# Differentiable Nonlinear Model Predictive Control

**Jonathan Frey**[1], **Katrin Baumgärtner**[1], **Gianluca Frison**[1], **Dirk Reinhardt**[2], **Jasper Hoffmann**[1], **Leonard Fichtner**[1], **Joschka Boedecker**[1], **Sebastien Gros**[2], **Moritz Diehl**[1]

[1]University of Freiburg [2]NTNU Trondheim

## Abstract

The efficient computation of parametric solution sensitivities is a key challenge in the integration of learning-enhanced methods with nonlinear model predictive control, as their availability is crucial for many learning algorithms. This paper discusses the computation of solution sensitivities of general nonlinear programs using the implicit function theorem and smoothed optimality conditions treated in interior-point methods (IPM). We introduce sequential barrier quadratic programming (SBQP), which generalizes SQP with an IPM quadratic programming (QP) solver by allowing the barrier parameter to approach a prescribed nonnegative target value, recovering classical SQP when this target is zero. Previous works are limited to convex or unconstrained formulations, or lack an implementation for efficient sensitivity evaluation. The publication is accompanied by an efficient open-source implementation within the `acados` framework, providing both forward and adjoint sensitivities for general optimal control problems, achieving speedups exceeding 3x over the state-of-the-art solvers `mpc.pytorch` and `cvxpygen`.

## 1 Introduction

In recent years, great research efforts have been made to combine the paradigms of learning and optimal control – in particular nonlinear model predictive control (MPC) – exploiting their complementary advantages. The outcomes of these efforts cover various fields: from MPC with learned system models to imitation learning and behavior cloning to MPC-based Reinforcement Learning (RL) (Görges, 2017; Hewing et al., 2020; Reiter et al., 2026b). In particular, the approach of performing learning-enhanced adaptations of MPC schemes is very general and promises to deliver excellent performance while attaining desirable properties of MPC schemes, such as (robust) constraint satisfaction and stability (Gros & Zanon, 2020, 2022; Kordabad et al., 2022; Kordabad & Gros, 2022; Kordabad et al., 2024). The approach has been successfully employed for a wide range of applications such as car racing (Reiter et al., 2026a; Jahncke et al., 2026), quadcopter control (Romero et al., 2024), energy-management systems in residential buildings (Cai et al., 2023b), microgrid operation for demand response (Cai et al., 2023a), and greenhouse climate control (Mallick et al., 2025). A wider adoption of this approach in control practice has been limited by the availability of dedicated software and, in particular, efficient solvers that provide parametric solution sensitivities – a crucial prerequisite for a variety of learning algorithms (Reinhardt et al., 2024). Inherent constraint satisfaction is a key advantage of learning-enhanced MPC over purely data-driven approaches. However, existing differentiable MPC implementations lack support for inequality constraints (cf. Table 2), which has so far prevented real-world applications from benefiting from this advantage (Rickenbach et al., 2025). Other techniques that can profit from OCP solution sensitivities include Sobolev learning for approximate MPC (Lüken et al., 2023) and differentiable robust MPC (Oshin et al., 2024).

As nonlinear MPC heavily relies on the numerical solution of nonlinear optimization problems in real time, tailored algorithms and efficient implementations have been a major research focus (Frison & Jørgensen, 2013; Vanroye et al., 2023; Wang & Boyd, 2010; Zavala & Biegler, 2009; Ferreau et al., 2014; Frey et al., 2020; Stellato et al., 2020; Arnström et al., 2022; Goulart & Chen, 2024; Ferreau et al., 2017; Houska et al., 2011; Zanelli et al., 2017). In particular, the open-source software

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

framework acados (Verschueren et al., 2021) implements many of these algorithms. The numerical solvers provided by acados are tailored to the problem structure specific to optimal control, thus achieving exceptional computational efficiency, while keeping the problem formulation as general as possible, covering general nonlinear and parametric costs, dynamics and inequality constraints, which can be different at each stage of the problem (Frey et al., 2024a).

The focus of this paper is the efficient computation of parametric solution sensitivities in the context of optimal control, as their availability – and efficient computation – is a key building block for a successful integration of learning-enhanced methods and optimization-based control.

**Contribution.** The contribution of this paper is threefold: First, this work discusses the efficient computation of parametric (adjoint) solution sensitivities for general nonlinear programs (NLP) via the implicit function theorem (IFT). In particular, this work highlights common pitfalls of the IFT and discusses their mitigation. Previous work in the machine learning (ML) literature was subject to this pitfall, or limited to convex or unconstrained nonlinear least squares problems. Second, we introduce SBQP, which generalizes SQP with an IPM QP solver by allowing the barrier parameter to approach a prescribed nonnegative target value. We detail how IPMs or SBQP can be used to obtain a differentiable approximation of the solution map and its sensitivities suitable for gradient-based learning algorithms by leveraging the smoothing of the optimality conditions that these methods employ. Finally, we present an efficient implementation of a differentiable solver for optimal control structured NLPs within the open-source software package acados which provides both forward and adjoint sensitivities. In particular, the optimal control problem (OCP) formulation covers general, potentially nonlinear, costs and constraints on both states and controls while allowing all problem functions, i.e., costs, constraints and dynamics, to be fully parametric. A comparison with state-of-the-art solvers demonstrates speedup factors greater than three on a CPU.

**Related work.** One approach to analyze and compute parametric solution sensitivities of general nonlinear programs is to apply the IFT (Dini, 1907) to the necessary conditions of optimality, also called Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Nocedal & Wright, 2006). An alternative approach differentiates through the iterations of an optimization algorithm (Tamar et al., 2017). In the following, we adopt the former approach, which offers superior memory efficiency and computational complexity compared to differentiating through the iterations of an optimization solver, as demonstrated in the context of MPC by Amos et al. (2018) using iLQR.

A qualitative comparison of available software tools for differentiable MPC is given in Table 2 and important relations are summarized next. Pirnay et al. (2012) details solution sensitivity computation via the smoothed KKT system tackled in an IPM, providing useful theoretical results and the implementation sIPOPT. This implementation builds on the popular NLP solver IPOPT (Wächter & Biegler, 2006), but requires defining parameters as variables and does not offer adjoint sensitivities. In Andersson & Rawlings (2018), the computation of forward and adjoint solution sensitivities based on an active-set QP solver was presented. More recently, the analysis of parametric solution sensitivities has received a lot of attention in the ML community. Amos & Kolter (2017) suggested embedding the solution of an optimization problem in a neural network, but their derivations and implementation are limited to quadratic programs (QPs). In Lee et al. (2019), the differentiable QP solver from Amos & Kolter (2017) was leveraged in the context of meta-learning. In Agrawal et al. (2019), neural network architectures comprising so-called differentiable convex optimization layers are introduced. These layers contain an optimization problem formulated via disciplined parametrized programming, closely related to disciplined convex programming (Grant et al., 2006) and are implemented in cvxpylayers. Recently, cvxpygen (Schaller et al., 2022) was extended to support solution sensitivities which showed significant speedups compared to the implementation in cvxpylayers (Schaller & Boyd, 2026). The software package Theseus provides solution sensitivities for nonlinear least-squares optimization problems (Pineda et al., 2022), but is limited to soft constraints. Mandi & Guns (2020) solve linear programs with an interior-point approach and stop the barrier parameter at a certain threshold to differentiate the solution of a smoothed KKT system. However, their work does not consider an efficient backward pass via adjoint solution sensitivities. In contrast to implicit smoothing via the IPM, the nondifferentiability of the solution map might be directly tackled using a nonsmooth variant of the IFT which is based on the assumption of path differentiability and the concept of conservative Jacobians (Davis et al., 2020; Bolte et al., 2021).

While the works mentioned above consider general, i.e. structure-agnostic, solvers, Amos et al. (2018) first considered differentiable solution maps within the context of structure-exploiting solvers tailored to OCPs coining the term *differentiable MPC*. However, their approach is restricted to problems with quadratic cost functions and simple bounds on the controls and only covers parametric

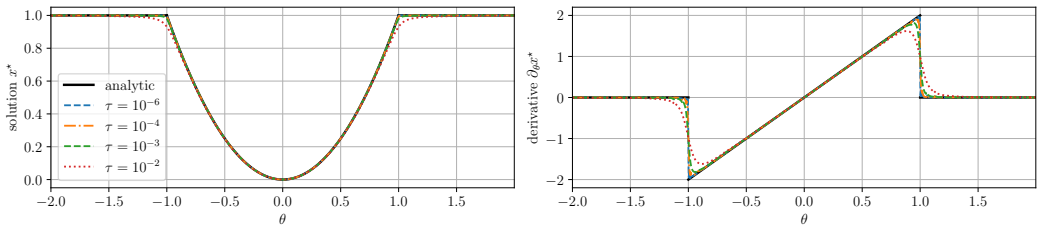


Figure 1: Simple example visualizing a nonsmooth solution map, its derivative and some smooth approximations.

nonlinear dynamics and parametric quadratic costs. Adabag et al. (2026) present an implementation of differentiable MPC for GPU, which tackles a more restrictive formulation with no inequalities and a cost function that is decoupled in controls and states. The papers by Amos et al. (2018); Adabag et al. (2026) also suggest the computation of solution sensitivities via the KKT system of a convex approximation of the problem, an approach which might yield highly degraded sensitivity results as found in Dinev et al. (2022) and shown in the present paper, see Remark 3 and Example 2. Jin et al. (2020) suggest computing solution sensitivities via Pontryagin's Maximum Principle, but the approach limited to unconstrained optimal control. Bounou et al. (2023) suggest a proximal solution approach to obtain sensitivities for OCPs. All works cited in this paragraph lack the ability to handle inequality constraints which are parametric, nonlinear or include the states. In contrast, Zuliani et al. (2025a) use the nonsmooth IFT variant to compute sensitivities in an MPC context considering convex quadratic programs, but do not provide an efficient implementation exploiting the OCP structure.

## 2 Preliminaries

This section briefly reviews the mathematical foundations and formal requirements underlying the sensitivity analysis of general parametric nonlinear programs (NLP) via the implicit function theorem (IFT), stated in A.2. Before diving into the mathematical details, we introduce a tutorial example which illustrates nondifferentiability of the solution map – a property that might pose major challenges in practice, in particular in the context of gradient-based learning.

**Example 1.** We regard the following scalar parametric optimization problem:

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

The solution map and its sensitivity are given by

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

Figure 1 visualizes the solution map as well as its derivative and their smooth approximations, which will be explained later in Section 2.2.

### 2.1 Solving nonlinear programs with an interior-point based SQP method

We are interested in differentiating the solution of a nonlinear program of the form

$$z^{\text{sol}}(\theta) \coloneqq \arg\min_{z\in\mathbb{R}^{n_z}} f(z;\theta) \ \text{ s. t. } \ g(z;\theta)=0, \ h(z;\theta)\le 0. \tag{2}$$

We assume that the objective function $f : \mathbb{R}^{n_z}\times\mathbb{R}^{n_\theta}\to\mathbb{R}$ and the constraint functions $g : \mathbb{R}^{n_z}\times\mathbb{R}^{n_\theta}\to\mathbb{R}^{n_g}$, $h : \mathbb{R}^{n_z}\times\mathbb{R}^{n_\theta}\to\mathbb{R}^{n_h}$ are twice continuously differentiable jointly in $(z,\theta)$. Introducing the Lagrange multipliers $\lambda\in\mathbb{R}^{n_g}$ and $\mu\in\mathbb{R}^{n_h}$, the Lagrangian function for (2) is given by

$$\mathcal{L}(z,\lambda,\mu;\theta) = f(z;\theta) + \lambda^\top g(z;\theta) + \mu^\top h(z;\theta). \tag{3}$$

The Karush-Kuhn-Tucker (KKT) conditions of NLP (2) can be written as

$$\nabla_z f(z;\theta) + \nabla_z g(z;\theta)\lambda + \nabla_z h(z;\theta)\mu = 0, \tag{4a}$$
$$g(z;\theta) = 0, \tag{4b}$$
$$h(z;\theta) \le 0, \tag{4c}$$
$$\mu \ge 0, \tag{4d}$$
$$\mu_i h_i(z;\theta) = 0, \ i=1,\dots,n_h. \tag{4e}$$

If there exists $\lambda^\star, \mu^\star$ such that $v^\star = (z^\star, \lambda^\star, \mu^\star)$ is a solution of (4), $v^\star$ is called a KKT point. A KKT point is a local minimizer of (2) if the following assumption holds, cf. Nocedal & Wright (2006).

**Assumption 1.** Let $v^\star$ be a KKT point of the NLP (2) satisfying linear independence constraint qualification (LICQ), second-order sufficient conditions of optimality (SOSC) and strict complementarity.

**Sequential quadratic programming.** Sequential quadratic programming (SQP) is an efficient solution method for solving NLPs of the form (2). In the context of nonlinear MPC, SQP is particularly popular due to its warm-starting capabilities and real-time variants (Diehl et al., 2005; Nurkanović et al., 2019; Frey et al., 2024b). For a given primal-dual initialization $(z_0, \lambda_0, \mu_0)$ and using the compact notation $v_k = (z_k, \lambda_k, \mu_k)$, an SQP method proceeds by successively solving quadratic programs (QPs) of the form

$$\Delta z^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v_k) \coloneqq \ \arg\min_{\Delta z} \ q_k^\top \Delta z + \tfrac{1}{2}\Delta z^\top Q_k \Delta z \quad \text{s.t.} \quad g_k + G_k \Delta z = 0,\ h_k + H_k \Delta z \leq 0, \tag{5}$$

where $Q_k$ is an approximation of the Hessian of the Lagrangian $\nabla^2_{zz}\mathcal{L}(z_k, \lambda_k, \mu_k; \theta)$, the vector $q_k = \nabla_z f(z_k; \theta)$ is the objective gradient, $g_k = g(z_k; \theta)$ and $h_k = h(z_k; \theta)$ are the constraint evaluations and $G_k = \nabla g(z_k; \theta)^\top$, $H_k = \nabla h(z_k; \theta)^\top$ the Jacobians of the constraints. The full-step SQP method updates its iterates by

$$z_{k+1} = z_k + \Delta z^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v_k), \quad \lambda_{k+1} = \lambda^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v_k), \quad \mu_{k+1} = \mu^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v_k), \tag{6}$$

where $\mu^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v_k)$, $\lambda^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v_k)$ denote the dual solution of QP (5). The QP subproblems can be tackled with different solution strategies, which can be classified into active-set methods, interior-point methods and first-order methods. This paper focuses on interior-point methods for solving the QPs, which are detailed in the following.

**Remark 1 (Hessian approximations).** Many SQP approaches use an approximate Hessian $Q_k$ instead of the exact Hessian of the Lagrangian. A popular class of Hessian approximations are the Gauss-Newton Hessian approximation and its generalizations. Their advantages are that they are inherently positive-semidefinite, and represent outer convex structure in the subproblems which leads to beneficial convergence properties (Bock et al., 2007; Messerer et al., 2021; Baumgärtner & Diehl, 2022; Baumgärtner et al., 2023). Additionally, these Hessian approximations are cheaper to compute compared to the exact Hessian. In the context of unconstrained optimal control, the Gauss-Newton Hessian is used, for instance by the popular iLQR method (Li & Todorov, 2004; Todorov & Li, 2005; Amos et al., 2018).

**QP solution via interior-point methods.** Within an interior-point QP solver, the following system of smoothed KKT conditions of QP (5) is solved:

$$\begin{aligned}
Q_k \Delta z + q_k + G_k^\top \lambda + H_k^\top \mu &= 0, && \text{(7a)} & \qquad \mu_i s_i &= \tau,\ \ i = 1, \ldots, n_h, && \text{(7d)}\\
g_k + G_k \Delta z &= 0, && \text{(7b)} & \qquad s, \mu &> 0, && \text{(7e)}\\
h_k + H_k \Delta z + s &= 0, && \text{(7c)} &&&&
\end{aligned}$$

where the slack vector $s \in \mathbb{R}^{n_h}$ and the barrier parameter $\tau > 0$ are introduced. This system of equations is solved repeatedly for values of $\tau$ that approach zero from above. In an IPM, the positivity conditions (7e) are enforced with strict inequality, typically by applying some fraction to the boundary rule. An IPM applies Newton steps to solve the system of equations consisting of (7a) - (7d). The linear system solved in an interior-point iteration $j$ with barrier parameter $\tau_j$ can be written as

$$\underbrace{\begin{bmatrix} Q_k & G_k^\top & H_k^\top & 0 \\ G_k & 0 & 0 & 0 \\ H_k & 0 & 0 & \mathbb{1} \\ 0 & 0 & S_j & M_j \end{bmatrix}}_{=:\mathcal{M}_k(s_j, \mu_j; \theta)} \begin{bmatrix} \Delta\hat{z} \\ \Delta\hat{\lambda} \\ \Delta\hat{\mu} \\ \Delta\hat{s} \end{bmatrix} = \ - \underbrace{\begin{bmatrix} \hat{q}_j \\ \hat{g}_j \\ \hat{h}_j \\ \hat{m}_j \end{bmatrix}}_{=:r_k(\Delta z_j, \lambda_j, \mu_j, s_j; \tau_j, \theta)} \tag{8}$$

with diagonal matrices $S_j = \mathrm{diag}(s_j)$ and $M_j = \mathrm{diag}(\mu_j)$ and residuals

$$\begin{aligned}
\hat{q}_j &= Q_k \Delta z_j + q_k + G_k^\top \lambda_j + H_k^\top \mu_j, \ \ \hat{g}_j = G_k \Delta z_j + g_k,\\
\hat{h}_j &= H_k \Delta z_j + h_k + s_j, \ \ \hat{m}_j = (\mu_{j,i} s_{j,i} - \tau_j)_{i=1,\ldots,n_h},
\end{aligned}$$

where $(\Delta z_j, \lambda_j, \mu_j, s_j)$ denotes the current iterate of the IPM. The linear system (8) can be reduced by eliminating $\Delta\hat{s}$ and $\Delta\hat{\mu}$ to arrive at a lower dimensional system with symmetric coefficient matrix

$$\widetilde{\mathcal{M}}_k(s_j, \mu_j; \theta) = \begin{bmatrix} Q_k + H_k^\top S_j^{-1} M_j H_k & G_k^\top \\ G_k & 0 \end{bmatrix}. \tag{9}$$

Eliminating $\Delta\hat{s}$ and $\Delta\hat{\mu}$ in this way is especially beneficial in the context of OCPs, as the reduced system can be solved very efficiently with a Riccati factorization, see Sec. 3. Invertibility of $\mathcal{M}_k$ follows from Assumption 1 for sufficiently small $\tau$, see Thm. 2, (ii).

**Sequential barrier quadratic programming (SBQP).** Next, we detail SBQP, which generalizes SQP with an interior-point QP solver by allowing the barrier parameter to approach a prescribed target $\tau_{\min} \geq 0$ rather than zero, such that classical SQP is recovered when $\tau_{\min} = 0$. When $\tau_{\min} > 0$, each SBQP iteration corresponds to solving the KKT conditions of QP (5), with the inequalities replaced by logarithmic barrier terms

$$\begin{aligned} \min_{\Delta z, s} \; & q_k^\top \Delta z + \tfrac{1}{2}\Delta z^\top Q_k \Delta z - \tau_{\min} \sum_{i=1}^{n_h} \log(s_i) \;\; \text{s.t.} \;\; g_k + G_k \Delta z = 0, \\ & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad h_k + H_k \Delta z + s = 0. \end{aligned} \tag{10}$$

We refer to (10) as a *barrier QP* and denote its solution map by $(\Delta z^{\text{sol}}_{\text{BQP}}(\tau_{\min}; \theta, v_k), s^{\text{sol}}_{\text{BQP}}(\tau_{\min}; \theta, v_k))$. In an IPM QP solver, the barrier parameter is iteratively reduced until a prescribed tolerance on the KKT residuals is met. In SBQP, instead, we let $\tau_j$ converge to the target value $\tau_{\min}$ from above and check the IPM residuals (7) with $\tau = \tau_{\min}$. For $\tau_{\min} > 0$, SBQP corresponds to solving the nonlinear smoothed interior-point KKT system (20), which consists of (4a)-(4d) and instead of the complementarity condition (4e), it contains

$$\mu_i h_i(z; \theta) = -\tau_{\min}, \; i = 1, \ldots, n_h. \tag{11}$$

Note that the termination criterion of the SBQP solver needs to be adjusted accordingly with respect to SQP. This allows us to obtain smoothed approximations of the solution map with valuable properties, see Thm. 2. These approximations are visualized for different $\tau_{\min}$ values in Fig. 1 and Fig. 2. Appendix A.3 provides more details on the algorithm presented in this section.

### 2.2 Parametric solution sensitivities within SBQP

This section discusses the computation of (approximate) parametric solution sensitivities of NLP (2) within SBQP. On the theoretical side, two levels need to be considered: First, we regard the QP (5) and review the conditions under which its solution sensitivities match those of the NLP. Second, we review the conditions under which the smoothed KKT system (8) yields an approximate solution to the QP and discuss its parametric sensitivities.

**Theorem 1 (NLP solution sensitivity existence).** Suppose Assumption 1 holds at a KKT point $z^\star$ of (2) with parameter $\bar{\theta}$. In a neighborhood of $\bar{\theta}$, there exists a differentiable function $z^{\text{sol}}(\theta)$ with $z^{\text{sol}}(\bar{\theta}) = z^\star$ that corresponds to a locally unique solution of (2).

Theorem 1 is proved via the IFT in A.4. Next, we consider the smoothed KKT system (20) as solved within an IPM or SBQP. The following theorem shows how the solution map of NLP (2) can be obtained as the solution to the smoothed KKT system as $\tau \to 0$.

**Theorem 2 (Smoothed KKT system).** Suppose Ass. 1 holds at a KKT point $z^\star$ of NLP (2) with multipliers $\lambda^\star$, $\mu^\star$. Then, for all sufficiently small $\tau := \tau_{\min} > 0$, the following holds:

(i) There exists a solution of the smoothed interior-point KKT system $z^{\text{sol}}_{\text{IPM}}(\tau; \bar{\theta})$ which is a continuously differentiable function with $\lim_{\tau \to 0^+} z^{\text{sol}}_{\text{IPM}}(\tau; \bar{\theta}) = z^{\text{sol}}(\bar{\theta})$ and $\left\| z^{\text{sol}}_{\text{IPM}}(\tau; \bar{\theta}) - z^\star \right\| \in \mathcal{O}(\tau)$.

(ii) 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 (20) and $v(0; \bar{\theta}) := \lim_{\tau \to 0^+} v(\tau; \bar{\theta}) = (z^\star, \lambda^\star, \mu^\star)$ holds.

Theorem 2 can be deduced from Prop. 1 and 3 in Pirnay et al. (2012). A comprehensive proof is accompanying Theorem 8 in Wright (1992). The next theorem states that the solution sensitivities of the smoothed interior-point KKT systems match the ones of the SBQP subproblem at convergence.

**Theorem 3 (Barrier NLP and barrier QP sensitivities coincide).** Fix $\tau > 0$ and a parameter $\bar{\theta}$, and let $w^\star = (z^\star, s^\star, \lambda^\star, \mu^\star)$ be a KKT point of the barrier NLP (21) with $v^\star := (z^\star, \lambda^\star, \mu^\star)$;

assume LICQ and SOSC hold at $w^\star$. Regard the barrier QP (10) obtained at the KKT point, and assume the exact Hessian $Q_\star = \nabla^2_{zz}\mathcal{L}(z^\star, \lambda^\star, \mu^\star; \bar{\theta})$ is used. Denote the solution map of the barrier NLP as $(z^{\text{sol}}_{\text{IPM}}, s^{\text{sol}}_{\text{IPM}})$ and the solution map of the barrier QP as $(\Delta z^{\text{sol}}_{\text{BQP}}(\tau;\theta,v^\star), s^{\text{sol}}_{\text{BQP}}(\tau;\theta,v^\star))$. Then, the solution maps $z^{\text{sol}}_{\text{IPM}}(\tau;\theta)$ and $z^\star + \Delta z^{\text{sol}}_{\text{BQP}}(\tau;\theta,v^\star)$, and their sensitivities, $\frac{\partial z^{\text{sol}}_{\text{IPM}}}{\partial\theta}(\tau;\theta)$ and $\frac{\partial \Delta z^{\text{sol}}_{\text{BQP}}}{\partial\theta}(\tau;\theta,v^\star)$ coincide at $\bar{\theta}$. The same holds for the solution maps of the Lagrange multipliers. The proof is given in Appendix A.5. Similarly, it is shown in Appendix A.6 that the QP sensitivities match the ones of the NLP, which corresponds to the proposed method for $\tau_{\min} = 0$, i.e. when SBQP corresponds to SQP.

Solutions to the smoothed KKT systems can be of particular interest in a learning context. First, their smoothness can be beneficial, as shown in an RL context in Kordabad et al. (2021). Second, their computation can be computationally cheaper, as less interior point iterations might be needed to shrink the barrier parameter.

**Remark 2 (Differentiating across active set changes).** If Ass. 1 is not satisfied, the solution map might be nondifferentiable, as in Example 1, or even exhibit jumps, as shown by the example in A.8. In case that only strict complementarity is violated in Ass. 1, but strong SOSC holds, the proposed method provides the correct sensitivities of the smoothed solution map, as shown in A.7.

**Forward and adjoint sensitivity computation.** In the remainder of this section, we use the compact notation $w \coloneqq (z, \lambda, \mu, s) \in \mathbb{R}^{n_w}$ with $n_w = n_z + n_g + 2n_h$ and assume that SBQP has converged to a solution $w^\star$ of the barrier KKT system (20) for a fixed $\tau > 0$. We assume the conditions of Theorem 3, under which the barrier QP sensitivities coincide at $\bar{\theta}$ with those of the smoothed NLP. We compute them from the barrier QP KKT system but denote them by the sensitivities of $w^{\text{sol}}_{\text{IPM}}$, the quantity of interest. Applying the IFT to the barrier QP KKT system gives

$$\frac{\partial w^{\text{sol}}_{\text{IPM}}}{\partial\theta}(w^\star;\tau,\theta) = -\mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta), \tag{12}$$

where we introduced $J_\star(\cdot) \coloneqq \frac{\partial r_\star}{\partial\theta}(\cdot)$ via the function $r_k$ defined in (8), but use the index $\star$ instead of $k$ to indicate that all functions are evaluated at $w^\star$, similarly for $\mathcal{M}_\star$. We call solving (12) forward sensitivity computation. Note that this linear system has the same structure as (8) and can be solved using a factorization of the reduced matrix $\widetilde{\mathcal{M}}_\star$ in (9). It is possible to obtain the sensitivities only for a subvector of $\theta$ by performing the backsolve (12) for the corresponding columns of $J_\star$.

For a given adjoint seed $\nu \in \mathbb{R}^{n_w}$, the adjoint sensitivity of the solution map is

$$s_{\text{adj}}{}^\top \coloneqq \nu^\top \frac{\partial w^{\text{sol}}_{\text{IPM}}}{\partial\theta}(w^\star;\tau,\theta) = -\nu^\top \mathcal{M}_\star(w^\star;\tau,\theta)^{-1} J_\star(w^\star;\tau,\theta). \tag{13}$$

Transposing both sides yields

$$s_{\text{adj}} = -J_\star(w^\star;\tau,\theta)^\top (\mathcal{M}_\star(w^\star;\tau,\theta)^{-\top}\nu). \tag{14}$$

Notice that a system of linear equations similar to (8) but with the coefficient matrix $\mathcal{M}_\star^\top$ has to be solved. Although $\mathcal{M}_\star$ is not symmetric, once $\mu$ and $s$ are eliminated, a system with the same lower dimensional symmetric coefficient matrix $\widetilde{\mathcal{M}}_\star$ is obtained. This reduced system can be tackled with the same efficient algorithm used for the QP solution and the forward sensitivities, while the block elimination of $\mathcal{M}_\star^\top$ requires a slightly different algorithm to process the right-hand side.

In the context of backpropagation through a neural network in which one layer is the solution of an optimization problem, the seed vector $\nu$ corresponds to the activations of the next layer.

**Remark 3 (Approximate Hessian pitfall).** At first glance the factorization of the matrix $\mathcal{M}_\star$, or to be more precise the factorization of $\widetilde{\mathcal{M}}_\star$, is naturally available at the solution when employing an interior-point NLP solver or an SQP solver that tackles the QP subproblems with an interior-point QP solver as outlined in Section 2.1. An efficient implementation thus only requires the evaluation of $J_\star$ and a backsolve routine to complete the solution sensitivity computation, if these conditions hold. However, the exact Hessian factorization is rarely available in practical NLP solvers. Firstly, it is common to use a Hessian approximation, see Remark 1. Secondly, the exact Hessian or its approximation are often regularized, e.g. by adding a positive constant onto the diagonal, commonly known as adding a Levenberg-Marquardt term (Nocedal & Wright, 2006), or by applying more advanced regularization techniques (Verschueren et al., 2017).

We emphasize the requirement of using an exact Hessian in the QP subproblem in the sensitivity computation referring to Thm. 5. If this assumption is not satisfied and a Hessian approximation is

used instead, as suggested in previous works (Amos et al., 2018; Adabag et al., 2026), the solution sensitivities might be highly degraded, as shown in Example 2.

**Computational complexity.** Both the forward and adjoint sensitivity computations require computing $J_\star$ and $\mathcal{M}_\star$, as well as factorizing $\widetilde{\mathcal{M}}_\star$. It is possible to recover the full solution sensitivity by solving (14) using all $n_w$ canonical unit vectors for $\nu$. Compared to evaluating (12) directly, this comes at the additional expense of a multiplication with $J_\star^\top$.

For optimization problems with many parameters, i.e. $n_\theta \geq n_w$ and if only one or a few adjoint sensitivities are needed, it is computationally beneficial to compute these adjoints via (14) and to avoid computing the full forward sensitivity matrix in (12). In particular, this is most efficient when only one adjoint sensitivity is needed as is the case when performing backpropagation through a neural network in which one layer is the solution of an optimization problem.

## 3 Differentiable MPC

In this section, we discuss the efficient solution of NLPs with an optimal control problem (OCP) structure using the tools introduced in the previous section. In nonlinear MPC, one solves at each time instant an OCP of the form

$$\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad \sum_{n=0}^{N-1} L_n(x_n, u_n; \theta) + M(x_N; \theta) \tag{15a}$$

$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{15b}$$

$$x_{n+1} = \phi_n(x_n, u_n; \theta), n = 0, \dots, N-1, \tag{15c}$$

$$0 \geq h_n(x_n, u_n; \theta), n = 0, \dots, N-1, \tag{15d}$$

$$0 \geq h_N(x_N; \theta), \tag{15e}$$

where $\bar{x}_0 \in \mathbb{R}^{n_x}$ is the initial state, $x_n \in \mathbb{R}^{n_x}$, $n = 0, \dots, N$, represent a discrete state trajectory, and $u_n \in \mathbb{R}^{n_u}$, $n = 0, \dots, N-1$, are control inputs. The dynamics $\phi_n$ describe evolution of the state from $x_n$ to $x_{n+1}$. The functions $h_n$ gather the constraints on $x_n$ and $u_n$, $n = 0, \dots, N-1$, and $h_N$ represents the constraints on the terminal state. The parameter vector $\theta \in \mathbb{R}^{n_\theta}$ represents all parameters that can change and with respect to which we want to compute the solution sensitivities. Note that the OCP-structure exploiting algorithms and their implementations described in the sequel are applicable to problems with stage-wise varying dimensions, as described in Frey et al. (2025a).

**Structure exploiting solvers.** While OCP (15) can be phrased as a general dense NLP as in (2), exploiting the specific OCP structure in a tailored algorithm typically is most efficient for their numerical solution (Kouzoupis et al., 2018). When applying an interior-point QP solver to the OCP-structured problem (15), the Riccati-factorization can be applied to efficiently solve the linear system with coefficient matrix $\widetilde{\mathcal{M}}$ (Steinbach, 1994; Frison, 2015). In addition to the problem structure in (15), some solvers, such as `acados` and `HPIPM`, offer special handling of slack variables. These variables are structurally similar to control inputs, but do not enter the dynamics and can only enter the cost linearly or quadratically with diagonal Hessian. This structure often occurs in practice, especially when formulating soft constraints.

**Efficient solution and sensitivity computation.** The computations required to compute the solution sensitivities of an OCP-structured NLP (2) can be summarized as follows:

- (S1) Solve (2) or a smoothed variant of its KKT conditions with $\tau_{\min}$ to obtain $w^{\text{sol}}_{\text{IPM}}(\bar{\theta})$.
- (S2) Evaluate the coefficient matrix $\mathcal{M}_\star$ with exact Hessian $Q_\star = \nabla^2_{zz}\mathcal{L}$.
- (S3) Factorize the corresponding matrix $\widetilde{\mathcal{M}}_\star$.

The above steps need to be carried out independent of whether forward or adjoint sensitivities are computed. For forward sensitivities, the remaining steps are:

- (S4f) Evaluate $J_\star$, the derivatives of the residual map with respect to the parameters.
- (S5f) Solve the linear system (12) to obtain the forward sensitivity result.

For adjoint sensitivities, the remaining steps are:

- (S4a) Solve the linear system in brackets of (14).
- (S5a) Compute the adjoint sensitivities with the seed computed in (S4a), without computing $J_\star$.

## 4 Efficient implementation in acados

In this section, we present the efficient open-source implementation of the algorithms conceptually described above within the acados software package.

**Two-solver approach.** In order to alleviate the potential pitfalls pointed out in Sec. 2.2, we suggest a two-solver approach to carry out the steps outlined in Sec. 3. This approach consists of

1. A *nominal solver*, which uses the most suitable Hessian approximation and regularization technique to obtain $w^\star(\bar{\theta})$ by performing step (S1).
2. A *sensitivity solver*, which carries out steps (S2) to (S5a)/(S5f) of the algorithm above.

The nominal solver could use any QP solver, cf. Remark 4, or even an optimization solver outside acados, as only the primal-dual solution $w^\star(\bar{\theta})$ needs to be transferred to the sensitivity solver. If choosing $\tau_{\min}$ larger than the desired tolerance, the nominal solver needs to compute the primal-dual solution of the smoothed KKT conditions which can be obtained by an SQP variant with a suitable IPM QP solver, like HPIPM, or an NLP IPM solver like IPOPT (Wächter & Biegler, 2006).

Even if all steps are carried out by acados, the flexibility of this approach allows for an overall more efficient algorithm, as structurally different techniques can be used for the two solvers. Firstly, regularization techniques and Hessian approximations can be beneficial for the solution, as motivated in Sec. 2.2, which can require explicit definitions of convex-over-nonlinear structures. Secondly, it might be attractive to choose different QP solvers for the subproblems, including the (partial) condensing algorithm (Axehill, 2015; Frison et al., 2016). Condensing is especially favorable if multiple QP solver iterations (each requiring a KKT matrix factorization within an IPM) are performed for a single condensed QP, as typically in (S1), but not in (S5a)/(S5f). In Appendix A.10, we provide a code snippet showing how these steps can be performed in the proposed implementation.

**Efficient linear system solves.** The linear systems associated with the computation of the forward and adjoint sensitivities, as well as the computation of the Newton step inside the interior-point QP solver, can be reduced to the Riccati factorization of $\widetilde{\mathcal{M}}$, which is efficiently implemented in HPIPM (Frison et al., 2020) utilizing the high-performance linear algebra library BLASFEO (Frison et al., 2018). For the computation of adjoint sensitivities, a routine was implemented in HPIPM to reduce the right-hand side of a linear system with coefficient matrix $\mathcal{M}^\top$ to one with $\widetilde{\mathcal{M}}$. The computational complexities of the Riccati factorization and the corresponding solution routine are $\mathcal{O}(N(n_x+n_u)^3)$ and $\mathcal{O}(N(n_x+n_u)^2)$, respectively.

**Remark 4 (Riccati variants & regularity).** Different variants of the Riccati factorization exist. The classic Riccati recursion allows indefinite full-space Hessians and only requires positive definiteness of the reduced Hessian with respect to the dynamics (15c). In addition, HPIPM provides a square-root implementation based on a Cholesky factorization, which has a lower associated computational cost, but requires the full-space Hessian to be positive definite (Frison & Diehl, 2020). Thus, the classic Riccati algorithm should be used for (S5a)/(S5f), while the square-root algorithm can be leveraged in an efficient implementation of (S1).

**Parallelization.** In order to utilize many CPU cores when computing the solution and sensitivities for a series of values $\bar{\theta}$, we implemented a batch solver class in Python based on a C implementation with OpenMP parallelization. This class also implements parallelized solver interactions, such as updating parameters, and storing and loading initializations.

**Limitations.** The proposed implementation is currently limited to CPU-based computation, reflecting the design philosophy of acados, which is implemented in C with minimal dependencies to support deployment on embedded platforms. While GPU-based implementations may be attractive for training networks that include optimization layers, efficient CPU execution remains essential for real-time MPC on embedded controllers, where GPUs are uncommon due to cost and energy constraints and are often inefficient for small-scale linear algebra. Moreover, nonlinear and constrained formulations hinder batch parallelization due to high variance in solution times, which partly explains the lack of GPU support (cf. Table 2). Another limitation is that the system dynamics $\phi_n$ must be provided in discrete time using CasADi expressions (Andersson et al., 2019). Support for general parametric solution sensitivities within the acados integrators, which are used to efficiently solve initial value problems and can be embedded within OCP solvers (Frey et al., 2019, 2023), is left for future work.

**Integration with ML frameworks.** Our implementation is wrapped for use in common ML frameworks, with additional support for handling diverse parameters. These functionalities are available

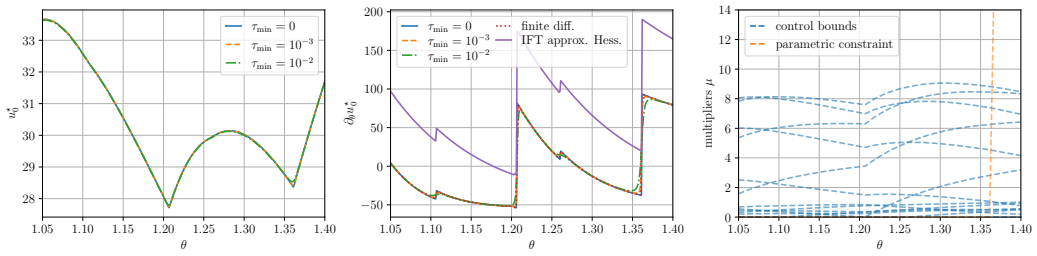


Figure 2: Optimal solution $u_0$ and sensitivities obtained by `acados` compared with finite differences for the OCP described in Example 2. At active set changes, which are indicated by multipliers switching from being zero to being positive (or vice versa), the solution map is nondifferentiable.

Table 1: Timings in [ms] for solving $n_{\text{batch}} = 128$ bounded LQR problems with varying parameters. In the config column, the difference to the default configuration is shown. In parentheses are multiples of the `acados` runtime.

| | **Nominal solution** | | | **Solution + adjoint sens.** | | |
|---|---|---|---|---|---|---|
| config | `acados` | `mpc.pytorch` | `cvxpygen` | `acados` | `mpc.pytorch` | `cvxpygen` |
| default | 8.6 | 59 (×6.9) | 235 (×27) | 21.0 | 102 (×4.9) | 329 (×16) |
| $u_{\max} = 1$ | 15.7 | 1697 (×110) | 1140 (×73) | 23.4 | 1793 (×77) | 1239 (×53) |
| $N = 100$ | 29.3 | 671 (×23) | 925 (×32) | 68.2 | 828 (×12) | 2268 (×33) |
| $n_x = 16, n_u = 8$ | 19.7 | 89 (×4.5) | 585 (×30) | 40.7 | 153 (×3.8) | 856 (×21) |

in the `leap-c` project (Fichtner et al., 2025); see A.9 for details. An example of learning-enhanced MPC using this pipeline is presented in A.14. End-to-end learning-enhanced MPC experiments that leverage the proposed differentiable nonlinear MPC implementation are described in Llanes et al. (2026); Jahncke et al. (2026).

## 5 Numerical examples

This section presents three numerical examples. The first example showcases the proposed implementation's ability to differentiate the solution of an OCP with parametric cost, dynamics and constraints. In a second example, we compare the efficiency of the proposed SBQP solution sensitivity computation with state-of-the-art methods on a linear-quadratic OCP with simple input bounds. Thirdly, a nonlinear OCP with high dimensional parameter vector allows us to evaluate the computation times of computing forward and adjoint solution sensitivities in `acados`. All code required to reproduce the results in this paper is available, see Appendix A.11.

**Example 2 (OCP with highly nonlinear parameter dependency).** We consider a nonlinear OCP with $n_x = 4, n_u = 1, n_\theta = 1$ and $N = 50$. The parameter $\theta$ enters the cost, the inequalities linearly and the dynamics nonlinearly. A detailed description of the example is provided in Appendix A.12. Figure 2 visualizes the optimal solution $u_0^\star(\theta)$, as well as the solution sensitivities obtained via `acados` with different values of $\tau_{\min}$. The solver tolerance is set to $10^{-8}$. Thus, for $\tau_{\min} = 0$, the interior-point KKT system is solved with this accuracy, while the barrier parameter values $\tau_k$ never reach zero. Additionally, sensitivities obtained by applying finite differences and sensitivities computed with the IFT approach and a Gauss-Newton Hessian approximation are plotted. The solution sensitivities obtained with $\tau_{\min} = 0$ are consistent with the ones obtained by finite differences, while the results obtained by the IFT approach with a Gauss-Newton Hessian approximation are completely off. Additionally, the smoothing effect of larger values for $\tau_{\min}$ is visible for both the solution and the corresponding sensitivities. The kinks in the solution map are very pronounced and align with the active set changes indicated by multipliers changing from zero to a positive value.

**Example 3 (Benchmark example).** Next, we compare the proposed differentiable OCP solver with `mpc.pytorch` (Amos et al., 2018) and `cvxpygen` paired with `OSQP` (Schaller & Boyd, 2026; Stellato et al., 2020). To enable comparison with these methods, we consider an OCP with quadratic cost, linear discrete-time dynamics, and bounds on the control inputs, cf. Table 2. We treat the dynamics matrices and offset, as well as the cost matrices, as parameters $\theta$. The problem dimensions are chosen as $N = 20$, $n_x = 8$, $n_u = 4$ if not written otherwise. We consider both a loose-bound case and a case with active control bounds. A detailed formulation and motivations for design choices of the benchmark problem can be found in Appendix A.13. We solve $n_{\text{batch}}$ problems with different initial states $\bar{x}_0$, whose components are sampled from a standard normal distribution.

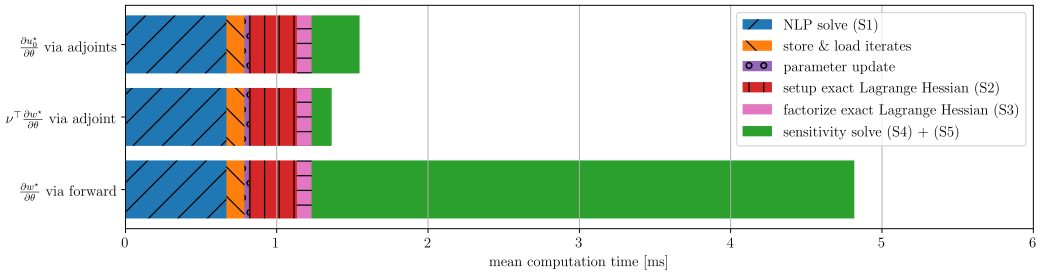


Figure 3: Computation time for evaluating different kinds of sensitivities for Example 4.

Table 1 reports nominal solution times and joint solution-plus-adjoint-sensitivity times. Since the OCPs are strictly convex, it is not necessary to use the two-solver approach in `acados`. The table reports more than three times lower runtimes for `acados` in every configuration compared to the other solvers. We verified that all solvers converge to the same primal solution. With tight control bounds, `mpc.pytorch` shows a large increase in runtime while the `acados` runtime changes much less. Thus `acados` achieves speedup factors greater than 50 in this case. The online computations needed when applying a (learned) MPC scheme correspond to the nominal solution, such that those timings can be seen as the inference time. The case $N = 100$ shows runtime growth for `acados` that is roughly consistent with linear scaling in the horizon. The larger-dimension case shows a qualitatively similar runtime comparison to the default case. Appendix A.13 also reports a GPU-enabled variant of the benchmark. The loose-bound results show a similar qualitative comparison; for tight bounds, however, the GPU results use an earlier dynamics setup for which `mpc.pytorch` fails to converge, so the reported runtime ratios do not represent speedups for successful solves. The main takeaway is that `acados` can handle more general problem formulations than the investigated alternatives while reducing the computation time.

**Example 4 (Chain of masses).** We regard the example of a nonlinear OCP with a chain of masses model and many parameters, as described in Reinhardt et al. (2024) with prediction horizon $N = 40$ and number of masses fixed to $n_{\text{mass}} = 3$ such that $n_x = 9, n_u = 3, n_\theta = 113$. The findings in Reinhardt et al. (2024) show a speedup of over one magnitude in forward sensitivity computation when using `acados` over general purpose libraries. We consider different variants of solution sensitivity computation and analyze their computation times in Figure 3. We observe that the evaluation of one or a few adjoint sensitivities is computationally much cheaper compared to a full forward Jacobian sensitivity. However, if many directions are needed, the computational cost of computing many adjoint direction of $J_\star$, see (14) and step (S5a), makes a computation via adjoints more expensive. All variants only differ in the last step, namely the sensitivity solve with a given factorization. The takeaway is that computing a single adjoint sensitivity is roughly 4 times faster than evaluating the full forward sensitivity and the adjoint sensitivity contains all required information when performing backpropagation to optimize parameters in a learning framework.

## 6 Conclusion and outlook

This paper presents an efficient, modular, open-source implementation for solving general optimal control structured nonlinear programs and computing the corresponding forward or adjoint solution sensitivities within the software framework `acados`. Various examples highlight its usability, flexibility and superior performance with respect to existing implementations. This work builds the basis for integrating efficient and reliable solvers for optimal control problems as differentiable layers in machine learning networks. Future work includes a comparison of the smoothed sensitivities with conservative Jacobians (Bolte et al., 2021) with respect to their performance within a learning pipeline, as well as extending the `acados` implementation to allow using its efficient integrators and implementing additional regularizations to obtain sensitivities of locally non-unique solutions as suggested in Zuliani et al. (2025b).

### Acknowledgements

This research was supported by DFG via projects 504452366 (SPP 2364), 560056112 (robust MPC), 535860958 (ALeSCo) and 525018088 (MAWERO).

## References


Adabag, E., Greiff, M., Subosits, J., and Lew, T. Differentiable model predictive control on the GPU. In *International Conference on Learning Representations*, volume 2026, pp. 132305–132323, 2026.

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

Amos, B. and Kolter, J. Z. Optnet: Differentiable optimization as a layer in neural networks. In *International conference on machine learning*, pp. 136–145. PMLR, 2017.

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

Andersson, J. A. E. and Rawlings, J. B. Sensitivity analysis for nonlinear programming in CasADi. In *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2018.

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

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

Axehill, D. Controlling the level of sparsity in MPC. *Systems & Control Letters*, 76:1–7, 2015.

Barto, A. G., Sutton, R. S., and Anderson, C. W. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5): 834–846, 1983.

Baumgärtner, K. and Diehl, M. 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)*, 2022.

Baumgärtner, K., Messerer, F., and Diehl, M. A unified local convergence analysis of differential dynamic programming, direct single shooting, and direct multiple shooting. In *Proceedings of the European Control Conference (ECC)*, 2023.

Bock, H., Kostina, E., and Schlöder, J. Numerical methods for parameter estimation in nonlinear differential algebraic equations. *GAMM Mitteilungen*, 30/2:376–408, 2007.

Bolte, J., Le, T., Pauwels, E., and Silveti-Falls, T. Nonsmooth implicit differentiation for machine-learning and optimization. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 13537–13549. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/70afbf2259b4449d8ae1429e054df1b1-Paper.pdf.

Bounou, O., Ponce, J., and Carpentier, J. Leveraging proximal optimization for differentiating optimal control solvers. In *2023 62nd IEEE Conference on Decision and Control (CDC)*, pp. 6313–6320, 2023. doi: 10.1109/CDC49753.2023.10384034.

Cai, W., Kordabad, A. B., and Gros, S. Energy management in residential microgrid using model predictive control-based reinforcement learning and Shapley value. *Engineering Applications of Artificial Intelligence*, 119:105793, 2023a. ISSN 0952-1976. doi: 10.1016/j.engappai.2022.105793.

Cai, W., Sawant, S., Reinhardt, D., Rastegarpour, S., and Gros, S. A learning-based model predictive control strategy for home energy management systems. *IEEE Access*, 11:145264–145280, 2023b. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3346324.

Davis, D., Drusvyatskiy, D., Kakade, S., and Lee, J. D. Stochastic subgradient method converges on tame functions. *Foundations of computational mathematics*, 20(1):119–154, 2020.

Diehl, M., Bock, H. G., and Schlöder, J. P. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005. doi: 10.1137/S0363012902400713. URL http://epubs.siam.org/sicon/resource/1/sjcodc/v43/i5/p1714_s1.

Dinev, T., Mastalli, C., Ivan, V., Tonneau, S., and Vijayakumar, S. Differentiable optimal control via differential dynamic programming, 2022. URL https://arxiv.org/abs/2209.01117.

Dini, U. Lezioni di analisi infinitesimale. *Tipogradico Succ. Fratelli Nistri, Pisa*, 1, 1907.

Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4): 327–363, 2014. doi: 10.1007/s12532-014-0071-1.

Ferreau, H. J., Almer, S., Verschueren, R., Diehl, M., Frick, D., Domahidi, A., Jerez, J. L., Stathopoulos, G., and Jones, C. Embedded optimization methods for industrial automatic control. In *Proceedings of the IFAC World Congress*, 2017.

Fichtner, L., Reinhardt, D., Hoffmann, J., Airaldi, F., Frey, J., Hromatko, J. K., Baumgaertner, K., Amria, M., Reiter, R., and Sawant, S. leap-c releases. https://doi.org/10.5281/zenodo.17244100, 2025.

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

Frey, J., Cairano, S. D., and Quirynen, R. Active-Set based Inexact Interior Point QP Solver for Model Predictive Control. In *Proceedings of the IFAC World Congress*, 2020.

Frey, J., De Schutter, J., and Diehl, M. Fast integrators with sensitivity propagation for use in CasADi. In *Proceedings of the European Control Conference (ECC)*, 2023.

Frey, J., Baumgärtner, K., and Diehl, M. Gauss-Newton Runge-Kutta integration for efficient discretization of optimal control problems with long horizons and least-squares costs. *European Journal of Control*, pp. 101038, 2024a.

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

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

Frey, J., Verschueren, R., Frison, G., Zanelli, A., Kouzoupis, D., Novoselnik, B., Sartor, T., van Duijkeren, N., Quirynen, R., Koenemann, J., Kirchengast, M., Baumgärtner, K., Kiessling, D., Schwienbacher, M., Qelibari, Y. C. R., Arnström, D., Gruijthuijsen, C., Geelen, T., Doan, D., Schöls, T., Çar, H. B., Schlagenhauf, J., Gillis, J., Jespersen, T., Michaud, F. B. B., Geweth, D., Ghezzi, A., Horn, G., Grunnet, J., Cecchin, L., Gao, Y., Wang, Y., Hänggi, S., Schäfer, H., Pramanick, S., Shihadeh, A., Koch, M., Saljanin, M., and Diehl, M. acados releases. https://www.doi.org/10.5281/zenodo.7371687, 2025b.

Frison, G. *Algorithms and Methods for High-Performance Model Predictive Control*. PhD thesis, Technical University of Denmark (DTU), 2015.

Frison, G. and Diehl, M. HPIPM: a high-performance quadratic programming framework for model predictive control. In *Proceedings of the IFAC World Congress*, Berlin, Germany, 2020.

Frison, G. and Jørgensen, J. B. A fast condensing method for solution of linear-quadratic control problems. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pp. 7715–7720, 2013.

Frison, G., Kouzoupis, D., Jørgensen, J. B., and Diehl, M. 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, 2016.

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

Frison, G., Sartor, T., Zanelli, A., and Diehl, M. The BLAS API of BLASFEO: Optimizing performance for small matrices. *ACM Transactions on Mathematical Software (TOMS)*, 46(2):15:1–15:36, 2020.

Ghezzi, A., Hoffman, J., Frey, J., Boedecker, J., and Diehl, M. Imitation learning from nonlinear MPC via the exact Q-loss and its Gauss-Newton approximation. *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2023.

Görges, D. Relations between model predictive control and reinforcement learning. *IFAC-PapersOnLine*, 50(1):4920–4928, 2017.

Goulart, P. J. and Chen, Y. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv preprint arXiv:2405.12762*, 2024.

Grant, M., Boyd, S., and Ye, Y. *Disciplined convex programming*. Springer, 2006.

Gros, S. and Zanon, M. Data-driven economic NMPC using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636648, 2020. ISSN 2334-3303. doi: 10.1109/tac.2019.2913768. URL http://dx.doi.org/10.1109/TAC.2019.2913768.

Gros, S. and Zanon, M. Learning for MPC with stability & safety guarantees. *Automatica*, 146: 110598, 2022.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th international conference on machine learning, ICML*, volume 80 of *Proceedings of machine learning research*, pp. 1856–1865. PMLR, 2018.

Hewing, L., Wabersich, K. P., Menner, M., and Zeilinger, M. N. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3:269–296, 2020.

Houska, B., Ferreau, H. J., and Diehl, M. ACADO toolkit – an open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011. doi: 10.1002/oca.939. URL http://onlinelibrary.wiley.com/doi/10.1002/oca.939/abstract.

Jahncke, F., Zarrouki, B., Piccinini, M., D'sa, J., Isele, D., Bae, S., and Betz, J. Differentiable weights-varying nonlinear mpc via gradient-based policy learning: An autonomous vehicle guidance example. *IEEE Robotics and Automation Letters*, pp. 1–8, 2026. doi: 10.1109/LRA.2026.3662644.

Jin, W., Wang, Z., Yang, Z., and Mou, S. Pontryagin differentiable programming: An end-to-end learning and control framework. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 7979–7992. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/5a7b238ba0f6502e5d6be14424b20ded-Paper.pdf.

Karush, W. Minima of Functions of Several Variables with Inequalities as Side Conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

Kordabad, A. B. and Gros, S. Q-learning of the storage function in economic nonlinear model predictive control. *Engineering Applications of Artificial Intelligence*, 116:105343, November 2022. ISSN 0952-1976. doi: 10.1016/j.engappai.2022.105343.

Kordabad, A. B., Cai, W., and Gros, S. MPC-based reinforcement learning for economic problems with application to battery storage. In *2021 European Control Conference (ECC)*, pp. 2573–2578, 2021. doi: 10.23919/ECC54610.2021.9654852.

Kordabad, A. B., Wisniewski, R., and Gros, S. Safe reinforcement learning using wasserstein distributionally robust MPC and chance constraint. *IEEE Access*, 10:130058–130067, 2022. ISSN 2169-3536. doi: 10.1109/ACCESS.2022.3228922.

Kordabad, A. B., Zanon, M., and Gros, S. Equivalence of optimality criteria for markov decision process and model predictive control. *IEEE Transactions on Automatic Control*, 69(2):1149–1156, February 2024. ISSN 1558-2523. doi: 10.1109/TAC.2023.3277309.

Kouzoupis, D., Frison, G., Zanelli, A., and Diehl, M. Recent advances in quadratic programming algorithms for nonlinear model predictive control. *Vietnam Journal of Mathematics*, 46(4):863–882, 2018.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Li, W. and Todorov, E. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics*, 2004.

Llanes, C., Jensen, S. W., and Coogan, S. Merging model-based control with multi-agent reinforcement learning for multi-agent cooperative teaming strategies. *arXiv preprint arXiv:2606.06011*, 2026.

Lüken, L., Brandner, D., and Lucia, S. Sobolev training for data-efficient approximate nonlinear mpc. *IFAC-PapersOnLine*, 56(2):5765–5772, 2023.

Mallick, S., Airaldi, F., Dabiri, A., Sun, C., and De Schutter, B. Reinforcement learning-based model predictive control for greenhouse climate control. *Smart Agricultural Technology*, 10: 100751, March 2025. ISSN 2772-3755. doi: 10.1016/j.atech.2024.100751.

Mandi, J. and Guns, T. Interior point solving for lp-based prediction+ optimisation. *Advances in Neural Information Processing Systems*, 33:7272–7282, 2020.

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

Nocedal, J. and Wright, S. J. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

Nurkanović, A., Zanelli, A., Albrecht, S., and Diehl, M. The Advanced Step Real Time Iteration for NMPC. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2019.

Oshin, A., Almubarak, H., and Theodorou, E. A. Differentiable robust model predictive control, 2024. URL `https://arxiv.org/abs/2308.08426`.

Pineda, L., Fan, T., Monge, M., Venkataraman, S., Sodhi, P., Chen, R. T., Ortiz, J., DeTone, D., Wang, A., Anderson, S., et al. Theseus: A library for differentiable nonlinear optimization. *Advances in Neural Information Processing Systems*, 35:3801–3818, 2022.

Pirnay, H., López-Negrete, R., and Biegler, L. T. Optimal sensitivity based on IPOPT. *Mathematical Programming Computation*, 4(4):307–331, 2012. ISSN 1867-2957. doi: 10.1007/s12532-012-0043-2. URL `https://doi.org/10.1007/s12532-012-0043-2`.

Reinhardt, D., Baumgärtner, K., Frey, J., Diehl, M., and Gros, S. Mpc4rl-a software package for reinforcement learning based on model predictive control. In *2024 IEEE 63rd Conference on Decision and Control (CDC)*, pp. 1787–1794. IEEE, 2024.

Reiter, R., Ghezzi, A., Baumgärtner, K., Hoffmann, J., McAllister, R. D., and Diehl, M. Ac4mpc: Actor-critic reinforcement learning for guiding model predictive control. *IEEE Transactions on Control Systems Technology*, 34(1):395–410, 2026a.

Reiter, R., Hoffmann, J., Reinhardt, D., Messerer, F., Baumgärtner, K., Sawant, S., Bödecker, J., Diehl, M., and Gros, S. Synthesis of model predictive control and reinforcement learning: Survey and classification. *Annual Reviews in Control*, 61:101045, 2026b. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2026.101045. URL https://www.sciencedirect.com/science/article/pii/S1367578826000015.

Rickenbach, R., Lahoud, A. A., Schaffernicht, E., Zeilinger, M. N., and Stork, J. A. Zipmpc: Compressed context-dependent mpc cost via imitation learning. *arXiv preprint arXiv:2507.13088*, 2025.

Romero, A., Song, Y., and Scaramuzza, D. Actor-critic model predictive control. In *IEEE International Conference on Robotics and Automation*, 2024. URL https://arxiv.org/pdf/2306.09852.pdf.

Schaller, M. and Boyd, S. Code generation for solving and differentiating through convex optimization problems. *Optimization and Engineering*, 27(2):1425–1449, 2026.

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

Steinbach, M. C. A structured interior point SQP method for nonlinear optimal control problems. In Bulirsch, R. and Kraft, D. (eds.), *Computation Optimal Control*, pp. 213–222, Basel Boston Berlin, 1994. Birkhäuser.

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

Tamar, A., Thomas, G., Zhang, T., Levine, S., and Abbeel, P. Learning from the hindsight plan episodic mpc improvement. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 336–343, 2017. doi: 10.1109/ICRA.2017.7989043.

Tassa, Y., Mansard, N., and Todorov, E. Control-limited differential dynamic programming. In *IEEE International Conference on Robotics and Automation*, 2014.

Todorov, E. and Li, W. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the American Control Conference (ACC)*, 2005.

Vanroye, L., Sathya, A., De Schutter, J., and Decré, W. Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10036–10043. IEEE, 2023.

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

Verschueren, R., Frison, G., Kouzoupis, D., Frey, J., van Duijkeren, N., Zanelli, A., Novoselnik, B., Albin, T., Quirynen, R., and Diehl, M. acados – a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, pp. 147–183, 2021. ISSN 1867-2957. doi: 10.1007/s12532-021-00208-8.

Wächter, A. and Biegler, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

Wang, Y. and Boyd, S. Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology*, 18(2):267–278, 2010.

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

Zanelli, A., Domahidi, A., Jerez, J. L., and Morari, M. FORCES NLP: An efficient implementation of interior-point methods for multistage nonlinear nonconvex programs. *International Journal of Control*, pp. 13–29, 2017. doi: 10.1080/00207179.2017.1316017.

Zavala, V. M. and Biegler, L. T. The advanced step NMPC controller: Optimality, stability and robustness. *Automatica*, 45:86–93, 2009.

Zuliani, R., Balta, E. C., and Lygeros, J. BP-MPC: Optimizing the closed-loop performance of MPC using backpropagation. *IEEE Transactions on Automatic Control*, 2025a.

Zuliani, R., Balta, E. C., and Lygeros, J. Differentiable-by-design nonlinear optimization for model predictive control, 2025b. URL `https://arxiv.org/abs/2509.12692`.

Table 2: Overview on differentiable optimization software and suitability for optimal control. *OCP tailored* indicates whether the internal algorithms exploit the OCP sparsity structure. The column *adjoint* indicates if adjoint sensitivities as needed for backpropagation are implemented. The column *ineq. support* indicates whether inequalities are supported. The following columns summarize which parts of the problem can be nonlinear.

| Software / publication | OCP tailored | ad-joint | ineq. support | Nonlinearity in cost | eq. | ineq. | exact results | parallel-ization | permissive license |
|---|---|---|---|---|---|---|---|---|---|
| `acados` this paper | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | `OpenMP` | ✅ BSD-2 |
| `mpc.pytorch` Amos et al. (2018) | ✅ | ✅ | ❌[a] | ✅ | ❌ | ❌ | ❌ | `PyTorch` | ✅ MIT |
| `cvxpygen` + `cvxpylayers` Schaller et al. (2022) Schaller & Boyd (2026) | ❌ | ✅ | ✅ | ⚪[b] | ❌ | ⚪[b] | ✅ | Python multi-threading | ✅ Apache 2.0 |
| `diffmpc` (Toyota) Adabag et al. (2026) | ✅ | ✅ | ❌ | ✅ | ✅ | ❌ | ❌ | `JAX` | ❌ CC BY-NC-ND 4.0 |
| `trajax` (Google) no paper | ✅ | ✅ | ❌ | ✅ | ✅ | ❌ | ⚪[c] | `JAX` | ✅ Apache 2.0 |
| `theseus` (Facebook) Pineda et al. (2022) | ❌ | ✅ | ❌ | ⚪[d] | ❌ | ❌ | ❌ | `PyTorch` | ✅ MIT |
| `sIPOPT` Pirnay et al. (2012)[e] | ❌ | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | – | ✅ EPL |
| `CasADi` native Andersson & Rawlings (2018) | ❌ | ✅ | ✅ | ✅ | ✅ | ✅ | ✅ | – | ✅ LGPL |

[a] while simple control bounds can be added to the problem formulation, the experiments in Example 3 and Appendix A.13 revealed that the solver fails to converge if active set is nontrivial.
[b] only convex cost and constraints.
[c] A set of options appears to allow the use of the exact Hessian for solution sensitivity computation; however, the software and documentation do not specify the conditions under which the resulting sensitivities are accurate.
[d] supports only nonlinear least-squares cost and no equality constraints, as needed for dynamics. OCPs with quadratic costs can be implemented using the single-shooting formulation.
[e] functionality extremely limited in practice. Not usable in `CasADi` and abandoned rewrite effort in `https://github.com/athrpf/sipopt2`.

# A Appendix

## A.1 Differentiable MPC software

Table 2 provides a qualitative comparison of differentiable optimization solvers, with a focus on optimal control. It can be seen that inequality treatment, a feature that is essential to leverage safety benefits of learning-enhanced MPC, is scarce. In particular, no OCP-tailored software other than the proposed implementation supports inequality constraints. Some existing solutions rely on identifying the active set at the solution and applying the IFT, e.g. the "native" `CasADi` solution sensitivity implementation (Andersson & Rawlings, 2018)[1]. This clear separation does not allow one to differentiate the solution across active set changes in contrast to the proposed approach.

## A.2 Implicit function theorem

As it is essential for this work, we state the implicit function theorem (IFT) in its classical form Dini (1907).

**Theorem 4 (Implicit function theorem (IFT)).** Let $r : \mathbb{R}^{n_\theta} \times \mathbb{R}^{n_w} \to \mathbb{R}^{n_w}$ be continuously differentiable in a neighborhood of $(\bar{\theta}, \bar{w})$ with $r(\bar{\theta}, \bar{w}) = 0$, and let the Jacobian $\nabla_w r(\bar{\theta}, \bar{w})$ be

[1] `https://github.com/casadi/casadi/discussions/3156`

nonsingular. Then the solution mapping

$$S : \theta \mapsto \{w \in \mathbb{R}^{n_w} \mid r(\theta, w) = 0\} \tag{16}$$

has a single-valued localization $s$ around $\bar{\theta}$ for $\bar{w}$ which is continuously differentiable in a neighborhood $Q$ of $\bar{\theta}$ with Jacobian satisfying

$$\nabla s(\theta) = -\nabla_w r(\theta, s(\theta))^{-1} \nabla_\theta r(\theta, s(\theta)) \text{ for all } \theta \in Q. \tag{17}$$

### A.3 Interpreting SBQP as Sequential Convex Programming (SCP) applied to the barrier problem

This section presents additional details on SBQP introduced in Section 2.1, in particular if a value $\tau_{\min} > 0$ is used. For $\tau_{\min} = 0$, the algorithm can be interpreted as a standard SQP method.

In the following, we assume $\tau \coloneqq \tau_{\min} > 0$ is used and show that SBQP solves the nonlinear barrier problem (21). Moreover, we show that, in some special cases, it can be interpreted in the framework of sequential convex programming (SCP).

Let us repeat the original NLP

$$\begin{aligned} \min_{z \in \mathbb{R}^{n_z}} \quad & f(z; \theta) \\ \text{s.t.} \quad & g(z; \theta) = 0, \\ & h(z; \theta) \leq 0. \end{aligned} \tag{18}$$

Replacing the inequality constraint by a barrier term, introducing the barrier parameter $\tau > 0$, yields the following log-barrier problem:

$$\begin{aligned} \min_{z \in \mathbb{R}^{n_z}} \quad & f(z; \theta) - \tau \sum_{i=1}^{n_h} \log(-h_i(z; \theta)) \\ \text{s.t.} \quad & g(z; \theta) = 0, \end{aligned} \tag{19}$$

with associated KKT conditions

$$\nabla_z f(z; \theta) + \nabla_z g(z; \theta) \lambda + \nabla_z h(z; \theta) \mu = 0, \tag{20a}$$
$$g(z; \theta) = 0, \tag{20b}$$
$$h(z; \theta) \leq 0, \tag{20c}$$
$$\mu \geq 0, \tag{20d}$$
$$\mu_i h_i(z; \theta) = -\tau_{\min}, \; i = 1, \ldots, n_h. \tag{20e}$$

Their derivation is similar to the ones derived next with linearized constraints and a quadratic cost approximation. We also call (20) the smoothed interior-point KKT system as introduced next to (11).

By introducing slack variables $s \in \mathbb{R}^{n_h}$, we can reformulate (19) as

$$\begin{aligned} \min_{z \in \mathbb{R}^{n_z}, s \in \mathbb{R}^{n_h}} \quad & f(z; \theta) - \tau \sum_{i=1}^{n_h} \log(s_i) \\ \text{s.t.} \quad & g(z; \theta) = 0, \\ & h(z; \theta) + s = 0. \end{aligned} \tag{21}$$

Let us only keep the convexity of the logarithmic barrier term, quadratically approximate the cost function and linearize the remaining parts of the problem (21) to arrive at

$$\begin{aligned} \min_{\substack{\Delta z \in \mathbb{R}^{n_z}, \\ s \in \mathbb{R}^{n_h}}} \quad & f_k + q_k^\top \Delta z + \tfrac{1}{2} \Delta z^\top \nabla_{zz}^2 f(z_k; \theta) \Delta z - \tau \sum_{i=1}^{n_h} \log(s_i) \\ \text{s.t.} \quad & g_k + G_k \Delta z = 0, \\ & h_k + H_k \Delta z + s = 0, \end{aligned} \tag{22}$$

where the same shorthands as in Section 2.1 are used, namely $q_k = \nabla_z f(z_k; \theta)$, $g_k = g(z_k; \theta)$, $h_k = h(z_k; \theta)$ and $G_k = \nabla g(z_k; \theta)^\top$, $H_k = \nabla h(z_k; \theta)^\top$, as well as $f_k = f(z_k; \theta)$.

The KKT conditions of this problem read:

$$q_k + \nabla^2_{zz} f(z_k;\theta)\Delta z + G_k^\top \lambda + H_k^\top \mu = 0, \tag{23a}$$
$$-\frac{\tau}{s_i} + \mu_i = 0,\ i = 1,\dots,n_h, \tag{23b}$$
$$g_k + G_k \Delta z = 0, \tag{23c}$$
$$h_k + H_k \Delta z + s = 0, \tag{23d}$$
$$s > 0. \tag{23e}$$

Equation (23b) corresponds to $\nabla_s \mathcal{L} = 0$ and (23e) is needed to ensure a well-defined barrier term. We can rewrite (23b) as $\mu_i s_i = \tau$ and deduce $\mu > 0$. This allows us to rewrite (23) as

$$\nabla^2_{zz} f(z_k;\theta)\Delta z + q_k + G_k^\top \lambda + H_k^\top \mu = 0, \tag{24a}$$
$$g_k + G_k \Delta z = 0, \tag{24b}$$
$$h_k + H_k \Delta z + s = 0, \tag{24c}$$
$$\mu_i s_i = \tau,\ i = 1,\dots,n_h, \tag{24d}$$
$$s, \mu > 0. \tag{24e}$$

These conditions correspond to the ones solved in an IPM QP solver, namely (7a)-(7d), with the only difference being that (7a) uses a Hessian approximation of the Lagrangian $Q_k$, while (24a) uses the exact Hessian of the objective.

These matrices coincide if the objective is of linear least-squares type and a Gauss-Newton Hessian approximation is used for $Q_k$. For an NLP with least-squares objective, i.e. $f(z;\theta) = \frac{1}{2}\|F(z;\theta) - y_{\text{ref}}\|^2_W$, the Gauss-Newton Hessian approximation is $Q_k = J_k^\top W J_k$ with $J = \frac{\partial F}{\partial z}(z_k,\theta)$. If the function $F$ is linear in $z$, the Gauss-Newton Hessian approximation corresponds to the exact Hessian of the objective $\nabla^2_{zz} f(z_k;\theta)$.

In case of a linear least-squares objective and a Gauss-Newton Hessian approximation, SBQP can be interpreted in the framework of Sequential Convex Programming (SCP) (Messerer et al., 2021) on the nonlinear barrier problem (21) and the linear convergence rate result, Thm 4.5 in Messerer et al. (2021), holds.

In the general case, where $f$ is not of linear least-squares type, and any Hessian approximation $Q_k$ is used in the QP solver, the algorithm still provides a solution of (21), assuming it converges.

### A.4 Solution sensitivities for NLPs

This section provides a proof of Theorem 1, which follows from Assumption 1 and Theorem 4.

Strict complementarity allows us to isolate the strictly active inequality constraints at the solution $z^\star(\bar\theta)$ and denote them by $h_\mathcal{A}$, with associated multipliers $\mu_\mathcal{A}$. All other inequalities are inactive with zero multiplier values. We want to apply the IFT in Theorem 4 at the solution of the KKT conditions $(z^\star(\bar\theta), \lambda^\star(\bar\theta), \mu^\star_\mathcal{A}(\bar\theta))$ to the residual function

$$r(z,\lambda,\mu_\mathcal{A};\theta) = \begin{bmatrix} \nabla_z f(z;\theta) + \nabla_z g(z;\theta)\lambda + \nabla_z h_\mathcal{A}(z;\theta)\mu_\mathcal{A} \\ g(z;\theta) \\ h_\mathcal{A}(z;\theta) \end{bmatrix}. \tag{25}$$

The matrix to be inverted can be written as

$$\frac{\partial r}{\partial(z,\lambda,\mu_\mathcal{A})} = \begin{bmatrix} \nabla^2_{zz}\mathcal{L}(z,\lambda,\mu;\theta) & \nabla_z g(z;\theta) & \nabla_z h_\mathcal{A}(z;\theta) \\ \nabla_z g(z;\theta)^\top & 0 & 0 \\ \nabla_z h_\mathcal{A}(z;\theta)^\top & 0 & 0 \end{bmatrix}. \tag{26}$$

LICQ and SOSC imply that this matrix is invertible at the solution, cf. Lemma 16.1 in Nocedal & Wright (2006), which allows us to apply the IFT and concludes the proof of Theorem 1.

In particular, the IFT implies that the solution sensitivity can be computed as

$$\begin{bmatrix} \frac{\partial z^\star}{\partial\theta}(\bar\theta) \\ \frac{\partial \lambda^\star}{\partial\theta}(\bar\theta) \\ \frac{\partial \mu^\star_\mathcal{A}}{\partial\theta}(\bar\theta) \end{bmatrix} = -\left(\frac{\partial r}{\partial(z,\lambda,\mu_\mathcal{A})}\right)^{-1} \frac{\partial r}{\partial\theta}, \tag{27}$$

where the derivatives of $r$ are evaluated at the solution $(z^\star(\bar\theta), \lambda^\star(\bar\theta), \mu^\star_\mathcal{A}(\bar\theta))$.

### A.5 Proof: Barrier NLP and barrier QP sensitivities coincide

This section provides a proof of Theorem 3, showing that the optimality conditions of the barrier QP match the nonlinear smoothed KKT conditions at convergence of SBQP.

The linearized problem quantities of the BQP at the KKT point are defined as $q_\star = \nabla_z f(z^\star;\theta)$, $Q_\star = \nabla^2_{zz}\mathcal{L}(z^\star,\lambda^\star,\mu^\star;\bar\theta)$, $g_\star = g(z^\star;\theta)$, $h_\star = h(z^\star;\theta)$, $G_\star = \nabla g(z^\star;\theta)^\top$, $H_\star = \nabla h(z^\star;\theta)^\top$. The KKT conditions of (10) at the KKT point are

$$Q_\star \Delta z + q_\star + G_\star^\top \lambda + H_\star^\top \mu = 0, \tag{28a}$$
$$g_\star + G_\star \Delta z = 0, \tag{28b}$$
$$h_\star + H_\star \Delta z + s = 0, \tag{28c}$$
$$\mu_i s_i = \tau,\ i = 1,\dots,n_h, \tag{28d}$$
$$s, \mu > 0. \tag{28e}$$

Since $(z^\star, s^\star, \lambda^\star, \mu^\star)$ is a KKT point of (21) at $\bar\theta$, the relations $g(z^\star;\bar\theta) = 0$, $h(z^\star;\bar\theta) + s^\star = 0$, $\mu_i^\star s_i^\star = \tau$, and

$$\nabla^2_{zz}\mathcal{L}(z^\star,\lambda^\star,\mu^\star;\bar\theta)\cdot 0 + q_\star + G_\star^\top\lambda^\star + H_\star^\top\mu^\star = 0$$

show that $(\Delta z, s, \lambda, \mu) = (0, s^\star, \lambda^\star, \mu^\star)$ satisfies (28), which completes the first part.

The sensitivities of the BQP solution can be computed via the IFT in Theorem 4, given LICQ and SOSC. This yields

$$\frac{\partial}{\partial\theta}\begin{bmatrix}\Delta z^{\mathrm{sol}}_{\mathrm{BQP}}(\tau;\bar\theta,v^\star)\\ \lambda^{\mathrm{sol}}_{\mathrm{BQP}}(\tau;\bar\theta,v^\star)\\ \mu^{\mathrm{sol}}_{\mathrm{BQP}}(\tau;\bar\theta,v^\star)\\ s^{\mathrm{sol}}_{\mathrm{BQP}}(\tau;\bar\theta,v^\star)\end{bmatrix} = -\begin{bmatrix}Q_\star & G_\star^\top & H_\star^\top & 0\\ G_\star & 0 & 0 & 0\\ H_\star & 0 & 0 & \mathbb{1}\\ 0 & 0 & S^\star & M^\star\end{bmatrix}^{-1} r^{\mathrm{BQP}}_{\mathrm{sens},\star}, \tag{29}$$

where $S^\star = \mathrm{diag}(s^\star)$ and $M^\star = \mathrm{diag}(\mu^\star)$ and

$$r^{\mathrm{BQP}}_{\mathrm{sens},\star} = \frac{\partial}{\partial\theta}\begin{bmatrix}Q_\star\Delta z + q_\star + G_\star^\top\lambda + H_\star^\top\mu\\ g_\star + G_\star\Delta z\\ h_\star + H_\star\Delta z + s\\ (\mu_i s_i - \tau)_{i=1,\dots,n_h}\end{bmatrix}. \tag{30}$$

This linear system coincides with (12) if the exact Hessian $Q_\star = \nabla^2_{zz}\mathcal{L}(z^\star,\lambda^\star,\mu^\star;\bar\theta)$ is used, which concludes the proof.

### A.6 Optimality conditions of the quadratic subproblem at SQP convergence

Since the proposed SBQP method corresponds to SQP when using $\tau_{\min} = 0$, we also show under which conditions the NLP sensitivities match those of the QP subproblem at SQP convergence in the next theorem.

**Theorem 5 (NLP and QP sensitivities coincide).** Suppose Ass. 1 holds at $z^\star$ with parameter $\bar\theta$, let $\lambda^\star, \mu^\star$ be the corresponding multipliers, and define $v^\star := (z^\star, \lambda^\star, \mu^\star)$. Regard the QP (5) which is obtained at the solution $(z^\star, \lambda^\star, \mu^\star)$ and assume an exact Hessian $Q_\star = \nabla^2_{zz}\mathcal{L}(z^\star,\lambda^\star,\mu^\star;\bar\theta)$ is used. Then, 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 at $\bar\theta$. The same holds for the solution maps of the Lagrange multipliers.

*Proof.* The KKT conditions of QP (5) read as follows:

$$Q_k\Delta z_{\mathrm{QP}} + q_k + G_k^\top\lambda_{\mathrm{QP}} + H_k^\top\mu_{\mathrm{QP}} = 0, \tag{31a}$$
$$g_k + G_k\Delta z_{\mathrm{QP}} = 0, \tag{31b}$$
$$h_k + H_k\Delta z_{\mathrm{QP}} \le 0, \tag{31c}$$
$$\mu_{\mathrm{QP},i}\cdot(h_k + H_k\Delta z_{\mathrm{QP}})_i = 0,\ \ i = 1,\dots,n_h, \tag{31d}$$
$$\mu_{\mathrm{QP}} \ge 0. \tag{31e}$$

SOSC of (2) at $(z^\star, \lambda^\star, \mu^\star)$ implies SOSC of the QP (5) at ${v_{\mathrm{QP}}}^\star = (\Delta z_{\mathrm{QP}}, \lambda^\star, \mu^\star)$, with $\Delta z_{\mathrm{QP}} = 0$. With the regularity properties in Ass. 1, it follows that this is a unique global solution of QP (5). Thus, we have $z^\star + \Delta z^{\mathrm{sol}}_{\mathrm{QP}}(\theta, v^\star) = z^\star$.

Next, we analyze the optimality conditions of the QP and show that at SQP convergence, the solution sensitivities of the NLP coincide with the ones of the exact Hessian QP subproblem. Assuming strict complementarity, the optimality conditions (31) simplify to

$$Q_k \Delta z_{\mathrm{QP}} + q_k + G_k^\top \lambda_{\mathrm{QP}} + H_{k,\mathcal{A}}^\top \mu_{\mathrm{QP},\mathcal{A}} = 0, \tag{32a}$$

$$g_k + G_k \Delta z_{\mathrm{QP}} = 0, \tag{32b}$$

$$h_{k,\mathcal{A}} + H_{k,\mathcal{A}} \Delta z_{\mathrm{QP}} = 0. \tag{32c}$$

The sensitivities of the QP solution $(\Delta z_{\mathrm{QP}}^\star, {\lambda_{\mathrm{QP}}}^\star, {\mu_{\mathrm{QP}}}^\star)$ can be computed via the IFT as in A.4, which yields

$$\begin{bmatrix} \frac{\partial \Delta z_{\mathrm{QP}}^\star}{\partial \theta}(\bar{\theta}) \\ \frac{\partial {\lambda_{\mathrm{QP}}}^\star}{\partial \theta}(\bar{\theta}) \\ \frac{\partial \mu_{\mathrm{QP},\mathcal{A}}^\star}{\partial \theta}(\bar{\theta}) \end{bmatrix} = - \begin{bmatrix} Q_k & G_k^\top & H_{k,\mathcal{A}}^\top \\ G_k & 0 & 0 \\ H_{k,\mathcal{A}} & 0 & 0 \end{bmatrix}^{-1} r_{\mathrm{sens},k}, \tag{33}$$

where

$$r_{\mathrm{sens},k} = \frac{\partial}{\partial \theta} \begin{bmatrix} Q_k \Delta z_{\mathrm{QP}} + q_k + G_k^\top \lambda_{\mathrm{QP}} + H_{k,\mathcal{A}}^\top \mu_{\mathrm{QP},\mathcal{A}} \\ g_k + G_k \Delta z_{\mathrm{QP}} \\ h_{k,\mathcal{A}} + H_{k,\mathcal{A}} \Delta z_{\mathrm{QP}} \end{bmatrix}. \tag{34}$$

At the NLP solution, the active sets of the QP and the NLP coincide, and we have $H_k = \nabla h(z^\star; \bar{\theta})^\top, G_k = \nabla g(z^\star; \bar{\theta})^\top$. If the QP uses an exact Hessian of the Lagrangian, i.e. $Q_k = \nabla_z^2 \mathcal{L}(z_k, \lambda_k, \mu_k; \bar{\theta})$, it follows that the coefficient matrices in (33) and (27) coincide at SQP convergence. Since at SQP convergence, we have $\Delta z_{\mathrm{QP}} = 0$ and $g_k = g(z^\star; \bar{\theta}), h_k = h(z^\star; \bar{\theta})$ and $\lambda_{\mathrm{QP}} = \lambda^\star, \mu_{\mathrm{QP}} = \mu^\star$, also the right-hand side of the linear systems in (33) and (27) coincide.

This shows, that at the NLP solution, the solution sensitivities of the NLP coincide with the one corresponding to the exact Hessian QP, which completes the proof of Theorem 5. □

### A.7 Regularity assumptions of original and barrier NLP

In this section, we show that the proposed barrier smoothing approach allows us to weaken Assumption 1 and that assuming strong SOSC instead of SOSC plus strict complementarity is sufficient to obtain well-defined solution sensitivities of the barrier problem. In particular, this makes the proposed method applicable to differentiate the smoothed solution map across active set changes of the original problem.

**Definition 1 (Strong second-order sufficient condition).** At a KKT point $(\bar{z}, \bar{\lambda}, \bar{\mu})$ for parameter $\bar{\theta}$, let $\mathcal{A}_+ := \{i : \bar{\mu}_i > 0\}$, $\bar{G} := \nabla_z g(\bar{z}; \bar{\theta})^\top$, and $\bar{H} := \nabla_z h(\bar{z}; \bar{\theta})^\top$. We say strong SOSC holds at $(\bar{z}, \bar{\lambda}, \bar{\mu})$ if

$$d^\top \nabla_{zz}^2 \mathcal{L}(\bar{z}, \bar{\lambda}, \bar{\mu}; \bar{\theta}) d > 0 \text{ for all } d \in K_0 \setminus \{0\}, \qquad K_0 := \{d : \bar{G} d = 0,\ \bar{H}_i d = 0 \text{ for all } i \in \mathcal{A}_+\}.$$

**Theorem 6.** Suppose at a solution $(\bar{z}, \bar{\lambda}, \bar{\mu})$ to the original NLP (2) for parameter value $\bar{\theta}$, LICQ and strong SOSC hold. Crucially, strict complementarity might *not* be satisfied.

Consider the barrier QP (10) formed at $(\bar{z}, \bar{\lambda}, \bar{\mu})$ and parameter $\bar{\theta}$, with $q_k = \bar{q} := \nabla_z f(\bar{z}; \bar{\theta})$, $Q_k = \nabla_{zz}^2 \mathcal{L}(\bar{z}, \bar{\lambda}, \bar{\mu}; \bar{\theta})$, $g_k = \bar{g}$, $G_k = \bar{G}$, $h_k = \bar{h}$, $H_k = \bar{H}$, and $\tau_{\min} = \tau > 0$. Assume its solution branch $(\Delta z^*(\tau), s^*(\tau), \lambda^*(\tau), \mu^*(\tau))$ converges to $(0, -\bar{h}, \bar{\lambda}, \bar{\mu})$ as $\tau \downarrow 0$. Note that Theorem 2 gives sufficient conditions for convergence of the nonlinear smoothed KKT branch. Then, the solution map corresponding to (10) is locally unique and differentiable with respect to $\theta$ for sufficiently small $\tau$.

*Proof.* Fix $\tau > 0$ on the assumed solution branch. By Theorem 1, it suffices to show that the barrier QP satisfies LICQ and SOSC, as strict complementarity is trivially satisfied for the equality-constrained barrier QP. LICQ of the original NLP implies that $\bar{G}$ has full row rank, and the identity block for the slack variables makes the barrier QP constraint Jacobian $\begin{bmatrix} \bar{G} & 0 \\ \bar{H} & I \end{bmatrix}$ full row rank. It remains to establish SOSC for the barrier QP for sufficiently small $\tau$.

Let $W := \nabla_{zz}^2 \mathcal{L}(\bar{z}, \bar{\lambda}, \bar{\mu}; \theta)$ and $K := \ker \bar{G}$. The tangent directions $(d_z, d_s)$ of (10) satisfy

$$\bar{G}d_z = 0, \qquad \bar{H}d_z + d_s = 0.$$

Hence each tangent direction is determined by $d \coloneqq d_z \in K$ with $d_s = -\bar{H}d$. The Lagrangian Hessian is $\mathrm{diag}(W,\, \tau S^{-2})$, where $S \coloneqq \mathrm{diag}(s^*(\tau))$. The corresponding reduced quadratic form is

$$Q_\tau(d) \coloneqq d^\top W d + \tau \sum_{i=1}^{n_h} \frac{(\bar{H}_i d)^2}{s_i^*(\tau)^2}, \qquad d \in K.$$

Thus SOSC holds exactly when $Q_\tau(d) > 0$ for every nonzero $d \in K$. By homogeneity, it suffices to check positivity on the unit sphere

$$\Sigma \coloneqq K \cap \{d : \|d\| = 1\}.$$

Let $\mathcal{A}_+ \coloneqq \{i : \bar{\mu}_i > 0\}$. Define the subspace $K_0$ and the function $\rho$ by

$$K_0 \coloneqq \{d \in K : \bar{H}_i d = 0 \text{ for all } i \in \mathcal{A}_+\},$$
$$\rho(d) \coloneqq \sum_{i \in \mathcal{A}_+} (\bar{H}_i d)^2, \qquad d \in K.$$

Then $K_0 \cap \Sigma = \{d \in \Sigma : \rho(d) = 0\}$. The remaining indices may be weakly active with zero multipliers or inactive. If $\Sigma$ is empty, then $K = \{0\}$ and SOSC holds. If $\mathcal{A}_+$ is empty, then $K_0 = K$ and strong SOSC gives $Q_\tau(d) \geq d^\top W d > 0$ for every nonzero $d \in K$. Thus, for the remainder of the proof, assume that both $\Sigma$ and $\mathcal{A}_+$ are nonempty. Since every term of the sum in $Q_\tau$ is nonnegative, dropping the indices outside $\mathcal{A}_+$ only decreases $Q_\tau$, i.e. we have

$$Q_\tau(d) \;\geq\; d^\top W d + \tau \sum_{i \in \mathcal{A}_+} \frac{(\bar{H}_i d)^2}{s_i^*(\tau)^2} \;=:\; \widehat{Q}_\tau(d).$$

So it suffices to prove that $\widehat{Q}_\tau(d) > 0$ on $\Sigma$ for $\tau$ small.

By the convergence assumption in the theorem, $\mu_i^*(\tau) \to \bar{\mu}_i > 0$ for every $i \in \mathcal{A}_+$. The barrier complementarity equation $\mu_i^*(\tau) s_i^*(\tau) = \tau$ therefore gives, for sufficiently small $\tau > 0$,

$$s_i^*(\tau) = \frac{\tau}{\mu_i^*(\tau)} = \Theta(\tau), \qquad i \in \mathcal{A}_+.$$

Let $E \coloneqq K_0 \cap \Sigma$. If $E$ is nonempty, strong SOSC and compactness imply that there is a $\sigma_E > 0$ such that $d^\top W d \geq \sigma_E$ for all $d \in E$. By continuity, choose $\delta > 0$ such that $\rho(d) < \delta$ implies $d^\top W d \geq \sigma_E/2$ on $\Sigma$. Set $\sigma \coloneqq \sigma_E/2$ in this case. If $E$ is empty, compactness gives $\min_{d \in \Sigma} \rho(d) > 0$; choose $\delta$ below this minimum and any $\sigma > 0$, such that the implication $\rho(d) < \delta \implies d^\top W d \geq \sigma$ becomes trivial.

*Case 1 (near $K_0$).* By this choice of $\delta$ and $\sigma$, $\widehat{Q}_\tau(d) \geq d^\top W d \geq \sigma > 0$ whenever $d \in \Sigma$ and $\rho(d) < \delta$.

*Case 2 (away from $K_0$).* On the compact set $\Sigma_\delta \coloneqq \{d \in \Sigma : \rho(d) \geq \delta\}$, boundedness of $W$ gives $M \geq 0$ with $d^\top W d \geq -M$. We have

$$\tau \sum_{i \in \mathcal{A}_+} \frac{(\bar{H}_i d)^2}{s_i^*(\tau)^2} \geq \frac{\tau\, \rho(d)}{\max_{i \in \mathcal{A}_+} s_i^*(\tau)^2} \geq \frac{\tau \delta}{\max_{i \in \mathcal{A}_+} s_i^*(\tau)^2} = \Omega(1/\tau) \to \infty,$$

uniformly over $\Sigma_\delta$. So there is $\bar{\tau} > 0$ such that this sum exceeds $M + 1$ for all $d \in \Sigma_\delta$, $\tau \in (0, \bar{\tau}]$, giving

$$\widehat{Q}_\tau(d) \geq -M + (M+1) = 1 > 0, \qquad d \in \Sigma_\delta,\ \tau \in (0, \bar{\tau}].$$

We have shown that $\widehat{Q}_\tau(d) > 0$ for all $d \in \Sigma$ and $\tau \in (0, \bar{\tau}]$. As $Q_\tau(d) \geq \widehat{Q}_\tau(d)$, the same holds for $Q_\tau$ and, by homogeneity, for all $d \in K \setminus \{0\}$. This is exactly SOSC for (10) at $(\Delta z^*(\tau), s^*(\tau), \lambda^*(\tau), \mu^*(\tau))$ for every $\tau \in (0, \bar{\tau}]$. □

**Example showing that original SOSC does not ensure barrier-QP SOSC.** The following example shows that ordinary SOSC for the original NLP does not by itself ensure SOSC for the barrier QP in Theorem 6. It illustrates why the proof uses strong SOSC to establish this condition. Consider an NLP with no equality constraints, $n_z = n_h = 3$, objective $f(z) = \frac{1}{2} z^\top W z$, and inequalities $h(z) = z \leq 0$. Let

$$e = (1,1,1)^\top, \qquad v = \frac{(1,-1,0)^\top}{\sqrt{2}}, \qquad u = \frac{(1,1,-2)^\top}{\sqrt{6}},$$

and define

$$W = \frac{ee^\top}{3} - vv^\top + 3uu^\top = \begin{bmatrix} 1/3 & 4/3 & -2/3 \\ 4/3 & 1/3 & -2/3 \\ -2/3 & -2/3 & 7/3 \end{bmatrix}.$$

The vectors $e/\sqrt{3}, v, u$ are orthonormal eigenvectors of $W$ with eigenvalues $1, -1, 3$, respectively.

### A.8 Example of non-continuous solution map

Let us regard the example

$$\min_x \quad (x-1)(x+1)x^2 - \theta x \tag{35a}$$

$$\text{s.t.} \quad -0.75 \leq x \leq 0.75. \tag{35b}$$

For $\theta = 0$, the problem has two local minimizers, with the same objective function value. Both local minimizers exist in a neighborhood of $\theta = 0$, we can denote them by $x^{\star,+}(\theta) > 0$ and $x^{\star,-}(\theta) < 0$. For values of $\theta > 0$, the local minimizer $x^{\star,+}$ is the global minimizer. When increasing $\theta$ further, approximately at a value of 0.54 for $\theta$, the local minimizer $x^{\star,-}$ vanishes. We visualize the solutions obtained with the NLP solver `IPOPT` (Wächter & Biegler, 2006) for different solver initialization $x_{\text{init}}$ in Figure 4. For $x_{\text{init}} = 0$, the solver converges always to the global solution, while when initializing with $x_{\text{init}} = -1$ or $x_{\text{init}} = 1$ the closest local optimizer is found, which is not necessarily a global one. At the value of $\theta$ where the local minimizer $x^{\star,-}$ vanishes, Assumption 1, in particular SOSC is not satisfied.

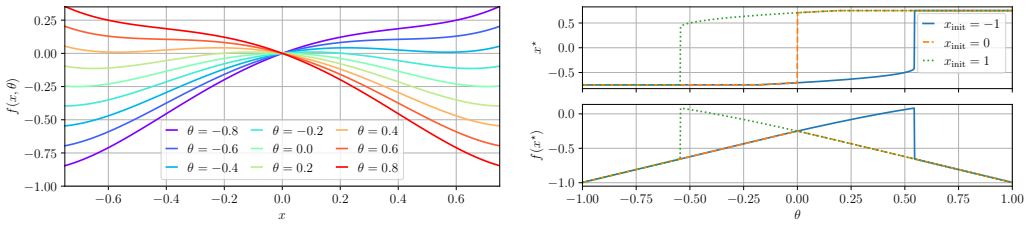


Figure 4: Example of an NLP (35) for which the global solution $x^\star(\theta)$ jumps. The objective function of (35) is visualized for different values of $\theta$ on the left. The plots on the right show the numerical solution and corresponding objective function value obtained with different solver initializations $x_{\text{init}}$.

### A.9 Differentiable `acados` layer in `leap-c`

The proposed implementation has been wrapped for integration into common ML frameworks, like `PyTorch` or `JAX`, together with additional convenience functionality to handle different kinds of parameters. This is contained in the `leap-c` project (Fichtner et al., 2025), which focuses on software for learning-enhanced control. In particular, the two-solver approach and the `acados` batch solver functionality are wrapped in the class `AcadosDiffMpcFunction`, which implements custom autograd functions in a way that is agnostic of the ML framework. On top of that is a thin `PyTorch` specific layer called `AcadosDiffMpcTorch` which is extensively tested in RL-enhanced MPC schemes. In addition to the solution sensitivities which are the focus of this paper, these classes also provide sensitivities of the value function, which can be leveraged for imitation learning (Ghezzi et al., 2023).

An example of learning-enhanced MPC using this pipeline is presented in A.14.

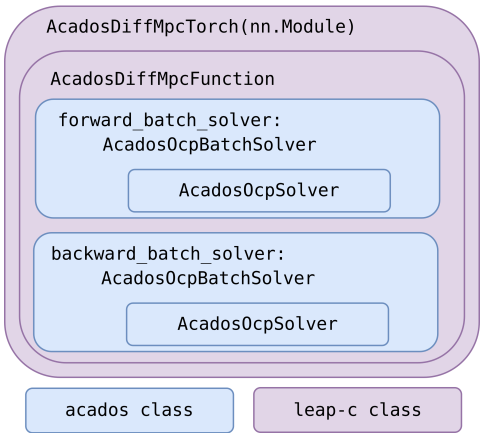


Figure 5: Schematic overview of the class hierarchy for integrating the proposed approach in common ML frameworks via `AcadosDiffMpcFunction` and specifically for `PyTorch` via `AcadosDiffMpcTorch`.

### A.10 Usage of the proposed implementation

The following code snippet shows how the proposed implementation can be used to compute forward or adjoint solution sensitivities. In particular, we refer to the steps in Section 3 and use the two-solver approach discussed in Section 4.

```
nominal_solver.solve() # (S1)
# two-solver approach iterate transfer
iterate = nominal_solver.store_iterate()
sens_solver.load_iterate_from_obj(iterate)
# (S2), (S3)
sens_solver.setup_qp_matrices_and_factorize()
# (S4f), (S5f): forward version
result = sens_solver.eval_solution_sensitivity(stages=[0], with_respect_to='p_global')
sens_u = result['sens_u']
# (S4a), (S5a): adjoint version
s_adj = sens_solver.eval_adjoint_solution_sensitivity(seed_u=[(0, np.ones((nu, 1)))])
```

Note that the forward version computes the full Jacobian (12), and the argument `stages = [0]` specifies that only the values corresponding to variables at stage 0 are unpacked in the `Python` wrapper. For the adjoint version, the seed vector $\nu$ from Sec. 2.2 is specified, by providing all non-zero entries, via the arguments `seed_u` and `seed_x`. In the snippet above, $\nu$ is specified as a vector of zeros except for the entries corresponding to $u_0$, which are set to 1. Since in this case all $\nu$ components corresponding to states are zero, `seed_x` is not specified.

### A.11 Reproducibility of the presented results

The core contribution is publicly available in the `acados` repository (Frey et al., 2025b) which is subject to the permissive 2-Clause BSD license. The code to reproduce all Figures and Tables in this paper is available in the public GitHub repository `https://github.com/FreyJo/differentiable_nmpc` with a permanant Zenodo release `https://doi.org/10.5281/zenodo.24015494`. All experiments, except for the ones corresponding to Table 3, were run on a Lenovo ThinkPad T490s with an Intel Core i7-8665U CPU and 32GB of RAM running Ubuntu 22.04.

### A.12 Details on OCP with highly nonlinear parameter dependency in Example 2

We consider an OCP, which is associated with a pendulum on cart model. The problem contains one parameter $\theta \in \mathbb{R}$, which enters the cost, dynamics and constraints of the OCP. Note that the parameter does not have a physical interpretation but is introduced for illustrative purposes only. The system is characterized by the state $x = [p, \phi, s, \omega]$, with cart position $p$, angle of the pendulum on the cart $\phi$, cart velocity $s$ and angular velocity $\omega$. The control input $u$ is a force acting on the cart in the horizontal plane and bound to be in $[-80, 80]$. The ODE describing the system dynamics can be found in Verschueren et al. (2021), with the modification that the mass of the cart $m$ is set to the parameter $\theta$. The discrete dynamics $\phi_n$ are obtained using a fourth-order Runge-Kutta integrator with constant step size $\Delta t = \frac{T}{N}$ over a prediction horizon $T = 2$ with $N = 50$ shooting intervals.

Table 3: Timings in [s] for solving $n_{\text{batch}} = 128$ bounded LQR problems with $N = 20$, $n_x = 8$, $n_u = 4$, $n_\theta = 248$. Run on a machine with an Nvidia GeForce RTX 3080 Ti GPU and an AMD Ryzen 9 5950X 16-Core CPU utilizing the GPU capabilities of `mpc.pytorch`. The results correspond to the earlier version of the benchmark with $A = \mathbb{1} + 0.2 \cdot M$, resulting in failures for `mpc.pytorch` in case of $u_{\max} = 1$.

| | **Nominal solution** | | | **Solution + adjoint sens.** | | |
|---|---|---|---|---|---|---|
| problem config | `acados` | `mpc.pytorch` | speedup | `acados` | `mpc.pytorch` | speedup |
| $u_{\max} = 10^4$ | 0.007 | 0.05 | 7.18 | 0.029 | 0.07 | 2.59 |
| $u_{\max} = 1.0$ | 0.008 | 15.12 | 1929.75 | 0.033 | 14.64 | 450.01 |

The cost function is given by

$$L_n(x_n, u_n; \theta) = \theta x_n^\top Q x_n + u_n^\top R u_n,$$
$$M(x_N; \theta) = \theta x_N^\top Q x_N$$

with weights $Q = 2 \cdot \text{diag}(10^3, 10^3, 10^{-2}, 10^{-2})$ and $R = 0.2$. In addition, we added the parametric constraint $-1.5 \leq p\theta \leq 1.5$. We fix the initial state to $(0, \frac{\pi}{2}, 0, 0)$ and solve the OCP for different $\theta$ values.

### A.13 Details and additional results on the benchmark problem in Example 3

The OCP with quadratic cost, linear discrete-time dynamics and bounds on the control inputs regarded in Example 3 can be written as

$$\min_{\substack{x_0,\ldots,x_N,\\ u_0,\ldots,u_{N-1}}} \quad \sum_{n=0}^{N-1} \begin{bmatrix} x_n \\ u_n \end{bmatrix}^\top H \begin{bmatrix} x_n \\ u_n \end{bmatrix} + x_N^\top H_x x_N \tag{36a}$$
$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{36b}$$
$$x_{n+1} = Ax_n + Bu_n + b, \ 0 \leq n < N, \tag{36c}$$
$$-u_{\max} \leq u_n \leq u_{\max}, \quad 0 \leq n < N, \tag{36d}$$

where $A = \mathbb{1} + 0.1 \cdot M$ and $B, b$ and $M$ consist of values sampled from a standard normal distribution. The cost matrix $H$ is set to the identity and $H_x$ denotes the submatrix consisting of the first $n_x$ rows and columns of $H$. The problem data $A, B, b, H$ is regarded as parameter $\theta$, such that $n_\theta = n_x^2 + n_x n_u + n_x + (n_x + n_u)^2$.

**Benchmark design choices.** While this paper focuses on constrained and nonlinear differentiable MPC, the benchmark problem considered is deliberately simple. This choice is primarily driven by limitations in the problem classes supported by existing methods; see Table 2. Restricting inequality constraints to simple control bounds facilitates a direct comparison with `mpc.pytorch`. Moreover, nonlinear problemsparticularly those with nonlinear dynamicsare not supported by `cvxpygen`.

For nonlinear problems, many solvers rely on inexact or regularized Hessian approximations, which can lead to arbitrarily inaccurate sensitivities in general. As a result, a runtime comparison would be unfair if different algorithms were allowed to employ arbitrarily coarse approximations.

Including additional software such as `diffmpc`, `trajax`, or `theseus` would require further simplification of the benchmark problem, as these tools do not support inequality constraints. Finally, a comparison with `sIPOPT` or `CasADi`s native sensitivity approach would add limited value beyond existing studies on nominal OCP solutions, which already demonstrate that structure-exploiting OCP solvers achieve substantially lower computation times, rendering general-purpose solvers non-competitive (Vanroye et al., 2023; Verschueren et al., 2021).

**Convergence of compared methods.** The code accompanying this paper shows that all solvers converge to the same primal solutions and that the adjoint solution sensitivities match.

In an earlier version of this benchmark, with more difficult dynamics, using $A = \mathbb{1} + 0.2 \cdot M$, and tight control bounds $u_{\max} = 1.0$, we obtained consistent results with `acados` and `cvxpygen`, while `mpc.pytorch` failed to converge. The convergence issues of `mpc.pytorch` can be attributed to the fact that the iLQR algorithm is based on an active-set heuristic. In particular, it is not clear how this algorithm is supposed to remove constraints from the guess of the active set (Tassa et al., 2014).

**Evaluation metrics.** We decided to only evaluate the pure solution and the solution together with adjoint sensitivity computation in the benchmark, as is reasoned next. Conceptually, one can combine two methods for the nominal solution and (adjoint) sensitivity computation in a two-solver

approach as outlined in Section A.4. However, when different software frameworks are used, this practically becomes nontrivial and introduces ambiguity into a corresponding runtime comparison. Evaluating the computation time for a sole sensitivity computation, in particular for `mpc.pytorch` and `cvxpygen` is different from evaluating it together with the solution itself, as they both reuse results obtained in the nominal solve. In particular, `mpc.pytorch` reuses the matrix factorization from the forward pass in the backward pass (Amos et al., 2018) and `cvxpygen` even uses low-rank updates to update the factorization between subsequent calls. For these reasons, the APIs of `mpc.pytorch` and `cvxpygen` are not made for solely computing sensitivities without previously solving the problem. In addition, the most common practical use-case is to use the same solver type for both steps, which makes regarding solution and sensitivity computation jointly meaningful for practitioners.

**Detailed discussion & additional results.** The speedups compared to `cvxpygen` can be attributed to different factors. Firstly, `acados` is tailored to exploit the OCP structure, while `OSQP` and the `cvxpygen` differentiator do not exploit this special problem structure. Secondly, the problems are solved to a tolerance of $10^{-6}$, while `OSQP` as a first order method is most suitable to achieve modest accuracies (Stellato et al., 2020). On the other hand, this benchmark is suitable for the `cvxpygen` differentiator, since the constraints and the Hessian do not vary between the different batch instances. This allows the `cvxpygen` differentiator to avoid a full factorization of the KKT matrix and instead perform a number of low-rank updates corresponding to the number of constraints that switch between active and inactive in subsequent batch instances, reducing the computational cost to scale quadratically in the number of variables compared to a cubic scaling for a full factorization. In particular, there are no active inequalities for $u_{\max} = 10^4$, and for $u_{\max} = 1.0$ only $\approx 7.3\%$ of the inequalities are active. For nonlinear constrained problems or nonlinear cost functions, such low-rank updates cannot be exploited, as the Hessian of the QP would always change between subsequent calls because the exact Lagrangian Hessian has to be used for sensitivity computations, as discussed in this work.

In addition to the results presented for Example 3, Table 3 presents results obtained when utilizing the GPU capabilities of `mpc.pytorch` for the same benchmark problem on a machine with an Nvidia GeForce RTX 3080 Ti GPU and an AMD Ryzen 9 5950X 16-Core CPU. The GPU table uses the same dimensions but the earlier dynamics $A = \mathbb{1} + 0.2 \cdot M$, rather than $A = \mathbb{1} + 0.1 \cdot M$ used for the current CPU benchmark. With tight control bounds, `mpc.pytorch` fails to converge in this earlier setup. The corresponding reported runtime ratios should therefore not be interpreted as speedups for successful solves. The speed differences reported in Table 3 and Table 1 are specific to the hardware and problem sizes used, and the difference in dynamics also means that the CPU/GPU comparison is not controlled. We note that the problem dimensions were chosen to represent dimensions that are typical in the context of MPC. For the loose-bound setting, the `mpc.pytorch` timings suggest no substantial speedup from GPU use; this is consistent with the findings in Adabag et al. (2026).

### A.14 RL-enhanced MPC for cart-pole swing-up

This section briefly presents an example of RL-enhanced MPC. The example is implemented using `leap-c` (Fichtner et al., 2025) with `acados` as the differentiable OCP solver. The environment is a cart-pole swing-up implementation, which is based on the classic cart-pole environment from Barto et al. (1983). Contrary to the classic environment, the pole starts hanging down and the task is to swing it up and balance it there.

To achieve this task, the agent receives a reward that is proportional to the angular distance of the pole to the upright position. The episode ends when the cart position exceeds the world bounds of $\pm 2.4$ or after 200 steps.

The MPC controller is based on the OCP formulation

$$\min_{\substack{x_0,\dots,x_N,\\ u_0,\dots,u_{N-1}}} \quad \sum_{n=0}^{N-1} L(x_n, u_n; \theta) + M(x_N; \theta) \tag{37a}$$

$$\text{s.t.} \quad x_0 = \bar{x}_0, \tag{37b}$$

$$x_{n+1} = \phi(x_n, u_n; \theta), \ 0 \le n < N, \tag{37c}$$

$$-80 \le u_n \le 80, \quad 0 \le n < N, \tag{37d}$$

$$-2.4 \le x_{1,n} \le 2.4, \quad 1 \le n < N. \tag{37e}$$

The dynamics function $\phi$ represents the discretized ODE of the system for a time interval of $0.25\text{s}$, which coincides with one step in the environment. The stage and terminal cost functions $L$, $M$ are

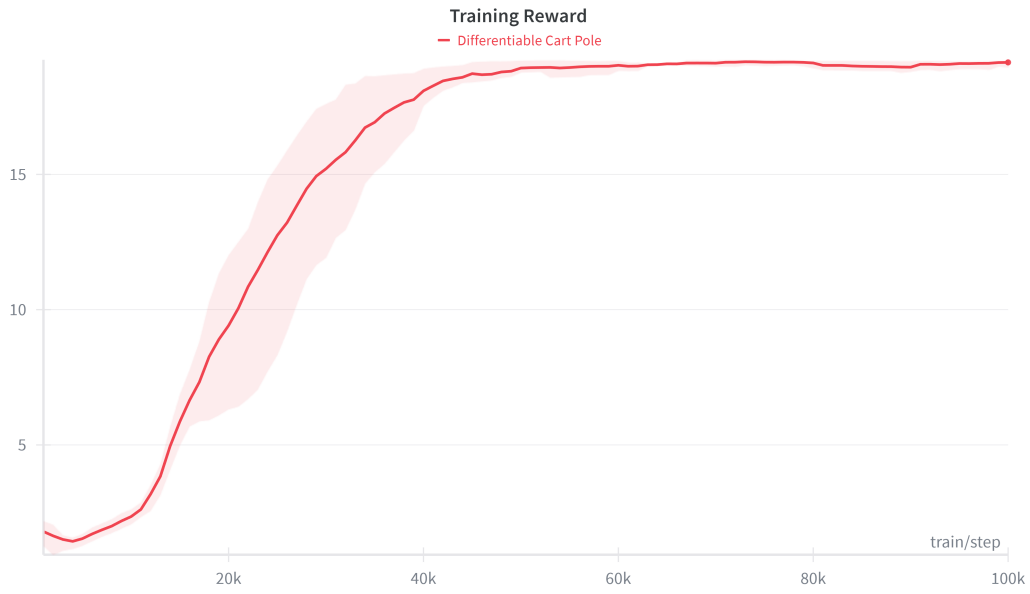


Figure 6: Training reward average and standard deviation of RL-enhanced MPC example in Sec. A.14 over five seeds.

of linear least-squares type with fixed weighting matrices, a learned reference value for the angle, and reference values zero for all other states and the control. The box constraints correspond to actuator bounds and the world bounds on the cart position. The horizon of the MPC controller is set to $N = 5$.

To learn the reference parameter, a variant of Soft Actor-Critic (Haarnoja et al., 2018) from `leap-c` is used, which in `leap-c` is called `sac-fop`. Instead of a Multilayer Perceptron (MLP) generating actions directly like in SAC, the actor in this variant consists of an MLP generating parameters for the MPC controller, which then computes the action. While training, the backpropagation uses the solution sensitivities to differentiate through the MPC controller, essentially treating the MPC controller as a final layer inside the actor, similar to Romero et al. (2024).

Figure 6 shows the evolution of the reward through the training. The reward is scaled such that one step can achieve a maximum of 0.1 reward, resulting in a maximum cumulative reward of 20 over one episode, hence it can be seen that the agent learns to successfully balance the pole. Note that due to the short horizon, even with the reference angle fixed to 0, the nominal MPC controller cannot balance the pole properly. This demonstrates that the proposed approach can be effectively integrated into a learning pipeline and achieves substantially better performance than MPC alone.

The results can be reproduced by running the script in the `leap-c` repository[2].

## Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Introduction state three contributions, each supported by the analysis, implementation, and experiments cited below.

   - **Claim 1:** "First, this work discusses the efficient computation of parametric (adjoint) solution sensitivities for general nonlinear programs (NLP) via the implicit function theorem (IFT). In particular, this work highlights common pitfalls of the IFT and discusses their mitigation. Previous work in the machine learning (ML) literature was

[2] `https://github.com/leap-c/mpc-sac/blob/d38726d76c9477274f7a588fe27a24e40120d897/scripts/run_sac_fop.py`

subject to this pitfall, or limited to convex or unconstrained nonlinear least squares problems."
**Justification:** Section 2 states the regularity assumptions and develops the IFT-based sensitivity analysis for general NLPs. Remark 1 and Example 2 show how an approximate Hessian can degrade sensitivities. The related-work discussion and Table 2 support the stated comparison with prior methods.

- **Claim 2:** "Second, we introduce SBQP, which generalizes SQP with an IPM QP solver by allowing the barrier parameter to approach a prescribed nonnegative target value. We detail how IPMs or SBQP can be used to obtain a differentiable approximation of the solution map and its sensitivities suitable for gradient-based learning algorithms by leveraging the smoothing of the optimality conditions that these methods employ."
  **Justification:** Section 2.1 defines SBQP and its zero-target case. Theorems 1, 2, and 3 in Section 2.2 establish the local sensitivity and smoothing results under their stated assumptions. Section 2.2 discusses the relevance of smoothed sensitivities to learning, and Section A.14 demonstrates their use in an RL example.
- **Claim 3:** "Finally, we present an efficient implementation of a differentiable solver for optimal control structured NLPs within the open-source software package `acados` which provides both forward and adjoint sensitivities. In particular, the optimal control problem (OCP) formulation covers general, potentially nonlinear, costs and constraints on both states and controls while allowing all problem functions, i.e., costs, constraints and dynamics, to be fully parametric. A comparison with state-of-the-art solvers demonstrates speedup factors greater than three on a CPU."
  **Justification:** Sections 3 and 4 describe the implementation, and Examples 2 and 4 demonstrate parametric nonlinear OCPs and sensitivity computation. Example 3 and Table 1 support the CPU speedup claim for the evaluated linear-quadratic OCP benchmark with simple input bounds; the reported speedups are specific to this benchmark.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of the approach presented in the paper can be structured into fundamental theoretical ones and ones that concern the proposed implementation.
   The theoretical limitations that need to hold for the existence of the solution sensitivities of NLP 2 are given in Assumption 1. Example 1 showcases that this assumption is not satisfied at active-set changes, but that reasonable results can be obtained by the smoothing employed by an interior-point method. The example in Appendix A.8 shows the fundamental issue of general nonconvex NLPs with multiple local minima and solution maps that exhibit jumps. Their treatment in practice is not part of this work and both examples violate the conditions in Assumption 1.
   The limitations of the proposed implementation are given in the paragraph "Limitations" in Section 4.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The main text presents three theorems in Section 2.2; the Appendix also states the IFT and the NLP/QP sensitivity result. The assumptions are stated alongside the results or referenced through Assumption 1. The Appendix proves Theorem 1, Theorem 3, and Theorem 5. Theorem 2 follows from Properties 1 and 3 in Pirnay et al. (2012), and a comprehensive proof is given alongside Theorem 8 in Wright (1992). The IFT is stated in the Appendix in its classical form and cited to Dini (1907).

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All information needed to reproduce the figures and tables in this paper is contained in the code in the supplemental material as described in Section A.11.

Guidelines:

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code to reproduce the figures and tables in this paper is available in the supplemental material as described in Section A.11.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All code, (including test details, parameter choices etc.) to reproduce the figures and tables in this paper is available in the supplemental material as described in Section A.11.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [N/A]

   Justification: The paper does not contain experiments for which this question is relevant.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The experiments presented in this paper can be run on a modern laptop. The runtime of the experiment in Example 3 is roughly 30 minutes. The runtime for the Examples 4 and 2 is below 5 minutes.

   The runtime for the example in Section A.14 is around 10 hours on a modern laptop. This example only shows how the building block of differentiable NMPC can be used in a learning framework, while this is not the focus of this work.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: After carefully reading the guidelines, the authors do not see any conflicts with the NeurIPS Code of Ethics.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: This work has the potential of advancing learning-enhanced MPC by combining explicit constraint handling with solution sensitivity computation. As previous related works mostly relied on heuristic constraint treatment, this can reduce operational risks and ensure that controllers respect physical and safety-critical limits. While the methods may be applied broadly, including in contexts without direct societal benefit, their impact ultimately depends on responsible deployment, highlighting the dual-use nature of the technology.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The answer NA means that the paper poses no such risks.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper only uses open-source software as dependencies. All assets used are properly credited to the best of our knowledge.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The core software contribution is publicly available in the `acados` repository Frey et al. (2025b) which is subject to the permissive 2-Clause BSD license. Its documentation is publicly available on `https://docs.acados.org/`. In particular, the functions introduced and showcased in this work, see Section A.10 are contained in the documentation.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The answer NA means that the paper does not involve crowdsourcing nor research with human subjects

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.