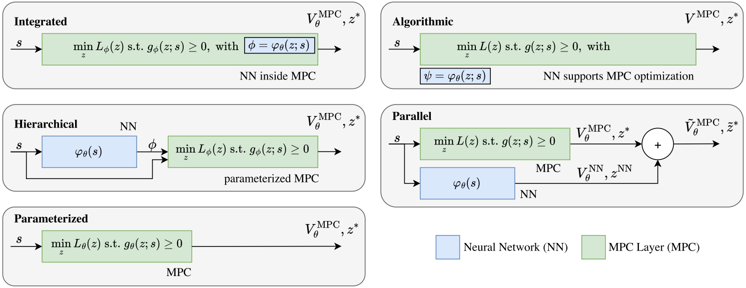
# Differentiable NMPC with acados

Katrin Baumgärtner

University of Freiburg

Predict, Control, Learn: Combining Model Predictive Control and Reinforcement Learning
Tutorial Session at ECC 2026, Reykjavík, July 2026

# Motivation




**Building blocks for practical implementations.**

- Efficient numerical solvers for optimal control problems.
- Availability of solution sensitivities with respect to problem parameters.

# Tutorial Roadmap

| | *Talk* | *Presenter* | *Duration* |
|---|---|---|---|
| ✓ | Markov Decision Processes – a unifying perspective on MPC and RL | Jasper Hoffmann | 20min |
| ✓ | Synthesis of Model Predictive Control and Reinforcement Learning | Rudolf Reiter | 30min |
| ✓ | Reinforcement Learning over MPC: Why does it work? | Dirk Reinhardt | 30min |
| ➤ | **Differentiable NMPC with `acados`** | **Katrin Baumgärtner** | **20min** |
| | MPCRL with `leap-c` | Jasper Hoffmann | 20min |

At the end of each talk there will be time for questions!

Tutorial webpage for slides and more!

# Reminder: Parameterized OCP

## Parameterized OCP with parameters $\phi$.

**OCP-structured NLP.**

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad & \bar{V}_\phi^{\mathrm{MPC}}(x_N) + \sum_{k=0}^{N-1} l_\phi^{\mathrm{MPC}}(x_k, u_k) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\phi^{\mathrm{MPC}}(x_k, u_k), \ 0 \le k < N, \\
& 0 \le h_\phi^{\mathrm{MPC}}(x_k, u_k), \ 0 \le k < N, \\
& 0 \le \tilde{h}_\phi^{\mathrm{MPC}}(x_N).
\end{aligned}
$$

Parameters $\phi$ can affect the objective $L_\phi$, the dynamics $f_\phi^{\mathrm{MPC}}$, and the constraints $h_\phi^{\mathrm{MPC}}$.

# acados – fast embedded solvers for nonlinear optimal control

open-source software package providing state-of-the-art algorithms for optimal control

- ⚡ 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

open-source software package providing state-of-the-art algorithms for optimal control

- ⚡ 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 $\implies$ embeddable

# acados – fast embedded solvers for nonlinear optimal control

acados: open-source software package providing state-of-the-art algorithms for optimal control

- 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: HPIPM, DAQP, qpOASES, qpDUNES, OSQP, Clarabel
- full & partial condensing via HPIPM

# acados – fast embedded solvers for nonlinear optimal control

acados

open-source software package providing state-of-the-art algorithms for optimal control

- 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: `HPIPM`, `DAQP`, `qpOASES`, `qpDUNES`, `OSQP`, `Clarabel`
- full & partial condensing via `HPIPM`
- NLP solvers: SQP, DDP, RTI, AS-RTI
- Robust & stochastic MPC via zoRO
- Exploit convex-over-nonlinear structures
- Customize the solver by providing a tailored Hessian or Jacobian approximation

# acados – fast embedded solvers for nonlinear optimal control

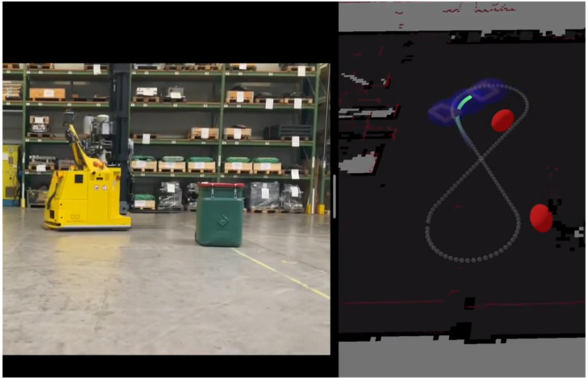
open-source software package providing state-of-the-art algorithms for optimal control

- ⚡ 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: `HPIPM`, `DAQP`, `qpOASES`, `qpDUNES`, `OSQP`, `Clarabel`
- 🤏 full & partial condensing via `HPIPM`
- 🎯 NLP solvers: SQP, DDP, RTI, AS-RTI
- 🦺 Robust & stochastic MPC via zoRO
- 🧠 Exploit convex-over-nonlinear structures
- 🎨 Customize the solver by providing a tailored Hessian or Jacobian approximation

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

# acados in practice

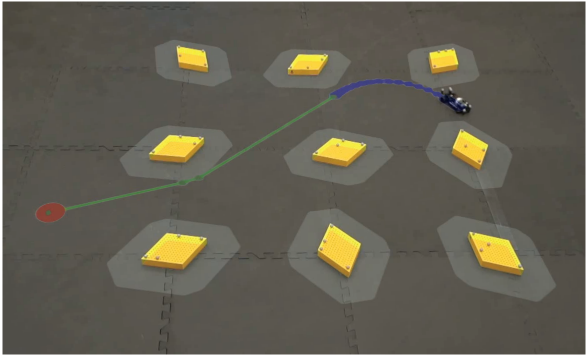
- Mobile robotics with obstacle avoidance



(Subash et al., 2024)

# acados in practice

- Mobile robotics with obstacle avoidance



(Köhler et al., 2025)

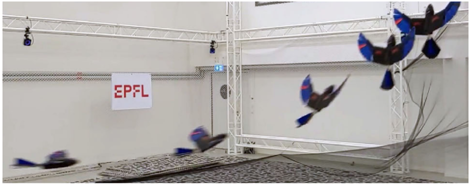
# acados in practice

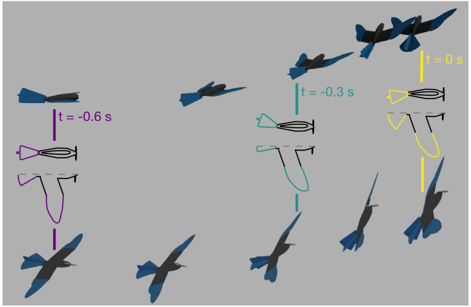
- Mobile robotics with obstacle avoidance
- Flight control for quadrotors and bird-like drones

(Romero et al., 2022)

# acados in practice

- Mobile robotics with obstacle avoidance
- Flight control for quadrotors and bird-like drones




(Wüest et al., 2024)

# acados in practice

- Mobile robotics with obstacle avoidance
- Flight control for quadrotors and bird-like drones
- Autonomous e-scooters

(Meister et al., 2025)

## Parameterized OCP with parameters $\phi$.

**OCP-structured NLP.**

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad & \bar{V}_\phi^{\mathrm{MPC}}(x_N) + \sum_{k=0}^{N-1} l_\phi^{\mathrm{MPC}}(x_k, u_k) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\phi^{\mathrm{MPC}}(x_k, u_k),\ 0 \le k < N, \\
& 0 \le h_\phi^{\mathrm{MPC}}(x_k, u_k),\ 0 \le k < N, \\
& 0 \le \tilde{h}_\phi^{\mathrm{MPC}}(x_N).
\end{aligned}
$$

**Short notation:**

$$
\begin{aligned}
z^{\mathrm{sol}}(\phi) := \arg \min_{z \in \mathbb{R}^{n_z}} \quad & f(z;\phi) \\
\text{s.t.} \quad & g(z;\phi) = 0, \\
& h(z;\phi) \le 0.
\end{aligned}
$$

# Parameterized OCP

## Parameterized OCP with parameters $\phi$.

**OCP-structured NLP.**

$$
\begin{aligned}
\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad & \bar{V}_\phi^{\mathrm{MPC}}(x_N) + \sum_{k=0}^{N-1} l_\phi^{\mathrm{MPC}}(x_k, u_k) \\
\text{s.t.} \quad & x_0 = s, \\
& x_{k+1} = f_\phi^{\mathrm{MPC}}(x_k, u_k), \; 0 \le k < N, \\
& 0 \le h_\phi^{\mathrm{MPC}}(x_k, u_k), \; 0 \le k < N, \\
& 0 \le \tilde{h}_\phi^{\mathrm{MPC}}(x_N).
\end{aligned}
$$

**Short notation:**

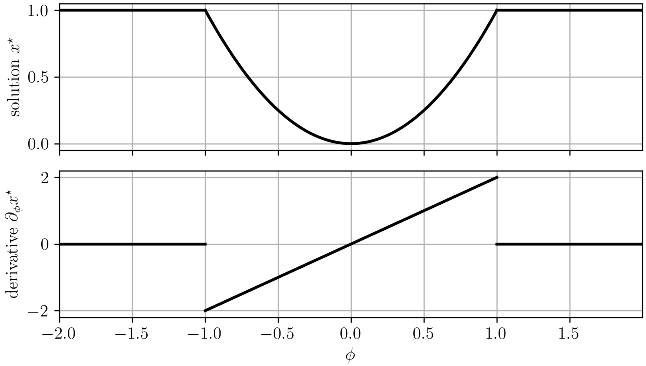
$$
\begin{aligned}
z^{\mathrm{sol}}(\phi) := \arg \min_{z \in \mathbb{R}^{n_z}} \quad & f(z;\phi) \\
\text{s.t.} \quad & g(z;\phi) = 0, \\
& h(z;\phi) \le 0.
\end{aligned}
$$

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

# Illustrative example

## Scalar example

$$\min_x \quad (x - \phi^2)^2$$
$$\text{s.t.} \quad -1 \le x \le 1.$$



# KKT conditions & smoothing

Parametric optimization problem.

$$
\begin{aligned}
z^{\mathrm{sol}}(\phi) := \arg \min_{z \in \mathbb{R}^{n_z}} \quad & f(z;\phi) \\
\text{s.t.} \quad & g(z;\phi) = 0, \\
& h(z;\phi) \leq 0.
\end{aligned}
$$

# KKT conditions & smoothing

Parametric optimization problem.

$$
\begin{aligned}
z^{\mathrm{sol}}(\phi) \coloneqq \arg \min_{z \in \mathbb{R}^{n_z}} \quad & f(z;\phi) \\
\text{s.t.} \quad & g(z;\phi) = 0, \\
& h(z;\phi) \leq 0.
\end{aligned}
$$

## KKT conditions

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

# KKT conditions & smoothing

Parametric optimization problem.

$$
\begin{aligned}
z^{\text{sol}}(\phi) := \arg \min_{z \in \mathbb{R}^{n_z}} \quad & f(z;\phi) \\
\text{s.t.} \quad & g(z;\phi) = 0, \\
& h(z;\phi) \le 0.
\end{aligned}
$$

## Interior-point smoothed KKT conditions.

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

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

# The implicit function theorem

## Regularity assumptions.

- $f, g, h$ are twice differentiable in $z$ and once differentiable in $\phi$.
- $z^\star = z^{\text{sol}}(\bar{\phi})$ is a KKT point of the parametric NLP with parameter $\bar{\phi}$ satisfying LICQ, SOSC, and strict complementarity.

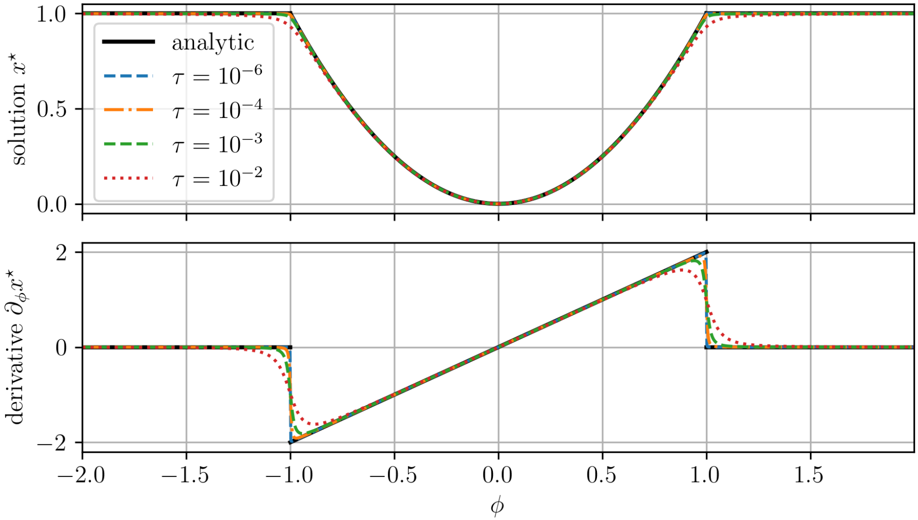
## Differentiable solution map.

If SOSC holds at $z^*$ with respect to the barrier problem with barrier parameter $\tau > 0$, the implicit function theorem implies that

1. There exists a solution of the smoothed interior-point KKT system $z_{\text{IPM}}^{\text{sol}}(\tau; \bar{\phi})$ with $\lim_{\tau \to 0^+} z_{\text{IPM}}^{\text{sol}}(\tau, \bar{\phi}) = z^\star$ and $\|z_{\text{IPM}}^{\text{sol}}(\tau; \bar{\phi}) - z^\star\| \in \mathcal{O}(\tau)$.
2. In a neighborhood of $\bar{\phi}$, there exists a differentiable function $v(\tau; \phi) = (z(\tau; \phi), \lambda(\tau; \phi), \mu(\tau; \phi))$ that corresponds to a locally unique solution of the smoothed interior-point KKT system.

Proofs are given in Pirnay, López-Negrete, and Biegler (2012); Wright (1992).

# Illustrative example



# Implementation in acados

**Nominal solution.** Smoothed interior-point KKT system is solved by iteratively solving quadratic barrier problems (barrier QPs):

1. Approximate the NLP by a QP.
2. Solve the smoothed interior-point KKT system associated with the QP with an interior-point solver for $\tau \to \bar{\tau}$ with $\bar{\tau} > 0$.

Equivalent to SQP with interior-point QP solver if $\bar{\tau} = 0$.

# Implementation in acados

**Nominal solution.** Smoothed interior-point KKT system is solved by iteratively solving quadratic barrier problems (barrier QPs):

1. Approximate the NLP by a QP.
2. Solve the smoothed interior-point KKT system associated with the QP with an interior-point solver for $\tau \to \bar{\tau}$ with $\bar{\tau} > 0$.

Equivalent to SQP with interior-point QP solver if $\bar{\tau} = 0$.

**Solution sensitivities.** A linear system with the same matrix as for the nominal solution needs to be solved.

⏩ Reuse efficient implementation via Riccati recursion provided by `HPIPM` (Frison & Diehl, 2020) exploiting the OCP structure.

# Hessian approximations & two-solver approach

**Hessian approximations in SQP.**

- (generalized and extended) Gauss-Newton Hessian
  (Messerer et al., 2021; Baumgärtner & Diehl, 2022)
- regularization techniques such as project, mirror, Levenberg-Marquardt, convexify
  (Verschueren et al., 2017)

Benefits are reduced computational complexity and convex QP subproblems.

# Hessian approximations & two-solver approach

**Hessian approximations in SQP.**

- (generalized and extended) Gauss-Newton Hessian
  (Messerer et al., 2021; Baumgärtner & Diehl, 2022)
- regularization techniques such as project, mirror, Levenberg-Marquardt, convexify
  (Verschueren et al., 2017)

Benefits are reduced computational complexity and convex QP subproblems.

⚠️ solution sensitivities require exact Hessian ⚠️

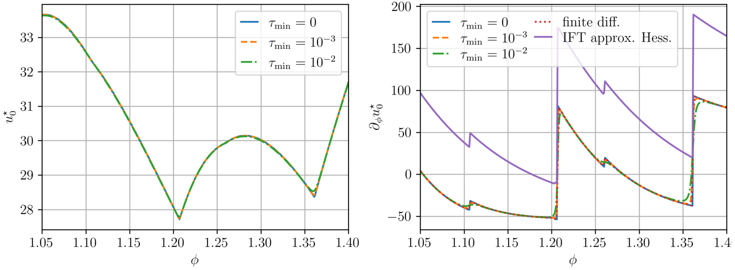
# Hessian approximations & two-solver approach

**Hessian approximations in SQP.**

- (generalized and extended) Gauss-Newton Hessian
  (Messerer et al., 2021; Baumgärtner & Diehl, 2022)
- regularization techniques such as project, mirror, Levenberg-Marquardt, convexify
  (Verschueren et al., 2017)

Benefits are reduced computational complexity and convex QP subproblems.

⚠️ solution sensitivities require exact Hessian ⚠️



# Hessian approximations & two-solver approach

**Hessian approximations in SQP.**

- (generalized and extended) Gauss-Newton Hessian (Messerer et al., 2021; Baumgärtner & Diehl, 2022)
- regularization techniques such as project, mirror, Levenberg-Marquardt, convexify (Verschueren et al., 2017)

Benefits are reduced computational complexity and convex QP subproblems.

⚠️ solution sensitivities require exact Hessian ⚠️ → two-solver approach

**🤸 Nominal solver**

Solve nominal problem using any algorithm.

- Hessian approximation
- regularization
- ...

**🧮 Sensitivity solver**

- Evaluate exact Hessian.
- Compute solution sensitivity via efficient linear solve based on the Riccati recursion provided by `HPIPM`.

# Forward vs. adjoint sensitivities

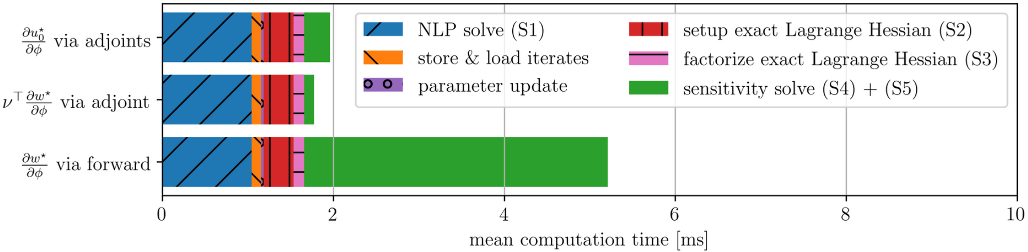



Figure: Computation times for a nominal solve plus forward or adjoint sensitivities. In this example, $n_x = 9$, $n_u = 3$, $n_\phi = 113$, $N = 40$

**Forward sensitivity.** Compute full Jacobian $\frac{\partial w^\star}{\partial \phi} \in \mathbb{R}^{n_w \times n_\phi}$.

**Adjoint sensitivity.** Compute single adjoint direction $\nu^\top \frac{\partial w^\star}{\partial \phi} \in \mathbb{R}^{1 \times n_\phi}$. $\leftarrow$ backpropagation

Here, $w^\star = (z^\star, \lambda^\star, \mu^\star)$.

# Benchmark Results

Table: Timings in [ms] for solving $n_{\text{batch}} = 128$ bounded LQR problems with $N = 20$, $n_x = 8$, $n_u = 4$, $n_\phi = 248$. 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 ($\times 160$) |

Remark: `mpc.pytorch` is not faster on GPU than CPU (both in our experiments and in Adabag et al. (2025))

# Summary

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

# Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation tailored to the OCP structure

# Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation tailored to the OCP structure
- 🦉 Beware of inexact Hessians!

# Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation tailored to the OCP structure
- 🦉 Beware of inexact Hessians!
- ⏩ Adjoint solution sensitivities for efficient backward pass

# Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation tailored to the OCP structure
- 🦉 Beware of inexact Hessians!
- ⏩ Adjoint solution sensitivities for efficient backward pass
- 🕯️ Wrapped in a differentiable layer in `leap-c` → upcoming talk

# Summary

- 🔮 Smoothed interior-point KKT conditions to differentiate across active-set changes
- ⚡ Fast implementation tailored to the OCP structure
- 🦉 Beware of inexact Hessians!
- ⏩ Adjoint solution sensitivities for efficient backward pass
- 🕯️ Wrapped in a differentiable layer in `leap-c` → upcoming talk

**Thanks for your attention!** 🙏
Looking forward to questions, discussions and collaborations!

**Preprint.** Jonathan Frey, Katrin Baumgärtner, Gianluca Frison, Dirk Reinhardt, Jasper Hoffmann, Leonard Fichtner, Sebastien Gros, Moritz Diehl: *Differentiable Nonlinear Model Predictive Control* (arxiv link)

**Code for figures and benchmark.** GitHub repository

⭐ Give us a star on GitHub: `github.com/acados/acados` ⭐

# Related work: no other differentiable solver tailored to OCPs with inequality support

| Software / publication | OCP-tailored | adjoints | ineq. support | Nonlinearity in cost & constraints | exact results | parallelization | permissive license | |
|---|---|---|---|---|---|---|---|---|
| `acados` | ✅ | ✅ | ✅ | ✅ | ✅ | `OpenMP` | ✅ | BSD-2 |
| `mpc.pytorch` (Amos et al., 2018) | ✅ | ✅ | ❌[a] | ❌ | ❌ | `PyTorch` | ✅ | MIT |
| `cvxpygen` + `cvxpylayers` (Schaller et al., 2022) (Schaller & Boyd, 2025) | ❌ | ✅ | ✅ | ⚪[b] | ✅ | `Python` multithreading | ✅ | Apache 2.0 |
| `diffmpc` (Toyota) (Adabag et al., 2025) | ✅ | ✅ | ❌ | ✅ | ❌ | `JAX` | ❌ | CC BY-NC-ND 4.0 |
| `trajax` (Google) no paper | ✅ | ✅ | ❌ | ✅ | ⚪[c] | `JAX` | ✅ | Apache 2.0 |
| `theseus` (Facebook) (Pineda et al., 2022) | ❌ | ✅ | ❌ | ❌ | ❌ | `PyTorch` | ✅ | MIT |
| `sIPOPT` (Pirnay et al., 2012)[d] | ❌ | ❌ | ✅ | ✅ | ✅ | – | ✅ | EPL |
| `CasADi` native (Andersson & Rawlings, 2018) | ❌ | ✅ | ✅ | ✅ | ✅ | – | ✅ | LGPL |

[a] while simple control bounds can be added, experiments revealed that the solver fails to converge if active set is nontrivial.
[b] only convex cost and constraints.
[c] a set of options seems to exist that allow using the exact Hessian. The documentation does not indicate in which case the sensitivities are accurate.
[d] functionality extremely limited in practice. Not usable in `CasADi` and abandoned rewrite effort in `https://github.com/athrpf/sipopt2`.

# References I

Adabag, E., Greiff, M., Subosits, J., & Lew, T. (2025). *Differentiable model predictive control on the GPU.* Retrieved from https://arxiv.org/abs/2510.06179

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., & Rawlings, J. B. (2018). Sensitivity analysis for nonlinear programming in CasADi. In *Proceedings of the ifac conference on nonlinear model predictive control (nmpc).*

Baumgärtner, K., & Diehl, M. (2022). The extended Gauss-Newton method for nonconvex loss functions and its application to time-optimal model predictive control. In *Proceedings of the american control conference (acc).*

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

Köhler, J., Zhang, D., Soloperto, R., Carron, A., & Zeilinger, M. (2025). *An mpc framework for efficient navigation of mobile robots in cluttered environments.* Retrieved from https://arxiv.org/abs/2509.15917

# References II

Meister, D., Strässer, R., Brändle, F., Seidel, M., Bassler, B., Gerber, N., … Allgöwer, F. (2025). Path-following model predictive control for autonomous e-scooters. *arXiv preprint arXiv:2505.05314*.

Messerer, F., Baumgärtner, K., & Diehl, M. (2021). Survey of sequential convex programming and generalized Gauss-Newton methods. *ESAIM: Proceedings and Surveys*, *71*, 64–88.

Pineda, L., Fan, T., Monge, M., Venkataraman, S., Sodhi, P., Chen, R. T., … others (2022). Theseus: A library for differentiable nonlinear optimization. *Advances in Neural Information Processing Systems*, *35*, 3801–3818.

Pirnay, H., López-Negrete, R., & Biegler, L. T. (2012). Optimal sensitivity based on IPOPT. *Mathematical Programming Computation*, *4*(4), 307–331. Retrieved from `https://doi.org/10.1007/s12532-012-0043-2` doi: 10.1007/s12532-012-0043-2

Romero, A., Penicka, R., & Scaramuzza, D. (2022). Time-optimal online replanning for agile quadrotor flight. *IEEE Robotics and Automation Letters*, *7*(3), 7730–7737.

Schaller, M., Banjac, G., Diamond, S., Agrawal, A., Stellato, B., & Boyd, S. (2022). Embedded code generation with cvxpy. *IEEE Control Systems Letters*, *6*, 2653–2658.

# References III


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`

Subash, A., Kloeser, D., Frey, J., Reiter, R., Diehl, M., & Bohlmann, K. (2024). Model predictive control for frenet-cartesian trajectory tracking of a tricycle kinematic automated guided vehicle. In *2024 ieee/rsj international conference on intelligent robots and systems (iros)* (p. 11948-11953). doi: 10.1109/IROS58592.2024.10802822

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.

Wright, M. H. (1992). Interior methods for constrained optimization. *Acta numerica*, *1*, 341–407.

Wüest, V., Jeger, S., Feroskhan, M., Ajanic, E., Bergonti, F., & Floreano, D. (2024). Agile perching maneuvers in birds and morphing-wing drones. *Nature Communications*, *15*(1), 8330.