., Bock, H. G., … Stein, E. (2001). Real-time optimization for large scale processes: Nonlinear model predictive control of a high purity distillation column. In M. Grötschel, S. O. Krumke, & J. Rambau (Eds.), *Online optimization of large scale systems: State of the art* (pp. 363–384). Springer. Retrieved from `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.8798` (download at: http://www.zib.de/dfg-echtzeit/Publikationen/Preprints/Preprint-01-16.html)

Frey, J., Baumgärtner, K., Frison, G., & Diehl, M. (2025). Multi-phase optimal control problems for efficient nonlinear model predictive control with acados. *Optimal Control Applications and Methods*, *46*(2), 827-845. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.3234` doi: https://doi.org/10.1002/oca.3234

Frey, J., Nurkanović, A., & Diehl, M. (2024). Advanced-step real-time iterations with four levels – new error bounds and fast implementation in acados. *IEEE Control Systems Letters*. doi: 10.1109/LCSYS.2024.3412007

References III

Frey, J., Quirynen, R., Kouzoupis, D., Frison, G., Geisler, J., Schild, A., & Diehl, M. (2019). Detecting and exploiting Generalized Nonlinear Static Feedback structures in DAE systems for MPC. In *Proceedings of the european control conference (ecc).*

Frison, G., & Diehl, M. (2020, July). HPIPM: a high-performance quadratic programming framework for model predictive control. In *Proceedings of the ifac world congress.* Berlin, Germany.

Frison, G., Kouzoupis, D., Jørgensen, J. B., & Diehl, M. (2016). An efficient implementation of partial condensing for nonlinear model predictive control. In *Proceedings of the ieee conference on decision and control (cdc)* (pp. 4457–4462).

Frison, G., Kouzoupis, D., Sartor, T., Zanelli, A., & Diehl, M. (2018). BLASFEO: Basic linear algebra subroutines for embedded optimization. *ACM Transactions on Mathematical Software (TOMS)*, *44*(4), 42:1–42:30. doi: 10.1145/3210754

Kiessling, D., Baumgärtner, K., Frey, J., Decré, W., Swevers, J., & Diehl, M. (2024). Fast generation of feasible trajectories in direct optimal control. *IEEE Control Systems Letters*.

References IV

Schaller, M., & Boyd, S. (2025). Code generation for solving and differentiating through convex optimization problems. *arXiv preprint arXiv:2504.14099*. Retrieved from `https://arxiv.org/abs/2504.14099`

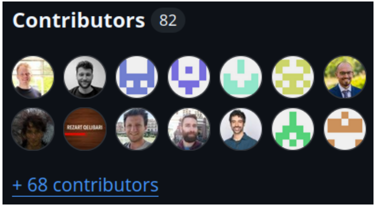
Stellato, B., Banjac, G., Goulart, P., Bemporad, A., & Boyd, S. (2020). OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, *12*(4), 637–672. Retrieved from `https://doi.org/10.1007/s12532-020-00179-2` doi: 10.1007/s12532-020-00179-2

Verschueren, R., Zanon, M., Quirynen, R., & Diehl, M. (2017). A sparsity preserving convexification procedure for indefinite quadratic programs arising in direct optimal control. *SIAM Journal of Optimization*, *27*(3), 2085–2109.

acados – fast embedded solvers for nonlinear optimal control

An open-source software package mainly developed in Freiburg, Germany

acados



Efficiency, usability, modularity, state-of-the-art optimization algorithms

- Written in `C` using high-performance linear algebra provided by `BLASFEO`
- Fully exploits sparsity of optimal control structured NLPs
- Interfaces to `Python`, MATLAB, `Simulink`
- nonlinear & symbolic models via `CasADi`
- Flexible problem formulation: multi-phase & MHE
- Minimal dependencies $\Longrightarrow$ embeddable
- Integrators for ODE & DAE: ERK & IRK, efficient sensitivity propagation
- QP solvers: full & partial condensing via `HPIPM` `HPIPM`, `DAQP`, `qpOASES`, `qpDUNES`, `OSQP`
- NLP solvers: SQP, DDP, RTI, AS-RTI
- Robust & stochastic MPC via zoRO
- Exploit convex-over-nonlinear structures

`github.com/acados/acados` `docs.acados.org` `discourse.acados.org`

QP solver types and sparsity – an overview

	Active-Set	Interior-Point	First-Order
dense	qpOASES, DAQP	HPIPM	
sparse	[PRESAS]	CVXGEN, OOQP	FiOrdOs, OSQP
OCP structure	qpDUNES, [ASIPM]	HPMPC, HPIPM, [ASIPM], [FORCES]	

Table: Overview: QP solver types and their way to handle sparsity.
underline: available in acados + support in Simulink
gray: not interfaced in acados, [proprietary]

efficient condensing from `HPIPM`:

- condensing: OCP structured $\rightarrow$ dense, expand solution
- partial condensing: OCP structured with horizon $N \rightarrow$ OCP structured with horizon $N_2 < N$, expand solution, $N_2 \hat{=}$ `qp_solver_cond_N`

Forward and adjoint solution sensitivities

Implicit function theorem implies: $\frac{\partial w^{\mathrm{sol}}_{\mathrm{IPM}}}{\partial \theta}(w^\star;\tau,\theta) = \mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta)$,
with $J_\star(\cdot) := \frac{\partial r_\star}{\partial \theta}(\cdot)$, residual function $r_\star(\cdot)$

Forward and adjoint solution sensitivities

Implicit function theorem implies: $\frac{\partial w_{\text{IPM}}^{\text{sol}}}{\partial \theta}(w^\star; \tau, \theta) = \mathcal{M}_\star(w^\star; \tau, \theta)^{-1} J_\star(w^\star; \tau, \theta)$,
with $J_\star(\cdot) := \frac{\partial r_\star}{\partial \theta}(\cdot)$, residual function $r_\star(\cdot)$

Structured linear system

Coeff. matrix $\mathcal{M}_\star = \begin{bmatrix} Q_\star & G_\star^\top & H_\star^\top & 0 \\ G_\star & 0 & 0 & 0 \\ H_\star & 0 & 0 & \mathbb{1} \\ 0 & 0 & S_\star & M_\star \end{bmatrix}$ reduces to $\widetilde{\mathcal{M}}_\star = \begin{bmatrix} Q_\star + H_\star^\top S_\star^{-1} M_\star H_\star & G_\star^\top \\ G_\star & 0 \end{bmatrix}$.

Forward and adjoint solution sensitivities

Implicit function theorem implies: $\frac{\partial w_{\text{IPM}}^{\text{sol}}}{\partial \theta}(w^\star;\tau,\theta) = \mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta)$,
with $J_\star(\cdot) := \frac{\partial r_\star}{\partial \theta}(\cdot)$, residual function $r_\star(\cdot)$

Structured linear system

Coeff. matrix $\mathcal{M}_\star = \begin{bmatrix} Q_\star & G_\star^\top & H_\star^\top & 0 \\ G_\star & 0 & 0 & 0 \\ H_\star & 0 & 0 & \mathbb{1} \\ 0 & 0 & S_\star & M_\star \end{bmatrix}$ reduces to $\widetilde{\mathcal{M}}_\star = \begin{bmatrix} Q_\star + H_\star^\top S_\star^{-1} M_\star H_\star & G_\star^\top \\ G_\star & 0 \end{bmatrix}$.

Adjoint sensitivity for adjoint seed $\nu \in \mathbb{R}^{n_w}$

$$s_{\text{adj}}{}^\top := \nu^\top \frac{\partial w_{\text{IPM}}^{\text{sol}}}{\partial \theta}(w^\star;\tau,\theta) = \nu^\top \mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta).$$

Transposing both sides yields

$$s_{\text{adj}} = J_\star(w^\star;\tau,\theta)^\top \left(\mathcal{M}_\star(w^\star;\tau,\theta)^{-\top}\nu\right).$$

Forward and adjoint solution sensitivities

Implicit function theorem implies: $\frac{\partial w^{\text{sol}}_{\text{IPM}}}{\partial \theta}(w^\star;\tau,\theta) = \mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta)$,
with $J_\star(\cdot) := \frac{\partial r_\star}{\partial \theta}(\cdot)$, residual function $r_\star(\cdot)$

Structured linear system

Coeff. matrix $\mathcal{M}_\star = \begin{bmatrix} Q_\star & G_\star^\top & H_\star^\top & 0 \\ G_\star & 0 & 0 & 0 \\ H_\star & 0 & 0 & \mathbb{1} \\ 0 & 0 & S_\star & M_\star \end{bmatrix}$ reduces to $\widetilde{\mathcal{M}}_\star = \begin{bmatrix} Q_\star + H_\star^\top S_\star^{-1} M_\star H_\star & G_\star^\top \\ G_\star & 0 \end{bmatrix}$.

Adjoint sensitivity for adjoint seed $\nu \in \mathbb{R}^{n_w}$

$$s_{\text{adj}}{}^\top := \nu^\top \frac{\partial w^{\text{sol}}_{\text{IPM}}}{\partial \theta}(w^\star;\tau,\theta) = \nu^\top \mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta).$$

Transposing both sides yields

$$s_{\text{adj}} = J_\star(w^\star;\tau,\theta)^\top \left(\mathcal{M}_\star(w^\star;\tau,\theta)^{-\top} \nu\right).$$

$\Longrightarrow$ Adjoint sensitivity can be obtained with 1 backsolve instead of n_θ many.