BP-MPC: Optimizing the Closed-Loop Performance of MPC Using Backpropagation

Riccardo Zuliani, *Graduate Student Member, IEEE*, Efe C. Balta, *Member, IEEE*, and John Lygeros, *Fellow, IEEE*

***Abstract*—Model predictive control (MPC) is pervasive in research and industry. However, designing the cost function and the constraints of the MPC to maximize closed-loop performance remains an open problem. To achieve optimal tuning, we propose a backpropagation scheme that solves a policy optimization problem with nonlinear system dynamics and MPC policies. We enforce the system dynamics using linearization and allow the MPC problem to contain elements that depend on the current system state and on past MPC solutions. Moreover, we propose a simple extension that can deal with losses of feasibility. Our approach, unlike other methods in the literature, enjoys convergence guarantees.**

***Index Terms*—Backpropagation, differentiable optimization, model predictive control (MPC), policy optimization.**

I. Introduction

Among optimization-based control schemes, *model predictive control* (MPC) has recently attracted increasing attention in both industry and academia. This technique enables feedback by repeatedly solving a numerical optimization problem at every time step, each time taking into account the current (measured or estimated) state of the system as well as process and input constraints. Because of its effectiveness in practical applications, researchers have dedicated significant effort to the task of designing MPC controllers. For example, Chen and Allgöwer [1] showed that the introduction of an appropriately selected terminal cost can ensure stability and feasibility of the closed loop. More recently, Cairano and Bemporad [2] proposed a design to ensure that the MPC behaves like a linear controller around a specified operating point, with the goal of inheriting the well-known stability and robustness properties of linear controllers. The objective function of an MPC can also be chosen to incentivize learning of an unknown model, as proposed in [3].

Received 15 December 2024; accepted 16 February 2025. Date of publication 25 February 2025; date of current version 29 August 2025. This work was supported by the Swiss National Science Foundation under NCCR Automation under Grant 51NF40_180545. Recommended by Associate Editor T. Faulwasser. *(Corresponding author: Riccardo Zuliani.)*

Riccardo Zuliani and John Lygeros are with the Automatic Control Laboratory (IfA), ETH Zürich, 8092 Zürich, Switzerland (e-mail: rzuliani@ethz.ch; lygeros@ethz.ch).

Efe C. Balta is with the Inspire AG, 8005 Zürich, Switzerland, and also with Automatic Control Laboratory (IfA), ETH Zürich, 8092 Zürich, Switzerland (e-mail: efe.balta@inspire.ch).

Digital Object Identifier 10.1109/TAC.2025.3545767

MPC design can be viewed as a policy optimization problem. *Policy optimization* is a well-known problem in reinforcement learning, where the goal is to obtain a control policy that minimizes some performance objective [4]. In common applications, the policy is parameterized by problem parameters, states, or inputs, and gradient-based techniques are used to learn the optimal parameters. In the context of MPC, the design parameters are generally the cost and the constraints of the problem. The challenge when considering MPC policies is that the MPC policy and resulting closed-loop performance are generally not differentiable with respect to the parameters.

Recently, differentiable optimization provided a principled way to overcome the nondifferentiability issue. Specifically, Amos and Kolter [5] proved that under certain conditions, the optimizer of a quadratic program (QP) is indeed continuously differentiable with respect to design parameters appearing in the cost and the constraints, and that the gradient can be retrieved by applying the implicit function theorem [6] to the KKT conditions of the QP. Since MPC problems are often formulated as QPs, this approach effectively allows for the differentiation of MPC policies. This discovery led to a plethora of applications of differentiable optimization in the realm of MPC. For example, Amos et al. [7] considered the problem of imitation learning, where the tuning parameters are the cost and the model of the linear dynamics of an MPC problem. The idea of utilizing the KKT conditions to obtain derivatives of an optimization problem does not stop with QPs. Oshin and Theodorou [8] used the same technique to compute gradients of a nonlinear optimal control problem and used this information to conduct online design of a robust model predictive controller. The goal in this case is to match the performance of a nominal controller. In [9], a predictive safety filter is used to ensure the safety of the closed-loop operation.

Similarly, over the course of several papers [10], [11], [12], the authors used policy gradient methods to optimize the performance of nonlinear economic MPC. The main idea behind these works is to utilize a nonlinear MPC as a function approximator that can encode both the value and the action-value functions of a given problem. Their algorithm can produce MPC schemes that are stabilizing by construction [10] and safe against additive disturbances (in the case of affine systems) [11]. In particular, the work that we believe closest to ours is [12], as it uses a linearization-based procedure to avoid solving a nonconvex problem online. In these references, however, the authors focus on infinite horizon problems and do not provide a detailed treatment of the convergence of the optimization algorithm. They do provide sufficient conditions under which the nonlinear MPC map is continuously differentiable, but ensuring this condition requires solving an LMI every time the MPC gets updated.

These sophisticated differentiable optimization-based methods rely on the assumption that the optimizer of the MPC problem is continuously differentiable, since the gradient of the optimizer is obtained using the implicit function theorem. It is well known, however, that this may not be the case and that the optimizer may not be everywhere differentiable even for simple projection problems [13]. The continuous differentiability assumption can be relaxed, thanks to the recently developed concept of conservative Jacobians [14]. *Conservative Jacobians* are set-valued operators that extend the concept of gradients to almost-everywhere differentiable functions. Like other generalized Jacobians, they obey the chain rule of differentiation and can be used to create first-order optimization schemes with convergence guarantees [15]. However, unlike, for example, Clarke Jacobians, conservative Jacobians satisfy a nonsmooth implicit function theorem, which is essential in our setting to obtain the sensitivity of the solution maps of the MPC problems [16].

We consider the problem of optimizing the closed-loop trajectory directly by backpropagation. Specifically, we compute the conservative Jacobian of the entire closed-loop trajectory with respect to variations of the design parameters by applying the chain rule to the conservative Jacobians of each MPC problem. We then apply a gradient-based scheme to update the value of the parameter to improve closed-loop performance. This is fundamentally different to optimizing a single MPC step as it accounts for the effect of the receding horizon, where past decisions influence future ones.

The idea of using backpropagation to improve closed-loop performance of MPC first appeared in [17] and [18]. These studies focused on linear dynamics without state constraints and did not provide formal convergence guarantees. Our work makes the following contributions.

1) We utilize the backpropagation paradigm to solve a nonconvex closed-loop policy optimization problem where the policy is a parameterized MPC. The MPC utilizes a linearized version of the system dynamics to retain convexity.
2) We provide conditions under which the closed-loop optimization problem is well posed by extending [5] to the nonsmooth regime and propose a gradient-based method that converges to a critical point.
3) We allow the MPC to have cost and constraints that depend on the current state of the system and on the solution of the MPC problem in the previous time steps.
4) We propose a simple extension to deal with cases where the MPC scheme loses feasibility and provide conditions under which the closed loop is guaranteed to converge to a safe operation.

To compute the conservative Jacobian of each optimization problem, we adapt and extend the techniques described in [19] to a control theoretic context. In addition, we derive problem-specific sufficient conditions under which the nonsmooth implicit function theorem in [19] can be applied. We finally showcase our findings through simulation on a nonlinear problem.

Our algorithm can be applied under the assumption that the initial condition of the system is known, as is the case, for example, for iterative control tasks. Our approach has been recently extended in [20] to uncertain systems subject to additive noise and uncertain initial conditions.

The rest of this article is organized as follows. Section II describes the system dynamics, the control policy, and the policy optimization problem. Section III presents a short recap of conservative Jacobians, their main calculus rules, and a way to minimize such functions with a first-order scheme. Section IV demonstrates how the conservative Jacobian of an MPC problem can be computed. Section V showcases our main algorithmic contribution by describing the backpropagation scheme and the main optimization algorithm. In Section VI we provide some useful extensions to our scheme, such as nonlinear dynamics and recovery from infeasibility. In Section VII, we showcase our methods in simulation. Finally, Section VIII concludes this article.

Notations: We use $\mathbb{N}$, $\mathbb{Z}$, and $\mathbb{R}$ to denote the set of natural, integer, and real numbers, respectively. $\mathbb{Z}_{[a,b]}$ is the set of integers z with $a \leq z \leq b$, for some $a \leq b$. If $C \subset \mathbb{R}^n$ is a convex set, we denote with P_C the orthogonal projector to the set. Given a matrix $A \in \mathbb{R}^{n\times m}$, we use $\mathbf{r}(A, j)$ to denote the jth row of A (with $j \in \mathbb{Z}_{[1,n]}$). We use $A \succ 0$ ($A \succeq 0$) to indicate that the symmetric matrix A is positive definite (positive semidefinite). We use $A \sim B$ to indicate that A is a function of B. $\|\cdot\|$ denotes the 2-norm, and $\langle a, b\rangle = a^\top b$ is the Euclidean inner product.

II. Problem Formulation

A. System Dynamics and Constraints

We consider a nonlinear time-invariant system where the state dynamics are given for each time-step $t \in \mathbb{N}$ by

$$\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t) \tag{1}$$

with f locally Lipschitz and $\bar{x}_0 \in \mathbb{R}^{n_x}$ known. We assume that $(\bar{x}_t, \bar{u}_t) = 0$ is an equilibrium for (1). The *state* $\bar{x}_t \in \mathbb{R}^{n_x}$ and the *input* $\bar{u}_t \in \mathbb{R}^{n_u}$ are subject to polytopic constraints

$$H_x\bar{x}_t \leq h_x,\ H_u\bar{u}_t \leq h_u. \tag{2}$$

The control input $\bar{u}_t$ is determined, at each time-step, by a parameterized control policy $\pi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$

$$\bar{u}_t = \pi(\bar{x}_t, p). \tag{3}$$

The *parameter vector* $p \in \mathbb{R}^{n_p}$ parameterizes the control policy π at any state $\bar{x}_t$ (for example, in Section II-B, p represents the terminal state cost and the input cost of an MPC). We require p to satisfy the constraint $p \in \mathscr{P}$, for some polytopic set $\mathscr{P}$. Below, we restrict attention to MPC control policies.

The goal of this article is to minimize an objective function involving p and the closed-loop state and input trajectory

$(\bar{x}, \bar{u}) := (\bar{x}_0, \ldots, \bar{x}_{T+1}, \bar{u}_0, \ldots, \bar{u}_T)$ for some finite time interval $T \in \mathbb{N}_{>0}$, under the constraints in (2)

$$\begin{aligned} \underset{\bar{x}, \bar{u}, \bar{p} \in \mathscr{P}}{\text{minimize}} \quad & \mathcal{C}(\bar{x}, \bar{u}, p) \\ \text{subject to} \quad & \bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t),\ \bar{x}_0 \text{ given} \\ & \bar{u}_t = \pi(\bar{x}_t, p) \\ & H_x \bar{x}_t \leq h_x,\ H_u \bar{u}_t \leq h_u,\ t \in \mathbb{Z}_{[0,T]} \end{aligned} \tag{4}$$

where $\mathcal{C} : \mathbb{R}^{(T+2)n_x} \times \mathbb{R}^{(T+1)n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}_{\geq 0}$ specifies the performance objective. In (4), T should be chosen large enough to reach the desired equilibrium condition. Note that problem (4) may be nonconvex. In the following, for simplicity, we consider the case

$$\mathcal{C}(\bar{x}, \bar{u}, p) = \mathcal{C}(\bar{x}) = \sum_{t=0}^{T} \|\bar{x}_t\|^2_{Q_x} \tag{5}$$

for some $Q_x \in \mathbb{R}^{n_x \times n_x}$ with $Q_x \succ 0$. Our method can easily be extended to more general cost functions as described in Section VI-C.

B. Model Predictive Control (MPC)

In this article, we restrict attention to MPC policies, where the control input is chosen as the solution of an optimal control problem. Specifically, after measuring the current state $\bar{x}_t$, we use the knowledge we possess about the system (1) to optimize the future prediction of the state-input trajectories of the system. The predicted trajectories are denoted by $x_t := (x_{0|t}, \ldots, x_{N|t}) \in \mathbb{R}^{(N+1)n_x}$ and $u_t := (u_{0|t}, \ldots, u_{N-1|t}) \in \mathbb{R}^{Nn_u}$, where $N \in \mathbb{N}_{>0}$, with $N \ll T$, is the *prediction horizon* of the MPC. The initial state is chosen to be equal to the true state of the system, $x_{0|t} = \bar{x}_t$ and, to ensure convexity, we approximate the state dynamics as

$$x_{k+1|t} = A_t x_{k|t} + B_t u_{k|t} + c_t$$

where A_t, B_t, and c_t are known at runtime and should be chosen to accurately approximate the real dynamics (1) in the vicinity of $\bar{x}_t$. We use $S_t := (A_t, B_t, c_t)$ to compactly represent the approximate dynamics at time t.

Each predicted state and input must satisfy the constraints (2). In addition, we generally impose different constraints on the predicted terminal state $x_{N|t}$

$$H_{x,N} x_{N|t} \leq h_N. \tag{6}$$

The objective function in the MPC is an approximation of the objective in (4), given by

$$\|x_{N|t}\|^2_P + \sum_{k=0}^{N-1} \|x_{k|t}\|^2_{Q_x} + \|u_{k|t}\|^2_{R_u}$$

where we added a terminal penalty $\|x_{N|t}\|^2_P$ and a penalty on the input, with $P, R_u \succ 0$, to ensure that the problem is strongly convex. The MPC problem that is solved online at each time-step is therefore given as follows:

$$\begin{aligned} \min_{x_t, u_t} \quad & \|x_{N|t}\|^2_P + \sum_{k=0}^{N-1} \|x_{k|t}\|^2_{Q_x} + \|u_{k|t}\|^2_{R_u} \\ \text{s.t.} \quad & x_{k+1|t} = A_t x_{k|t} + B_t u_{k|t} + c_t,\ x_{0|t} = \bar{x}_t \\ & H_x x_{k|t} \leq h_x,\ H_u u_{k|t} \leq h_u,\ k \in \mathbb{Z}_{[0,N-1]} \\ & H_{x,N} x_{N|t} \leq h_{x,N}. \end{aligned} \tag{7}$$

At each time-step, after measuring $\bar{x}_t$ and obtaining S_t, we solve (7) and choose $\pi(\bar{x}_t, p) = u_{0|t}$, where $u_{0|t}$ is the first entry of the input trajectory. We use $\text{MPC}(\bar{x}_t, S_t, p)$ to denote the function that maps a parameter p, a nominal system S_t, and an initial condition $\bar{x}_t$ to a control input $\bar{u}_t = u_{0|t}$, so that

$$\pi(\bar{x}_t, p) = \text{MPC}(\bar{x}_t, S_t, p). \tag{8}$$

Here, we treat the terminal cost and the input cost as tunable parameters, by letting $p := (P, R_u)$. However, with the same framework, one can also choose p as any other element appearing in the cost or in the constraints of (7).

C. Projected Gradient-Based Framework

For the time being, we assume that $S_t \equiv S$ and drop it from the notation; we deal with the more complex case where S_t is determined online in Section VI-A. Combining problem (4) with the cost function (5) and the controller (8) leads to the closed-loop control problem

$$\begin{aligned} \underset{\bar{x}, p \in \mathscr{P}}{\text{minimize}} \quad & \sum_{t=0}^{T} \|\bar{x}_t\|^2_{Q_x} \\ \text{subject to} \quad & \bar{x}_{t+1} = f(\bar{x}_t, \text{MPC}(\bar{x}_t, p)),\ \bar{x}_0 \text{ given} \\ & H_x \bar{x}_t \leq h_x,\ t \in \mathbb{Z}_{[0,T]}. \end{aligned} \tag{9}$$

Note that we can remove the input constraints in (2), as they are automatically satisfied if the inputs are obtained from the MPC (7). As shown in Appendix A, (9) can be compactly rewritten as follows:

$$\begin{aligned} \underset{p \in \mathscr{P}}{\text{minimize}} \quad & \mathcal{C}(\bar{x}(p)) \\ \text{subject to} \quad & \mathcal{H}(\bar{x}(p)) \leq 0 \end{aligned} \tag{10}$$

where $\bar{x}(p)$ is the closed-loop state trajectory generated by the dynamics (1) under controller (8) for a given value of p. In the following section, we derive an efficient procedure to obtain generalized gradients of the function $\mathcal{C}$ with respect to p.

III. Conservative Jacobians

In the upcoming sections, we repeatedly deal with the problem of minimizing a nonsmooth, nonconvex function. These problems admit a simple solution strategy based on a descent algorithm. However, because of the nonsmoothness, we cannot always guarantee the existence of a gradient. Luckily, we can still devise descent algorithms if the function is almost everywhere differentiable, thanks to the concept of *conservative Jacobian*. This section describes how conservative Jacobians generalize the notion of gradient to functions that are almost everywhere differentiable.

A *path* is an absolutely continuous function $x : [0,1] \to \mathbb{R}^n$, which admits a derivative $\dot{x}$ for almost every $t \in [0,1]$, and for which $x(t) - x(0)$ is the Lebesgue integral of $\dot{x}$ between 0 and t for all $t \in [0,1]$.

Definition 1 (See [16, Sect. 2]): A locally Lipschitz function $\varphi : \mathbb{R}^n \to \mathbb{R}^m$ admits $\mathcal{J}_\varphi : \mathbb{R}^n \rightrightarrows \mathbb{R}^{m\times n}$ as a *conservative Jacobian*, if $\mathcal{J}_\varphi$ is nonempty-valued, outer semicontinuous, locally bounded, and for all paths $x : [0,1] \to \mathbb{R}^n$ and almost all $t \in [0,1]$

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(x(t)) = \langle v, \dot{x}(t)\rangle \quad \forall v \in \mathcal{J}_\varphi(x(t)). \tag{11}$$

A locally Lipschitz function φ that admits a conservative Jacobian $\mathcal{J}_\varphi$ is called *path-differentiable*.

If $\varphi : \mathbb{R}^n \times \mathbb{R}^p \to \mathbb{R}^n$ is a function of two arguments p and x, we define $\mathcal{J}_{\varphi,x}(\tilde{p},\tilde{x}) = \{V : [U\, V] \in \mathcal{J}_\varphi(\tilde{p},\tilde{x})\}$ as the conservative Jacobian of φ with respect to x (and similarly for $\mathcal{J}_{\varphi,p}(\tilde{p},\tilde{x})$). Note that $\mathcal{J}_{\varphi,x}(\tilde{p},\tilde{x})$ and $\mathcal{J}_{\varphi,p}(\tilde{p},\tilde{x})$ are obtained through the projections of the conservative Jacobian $\mathcal{J}_\varphi$ onto the p and x coordinates.

Conservative Jacobians extend the concept of gradient to nonsmooth almost everywhere differentiable functions. If φ is such a function, the conservative Jacobian $\mathcal{J}_\varphi$ coincides with the gradient $\nabla_x \varphi(x)$ almost everywhere [14, Thm. 1]; moreover, if φ is convex, then the standard subdifferential ∂f is a conservative Jacobian for φ.

The most useful property that conservative Jacobians possess is that they admit the chain rule (11). This immediately implies the following composition rule.

Lemma 1 (See [14, Lemma 6]): Given two path-differentiable functions $\varphi : \mathbb{R}^{n_x} \to \mathbb{R}^{n_p}$, $\psi : \mathbb{R}^{n_p} \to \mathbb{R}^{n_u}$, with conservative Jacobians $\mathcal{J}_\varphi$ and $\mathcal{J}_\psi$, the function $\varphi \circ \psi$ is path-differentiable with conservative Jacobian $\mathcal{J}_{\varphi\circ\psi}(x) = \mathcal{J}_\varphi(\psi(x))\mathcal{J}_\psi(x)$.

Unfortunately, not every locally Lipschitz function is path-differentiable. For this reason, we restrict our attention to *definable functions*.

Definition 2 (See [21, Def. 1.4 and 1.5]): A collection $\mathcal{O} = (\mathcal{O}_n)_{n\in\mathbb{N}}$, where each $\mathcal{O}_n$ contains subsets of $\mathbb{R}^n$, is an *o-minimal structure* on $(\mathbb{R}, +, \cdot)$ if the following holds.

1) All semialgebraic subsets of $\mathbb{R}^n$ belong to $\mathcal{O}_n$.
2) The elements of $\mathcal{O}_1$ are precisely the finite unions of points and intervals.
3) $\mathcal{O}_n$ is a boolean subalgebra of the powerset of $\mathbb{R}^n$.
4) If $A \in \mathcal{O}_n$ and $B \in \mathcal{O}_m$, then $A \times B \in \mathcal{O}_{n+m}$.
5) If $A \in \mathcal{O}_{n+1}$, then the set containing the elements of A projected onto their first n coordinates belongs to $\mathcal{O}_n$.

A subset of $\mathbb{R}^n$ that belongs to $\mathcal{O}$ is said to be *definable* (in the o-minimal structure). A function $\varphi : \mathbb{R}^n \to \mathbb{R}^p$ is *definable* if its graph $\{(x,v) : v = \varphi(x)\}$ is definable.

Definable functions possess the following useful property.

Lemma 2 (See [14, Prop. 2]): All locally Lipschitz definable functions are path-differentiable.

Another crucial property of locally Lipschitz definable functions is that obey a nonsmooth version of the implicit function theorem.

Algorithm 1: Minimization of Path-Differentiable Functions.

Input: x^0, $\{\alpha_k\}_{k\in\mathbb{N}}$.
Init: $k = 0$.
1: **while** not converged **do**
2: $\quad p^k \in \mathcal{J}_\varphi(x^k)$
3: $\quad x^{k+1} = P_{\mathscr{P}}[x^k - \alpha_k p^k]$
4: $\quad k \leftarrow k+1$
5: **end while**

Lemma 3 (See [22, Thm. 5]): Let $\varphi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ be a locally Lipschitz definable function and let $\mathcal{J}_\varphi$ be its conservative Jacobian. Suppose $\varphi(\tilde{x},\tilde{p}) = 0$ for some $\tilde{x} \in \mathbb{R}^{n_x}$ and $\tilde{p} \in \mathbb{R}^{n_p}$. Assume that $\mathcal{J}_\varphi$ is convex and that for every $[U\, V] \in \mathcal{J}_\varphi(\tilde{x},\tilde{p})$, the matrix U is invertible. Then, there exists a neighborhood $\mathcal{N}(\tilde{x}) \times \mathcal{N}(\tilde{p})$ of $(\tilde{x},\tilde{p})$ and a path differentiable definable function $x : \mathcal{N}(\tilde{p}) \to \mathcal{N}(\tilde{x})$ such that for all $p \in \mathcal{N}(\tilde{p})$ it holds that $\varphi(x(p),p) = 0$, and the conservative Jacobian $\mathcal{J}_x$ of x is given for all $p \in \mathcal{N}(\tilde{p})$ by

$$\mathcal{J}_x(p) = \{[-U^{-1}V] : [U\, V] \in \mathcal{J}_\varphi(x(p),p)\}.$$

Path-differentiable definable functions can be minimized using simple projected-gradient based scheme as outlined in Algorithm 1.

Here, we used $P_{\mathscr{P}}$ to denote the projector to the set $\mathscr{P}$. Typically, we stop the algorithm, e.g., when $\|p^k - p^{k-1}\| < \texttt{tol}$ for some positive tolerance `tol`, or after exceeding a certain number of iterations. The following results demonstrate that, under certain conditions on the step-size $\{\alpha_k\}_{k\in\mathbb{N}}$, Algorithm 1 is guaranteed to converge to a critical point of φ.

Lemma 4 (See [15, Thm. 6.2]): Assume that φ is path-differentiable and definable, that $\mathscr{P}$ is a polytopic set, that the stepsizes $\{\alpha_k\}_{k\in\mathbb{N}} \subset \mathbb{R}_{>0}$ satisfy

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \quad \sum_{k=0}^{\infty} \alpha_k^2 < \infty \tag{12}$$

and that $\sup_k \|x^k\| < \infty$. Then, x^k as obtained via Algorithm 1 converges to a critical point of φ, i.e., a point $\tilde{x}$ for which $0 \in \mathcal{J}_\varphi(\tilde{x})$.

The assumption on the boundedness of the conservative Jacobians is not restrictive in practice and it is generally satisfied under a suitable stepsize choice, or if $\mathscr{P}$ is a bounded set. Readers should refer to [15] for more details.

IV. Differentiating the MPC Policy

In this section, we rewrite (7) in a more convenient form and then show that, under certain conditions, the map $\mathrm{MPC}(\bar{x}_t, p)$ admits a conservative Jacobian, leading to a descent algorithm for (9).

A. Writing the MPC Problem as a QP

Problem (7) can be reformulated as a QP in standard form (see, e.g., [23, Sect. III])

$$\underset{y}{\text{minimize}}\ \frac{1}{2}y^\top Q(p)y + q(\bar{p})^\top y$$

$$\begin{aligned}\text{subject to}\quad & F(p)y = \phi(\bar{p})\\ & G(p)y \leq g(\bar{p})\end{aligned}\tag{13}$$

where $\bar{p} := (\bar{x}_t, p)$. We denote with n_{in} and n_{eq} the number of inequality and equality constraints in (13), respectively. Note that parameter p can potentially affect all terms in the cost and in the constraints, whereas initial condition $\bar{x}_t$ can only affect the linear part of the cost and the affine term in the constraints.

To simplify the computation of the conservative Jacobian, we operate on the Lagrange dual problem associated with (13). In this case, the constraints are parameter-independent, as $\bar{p}$ only affects the cost function of the problem. To ensure that the dual problem has a unique solution for every value of $\bar{p}$, we impose the following assumption.

Assumption 1: For all parameter vectors $\bar{p}$ in some polytopic set $\mathscr{Y}$, the matrix $Q(p)$ in (13) is positive definite, and problem (13) is feasible and satisfies the linear independence constraint qualification (LICQ).

Recall that (13) satisfies the LICQ if given an optimizer $y(\bar{p})$, the rows of G associated with the active inequality constraints and the rows of F are linearly independent. Note that the LICQ assumption holds if, for example, the constraints on $x_{k|t}$ and $u_{k|t}$ in (7) are simple box constraints

$$x_{\min} \leq x_{k|t} \leq x_{\max}, \quad u_{\min} \leq u_{k|t} \leq u_{\max}$$

for some $x_{\min}, x_{\max} \in \mathbb{R}^{n_x}$, $x_{\min} < x_{\max}$, $u_{\min}, u_{\max} \in \mathbb{R}^{n_u}$, and $u_{\min} < u_{\max}$.

The feasibility condition in Assumption 1 can be restrictive in practical scenarios. We propose a simple extension of our method that can deal with losses of feasibility in Section VI-D.

Under Assumption 1, we can obtain the Lagrange dual of (13) following the procedure outlined in Appendix B:

$$\begin{aligned}\text{minimize}_z\quad & \frac{1}{2} z^\top H(\bar{p}) z + h(\bar{p})^\top z\\ \text{subject to}\quad & Ez \geq 0.\end{aligned}\tag{14}$$

Note that in (14), the parameters p and $\bar{x}_t$ only affect the quadratic part H and the linear part h of the cost, whereas the matrix E in the constraints is parameter-independent.

The primal solution $y(\bar{p})$ can be obtained from the dual solution $z(\bar{p}) = (\lambda(\bar{p}), \mu(\bar{p}))$ as

$$y(\bar{p}) = -Q(p)^{-1}(F(p)^\top \mu(\bar{p}) + G(p)^\top \lambda(\bar{p}) + q(\bar{p})).\tag{15}$$

B. Writing the Dual as Fixed-Point Condition

To obtain the conservative Jacobian $\mathcal{J}_z$ of the dual optimizer, we follow the procedure proposed in [19]: we write the optimality conditions of (14) as a fixed-point equation $\mathcal{F}(z, \bar{p}) = 0$, obtain the conservative Jacobian of $\mathcal{F}$ with respect to z and $\bar{p}$, and apply the implicit function theorem in Lemma 3. Since (14) is a QP, a necessary and sufficient condition for optimality [24, Thm. 3.67] is

$$0 \in H(\bar{p})z + h(\bar{p}) + N_C(z)\tag{16}$$

where N_C is the normal cone mapping of $C := \{z \in \mathbb{R}^{n_z} : Ez \geq 0\}$, with $n_z = n_{\text{in}} + n_{\text{eq}}$ [24, Example 3.5]. Leveraging [25, Corollary 27.3], we have that (16) is equivalent to

$$0 \in P_C[z - \gamma(H(\bar{p})z + h(\bar{p}))] - z =: \mathcal{F}(z, \bar{p})\tag{17}$$

where $\gamma \in \mathbb{R}_{>0}$ is a positive scalar and $P_C : \mathbb{R}^{n_z} \to C$ is the projection operator to the set C. To ensure the existence of the conservative Jacobian of $\mathcal{F}$, we impose the following assumption.

Assumption 2: The maps $Q(p)$, $q(\bar{p})$, $F(p)$, $\phi(\bar{p})$, $G(p)$, and$g(\bar{p})$ are locally Lipschitz and definable.

Assumption 2 is not restrictive in practice as definable functions include most common functions of interest in optimization and control. For example, all semialgebraic functions, real analytic functions (restricted to a definable domain), and any product, sum, inverse, and composition of definable functions are definable [21]. Moreover, derivatives of definable functions are definable [21, Lemma 6.1]. Hence, if F and ϕ are obtained by linearizing f (which is definable by Assumption 3) using the dynamic linearization technique outlined in Section VI-A, the definability assumption is immediately satisfied.

Lemma 5: Under Assumptions 1 and 2, $\mathcal{F}$ is locally Lipschitz definable.

Proof: The projector P_C is given by

$$P_C(z) = \operatorname{diag}(P_{\mathbb{R}_{\geq 0}}(\lambda_1), \ldots, P_{\mathbb{R}_{\geq 0}}(\lambda_{n_{\text{in}}}), \mu_1, \ldots, \mu_{n_{\text{eq}}})\tag{18}$$

where $P_{\mathbb{R}_{\geq 0}} : \mathbb{R} \to \mathbb{R}_{\geq 0}$ is the 1-D projector to the set of nonnegative real numbers

$$P_{\mathbb{R}_{\geq 0}}(z) = \max\{0, z\}.\tag{19}$$

The function $P_{\mathbb{R}_{\geq 0}}$ is locally Lipschitz and piecewise linear, therefore definable. We conclude that P_C is locally Lipschitz definable.

Next, the function $z \mapsto z - \gamma(H(\bar{p})z + h(\bar{p}))$ is linear in z, and therefore both definable and Lipschitz. Moreover, thanks to Assumption 2, we have that both H and h are locally Lipschitz definable in $\bar{p}$ since they are constructed as products or sums of locally Lipschitz definable functions (as shown in Appendix B), and these operations preserve both local Lipschitz continuity and definability [26, Corollary 2.9]. Note that $Q^{-1}(p)$ is also locally Lipschitz definable since each of its entries is the ratio of two polynomial functions (i.e. semialgebraic) of the entries of Q. We conclude that $\bar{p} \mapsto z - \gamma(H(\bar{p})z + h(\bar{p}))$ is both locally Lipschitz definable in $\bar{p}$.

We conclude that $\mathcal{F}$ is locally Lipschitz definable since it is the composition of locally Lipschitz definable functions. ■

The conservative Jacobian of the dual variable z can now be readily obtained by applying the implicit function theorem in Lemma 3 to the map $\mathcal{F}$.

Theorem 1: Under Assumptions 1 and 2, the optimizer $z(\bar{p})$ of (14) is unique and locally Lipschitz definable for any $\bar{p} \in \mathscr{Y}$. Its conservative Jacobian $\mathcal{J}_z(\bar{p})$ contains elements of the form $-U^{-1}V$, where

$$U \in J_{P_C}(I - \gamma H(\bar{p})) - I\tag{20a}$$
$$V \in -\gamma J_{P_C}(Az + B)\tag{20b}$$

Algorithm 2: Computing $\mathcal{J}_{\text{MPC}}(\bar{p})$.

Input: $\bar{p}$
1: Solve (7) and get dual optimizers $z = (\lambda, \mu)$.
2: Compute $[U\, V] \in \mathcal{J}_{\mathcal{F}}(z, \bar{p})$ using (20).
3: Compute $Z = -U^{-1}V \in \mathcal{J}_z(\bar{p})$.
4: Compute $\mathcal{J}_y(\bar{p})$ using (21).
5: **return** $\mathcal{J}_{\text{MPC}}(\bar{p})$ (extracted from $\mathcal{J}_y(\bar{p})$).

with $J_{P_C} \in \mathcal{J}_{P_C}(z - \gamma(H(\bar{p})z + h(\bar{p})))$ and $A \in \mathcal{J}_H(\bar{p})$, $B \in \mathcal{J}_h(\bar{p})$.

Proof: See Appendix C. ∎

Remark 1: The proof in Appendix C also establishes that we can always choose J_{P_C} of the following form:

$$J_{P_C} = \text{diag}(\text{sign}(\lambda_1), \ldots, \text{sign}(\lambda_{n_{\text{in}}}), 1, \ldots, 1)$$

justifying our choice of working with the dual problem (14) instead of the primal (13).

The conservative Jacobian $\mathcal{J}_y(\bar{p})$ of the primal optimizer $y(\bar{p})$ can then easily be retrieved from $\mathcal{J}_z(\bar{p})$ using (15). For simplicity, define

$$\mathcal{G}(z, \bar{p}) := -Q(p)^{-1}(F(p)^\top \mu + G(p)^\top \lambda + q(\bar{p})).$$

Corollary 1: Under Assumptions 1 and 2, the optimizer $y(\bar{p})$ of (13) is unique and locally Lipschitz definable for any $\bar{p} \in \mathscr{Y}$. Its conservative Jacobian $\mathcal{J}_y(\bar{p})$ contains elements of the following form:

$$W - Q(p)^{-1}[G(p)^\top\, F(p)^\top]Z \in \mathcal{J}_y(\bar{p}) \tag{21}$$

where $Z \in \mathcal{J}_z(\bar{p})$ and $W \in \mathcal{J}_{\mathcal{G},\bar{p}}(z, \bar{p})$.

Proof: Follows immediately from the fact that composition preserves the local Lipschitz continuity and definability, and by applying the chain rule of differentiation to (15). ∎

The algorithm below summarizes a procedure for computing the conservative Jacobians $\mathcal{J}_{\text{MPC},\bar{x}_t}$ and $\mathcal{J}_{\text{MPC},p}$.

Notice that $\mathcal{J}_{\text{MPC},\bar{x}_t}$ and $\mathcal{J}_{\text{MPC},p}$ are contained in $\mathcal{J}_y$, and we can therefore retrieve them by selecting the appropriate entries in $\mathcal{J}_y(\bar{x}_t, p)$.

The procedure outlined so far allows for the computation of conservative Jacobians of QPs. Extending this method to more general classes of problems is a promising direction for future research. One way to proceed could be to apply the implicit function theorem to the optimality conditions of the nonlinear problem, as done in [10]. In this case, however, it is unclear whether the resulting Jacobian will be conservative.

V. Closed-Loop Optimization Scheme

A. Backpropagation

Next, we develop a modular mechanism, based on backpropagation, to obtain the conservative Jacobian of the entire closed-loop trajectory $\bar{x}$ using the individual conservative Jacobians of each optimization problem.

In machine learning, the term *backpropagation* is used to describe methods that efficiently construct gradients with respect to design parameters of algorithms involving several successive steps. The idea is to compute the gradients of each step and combine them using the chain rule, eliminating redundant calculations and improving efficiency.

Algorithm 3: Backpropagation.

Input: $\bar{x}_0$, p.
Init: $\mathcal{J}_{\bar{x}_0}(p) = 0$.
1: **for** $t = 0$ to T **do**
2: Solve (7) and set $\bar{u}_t = \text{MPC}(\bar{x}_t, p)$.
3: Get next state $\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t)$.
4: Compute $\mathcal{J}_{\text{MPC}}(\bar{x}_0, p)$ using Algorithm 2.
5: Compute $\mathcal{J}_{\bar{x}_{t+1}}(p)$ using (23).
6: **end for**
7: **return** $\bar{x} = \bar{x}(p)$ and $\mathcal{J}_{\bar{x}}(p)$.

In our case, the closed-loop dynamics can be expressed as a recursive equation

$$\bar{x}_{t+1} = f(\bar{x}_t, \text{MPC}(\bar{x}_t, p)) \tag{22}$$

where every state $\bar{x}_{t+1}$ depends solely on its predecessor $\bar{x}_t$ and the design parameters p. To be able to propagate the conservative Jacobians through the dynamics of the system, we require f to be path-differentiable.

Assumption 3: The function f is locally Lipschitz definable.

Under Assumption 3, we can compute the conservative Jacobian $\mathcal{J}_{\bar{x}_{t+1}}(p)$ of the state $\bar{x}_{t+1}$ with respect to the design parameters p recursively as follows:

$$\begin{aligned}\mathcal{J}_{\bar{x}_{t+1}}(p) = {} & \mathcal{J}_{f,x}(\bar{x}_t, \bar{u}_t)\mathcal{J}_{\bar{x}_t}(p) + \mathcal{J}_{f,u}(\bar{x}_t, \bar{u}_t)\,[\mathcal{J}_{\text{MPC},\bar{x}_t}(\bar{x}_t, p) \\ & \cdot \mathcal{J}_{\bar{x}_t}(p) + \mathcal{J}_{\text{MPC},p}(\bar{x}_t, p)]\,.\end{aligned} \tag{23}$$

Note that $\mathcal{J}_{\bar{x}_{t+1}}(p)$ depends on $\mathcal{J}_{\bar{x}_t}(p)$, and since $\bar{x}_0$ is given, we have $\mathcal{J}_{\bar{x}_0}(p) = 0$. As a result, we can easily construct an algorithm that computes the conservative Jacobian of the closed-loop trajectory $\bar{x}$ for a given value of p iteratively. The algorithm, summarized in Algorithm 3, can be implemented online, as the closed-loop is being simulated and the values of $\bar{x}_t$ are being measured. Note that the simulation needs to span the entire horizon T.

Proposition 1: Under Assumptions 1–3, the closed-loop trajectory $\bar{x}$ is locally Lipschitz definable in p, with conservative Jacobian $\mathcal{J}_{\bar{x}}(p)$ as given by Algorithm 3.

Proof: The closed loop $\bar{x}$ is locally Lipschitz definable since it is given by the composition of locally Lipschitz definable functions. We now prove by induction that Algorithm 3 produces $\mathcal{J}_{\bar{x}}(p)$. First, $\mathcal{J}_{\bar{x}_0}(p) = 0$ since $\bar{x}_0$ is fixed a priori. Next, suppose $\mathcal{J}_{\bar{x}_t}(p)$ has been computed correctly by the algorithm. The correctness of $\mathcal{J}_{\bar{x}_{t+1}}(p)$ follows immediately (23) and Lemma 1. ∎

B. Optimization Algorithm

Once the conservative Jacobian is available, we can utilize it to update the parameter p with a gradient-based scheme. To guarantee convergence, it suffices to meet the conditions of Algorithm 1. We, therefore, choose the following update

Algorithm 4: Closed-Loop Optimization Scheme.

Input: p^0, $\bar{x}_0$, $\{\alpha_k\}_{k\in\mathbb{N}}$.
Init: $k = 0$.
1: **while** not converged **do**
2: Compute $\bar{x}^k = \bar{x}(p^k)$ and $\mathcal{J}_{\bar{x}}(p^k)$ with Algorithm 3.
3: Compute $J^k = J_1^k J_2^k$ with $J_1^k \in \mathcal{J}_{\mathcal{C}}(\bar{x}^k)$, $J_2^k \in \mathcal{J}_{\bar{x}}(p^k)$.
4: Update $p^{k+1} = P_{\mathscr{P}}[p^k - \alpha_k J]$.
5: $k \leftarrow k+1$.
6: **end while**
7: **return** $p^* = p^k$

scheme:

$$p^{k+1} = P_{\mathscr{P}}[p^k - \alpha_k J] \tag{24}$$

for any $J = J_1 J_2$ with

$$J_1 \in \mathcal{J}_{\mathcal{C}}(\bar{x}^k),\ \ J_2 \in \mathcal{J}_{\bar{x}}(p^k)$$

where $\bar{x}^k = \bar{x}(p^k)$, and α_k satisfies the conditions in (12). Algorithm 4 combines all the steps described so far.

As long as the map $\mathrm{MPC}(\bar{p})$ is well defined, i.e., problem (7) admits a feasible solution throughout the entirety of the execution of Algorithm 4, we have the following.

Theorem 2: Suppose Assumptions 1–3 hold, and that (7) is feasible for all $\bar{x}_t$ and p^k as setup in Algorithm 4. Suppose α_k satisfies (12) and $\sup_k \|p^k\| < \infty$. Then, p^k converges to a critical point of problem (4).

Proof: Since $\mathcal{J}_{\bar{x}}(p^k)$ is the conservative Jacobian of $\bar{x}$ with respect to the design parameter p, thanks to Proposition 1, we have from Lemma 4 that the iterates p^k are guaranteed to converge to a critical point $\bar{p}$ of the problem

$$\underset{p\in\mathscr{P}}{\text{minimize}} \quad \mathcal{C}(p). \tag{25}$$

Moreover, since the constraints $\mathcal{H}(p) \leq 0$ are automatically satisfied if $\mathrm{MPC}(\bar{p})$ is feasible throughout the entire runtime of Algorithm 4, we conclude that the critical point $\bar{p}$ of (25) is also a critical point of (10), which is equivalent to (4). This concludes the proof. ∎

The condition $\sup_k \|p^k\| < \infty$ holds trivially if $\mathscr{P}$ is compact. Otherwise, one can augment the cost function with a regularizer that ensures boundedness of the iterates, as discussed in [15, Sect. 6.1].

Remark 2: Since the horizon of the optimization problem (4) is finite, it is difficult to directly provide guarantees on the stability of the closed-loop dynamics (22). To obtain a controller that stabilizes (22), a simple solution would be to use the MPC controller (7) with parameter p^* for $t \in \mathbb{Z}_{[0,T]}$, and switch to a stabilizing state-feedback controller for $t \geq T$ (e.g., an LQR). Note that if T is chosen appropriately large, the closed-loop state $\bar{x}_T$ should be in a neighborhood of the origin, and a simple LQR controller (obtained by linearizing the dynamics at the origin if the system is nonlinear) should suffice.

VI. Extensions

A. Choosing S_t by Linearization

The accuracy of the approximate model S_t significantly impacts the control performance. To improve precision, we can allow the nominal dynamics to vary at different time-steps within the same MPC problem

$$x_{k+1|t} = A_{k|t}x_{k|t} + B_{k|t}u_{k|t} + c_{k|t}$$

and construct $S_t = \{(A_{k|t}, B_{k|t}, c_{k|t})\}_{k=0}^{N-1}$ by linearizing f along the state-input trajectory (x_{t-1}, u_{t-1})

$$A_{k|t} = \left.\frac{\partial f(x,u)}{\partial x}\right|_{\substack{x=x_{k+1|t-1}\\u=u_{k+1|t-1}}},\ \ B_{k|t} = \left.\frac{\partial f(x,u)}{\partial u}\right|_{\substack{x=x_{k+1|t-1}\\u=u_{k+1|t-1}}}$$

$$c_{k|t} = f(x_{k+1|t-1}, u_{k+1|t-1}) - A_{k|t}x_{k+1|t-1} - B_{k|t}u_{k+1|t-1}$$

with $u_{N|t-1} = u_{N-1|t-1}$. Alternatively, we can use $\bar{x}_t$ in place of $x_{1|t-1}$. If the state-input trajectory (x_t, u_t) predicted by the MPC at time t does not deviate significantly from $(\bar{x}_t, u_{1|t-1})$, then the linearized dynamics are expected to be a good approximation of the true system dynamics.

Since S_t now depends on the entire solution $y_{t-1} := (x_{t-1}, u_{t-1})$ of the MPC problem at time $t-1$, and possibly also on $\bar{x}_t$, the computation of $\mathcal{J}_{\bar{x}_{t+1}}(p)$ in Algorithm 3 needs to be modified

$$\begin{aligned}\mathcal{J}_{\bar{x}_{t+1}}(p) = {} & \mathcal{J}_{f,x}(\bar{x}_t, \bar{u}_t)\mathcal{J}_{\bar{x}_t}(p) + \mathcal{J}_{f,u}(\bar{x}_t, \bar{u}_t) \\ & \cdot [\mathcal{J}_{\mathrm{MPC},\bar{x}_t}(\bar{x}_t, y_{t-1}, p)\mathcal{J}_{\bar{x}_t}(p) \\ & + \mathcal{J}_{\mathrm{MPC},y_{t-1}}(\bar{x}_t, y_{t-1}, p)\mathcal{J}_{y_{t-1}}(p) \\ & + \mathcal{J}_{\mathrm{MPC},p}(\bar{x}_t, y_{t-1}, p)]\end{aligned} \tag{26}$$

where we used $\mathrm{MPC}(\bar{x}_t, y_{t-1}, p)$ instead of $\mathrm{MPC}(\bar{x}_t, p)$ to emphasize the dependence on y_{t-1}. The term $\mathcal{J}_{y_{t-1}}(p)$ can be constructed using a simple backpropagation rule

$$\begin{aligned}\mathcal{J}_{y_{t-1}}(p) = {} & \mathcal{J}_{\mathrm{QP},\bar{x}_{t-1}}(\bar{x}_{t-1}, y_{t-2}, p)\mathcal{J}_{x_{t-1}}(p) \\ & + \mathcal{J}_{\mathrm{QP},y_{t-2}}(\bar{x}_{t-1}, y_{t-2}, p)\mathcal{J}_{y_{t-2}}(p) \\ & + \mathcal{J}_{\mathrm{QP},p}(\bar{x}_{t-1}, y_{t-2}, p)\end{aligned} \tag{27}$$

where $y_{t-1} = \mathrm{QP}(\bar{x}_{t-1}, y_{t-2}, p)$. The modified backpropagation algorithm is given in Algorithm 5. Note that we can compute $\mathcal{J}_{\mathrm{MPC}}$ using Algorithm 2 by setting $\bar{p} := (\bar{x}_t, y_{t-1}, p)$.

Before beginning the simulation of the system, we need to choose the linearization trajectory y_{-1} for time-step $t = 0$, either heuristically, or by letting y_{-1} be part of p, thus allowing the optimization process to select the value of y_{-1} that yields the best closed-loop performance.

Remark 3: The linearization strategy of this section can be replaced with simpler strategies, such as choosing a fixed A and B throughout the entire MPC horizon or choosing $A_{k|t} \equiv A_t = \left.\frac{\partial f(x,u)}{\partial x}\right|_{\substack{x=x_t,\\u=u_{1|t-1}}}$ and similarly for B and c. This, however, may negatively impact the performance of the MPC controller, especially when the system dynamics are highly nonlinear. The choice of S_t is, therefore, a tradeoff between computational

Algorithm 5: Backpropagation With Linearization.

Input: $\bar{x}_0$, p, y_{-1}.
Init: $\mathcal{J}_{y_{-1}}(p)$, $\mathcal{J}_{\bar{x}_0}(p) = 0$.
1: **for** $t = 0$ to T **do**
2: Solve (7) and set $\bar{u}_t = \mathrm{MPC}(\bar{x}_t, y_{t-1}, p)$.
3: Get next state $\bar{x}_{t+1} = f(\bar{x}_t, \bar{u}_t)$.
4: Compute $\mathcal{J}_{\mathrm{MPC}}(\bar{x}_t, y_{t-1}, p)$ using Algorithm 2.
5: Compute $\mathcal{J}_{y_t}(p)$ using (27).
6: Compute $\mathcal{J}_{\bar{x}_{t+1}}(p)$ using (26).
7: **end for**
8: **return** $\bar{x} = \bar{x}(p)$ and $\mathcal{J}_{\bar{x}}(p)$.

complexity and control performance. In practice, we observed that the linearization technique of this section has an overall satisfactory performance (compare Section VII).

B. State-Dependent Cost and Constraints

The closed-loop performance of receding-horizon MPC schemes can be greatly improved by allowing certain elements in the MPC problem to be adapted online based on the state of the system. For example, in [27], the terminal cost and constraints are constructed online as functions of the state $\bar{x}_t$. This choice is shown to enlarge the region of attraction of the scheme.

Our backpropagation framework allows the incorporation of state-dependent elements in the MPC problem by letting $H_{x,N}$, $h_{x,N}$, and P be functions of both p and $\bar{x}_t$. Since both $\bar{x}_t$ and p are known at runtime, the MPC problem solved online is a QP in the form

$$\begin{aligned} \underset{y}{\text{minimize}} \quad & \frac{1}{2} y^\top Q(\bar{x}_t, p) y + q(\bar{x}_t, p)^\top y \\ \text{subject to} \quad & F(\bar{x}_t, p) y = \phi(\bar{x}_t, p) \\ & G(\bar{x}_t, p) y \leq g(\bar{x}_t, p) \end{aligned}$$

which differs from (13) only because Q, F, and G now depend on both $\bar{x}_t$ and p. As a result, we can perform closed-loop optimization using the same algorithmic procedure as in Algorithm 4 without any modification, exception made for the symbolic expression of Q, F, and G, which depend on $\bar{p} = (\bar{x}_t, p)$ instead of only p.

Remark 4: The same procedure can be applied to the case where H_x, H_u, h_x, h_u, Q_x, and R_u depend on $\bar{x}_t$. Moreover, one can easily incorporate cost matrices Q_x and R_u that also depend on y_{t-1}, for example, by linearizing the possibly nonlinear cost function $\mathcal{C}$ along the trajectory y_{t-1} and adding sufficient regularization to ensure the positive definiteness of both Q_x and R_u. We leave such cases for future work and emphasize that our framework is flexible to tune any component of the underlying MPC problem.

C. Nonconvex Cost

Our framework easily extends to scenarios where the quadratic cost in (4) is replaced with more sophisticated costs that can possibly involve other terms in addition to $\bar{x}$. Consider, for example, problem (4) with the cost $\mathcal{C}(\bar{x}, y, z, p)$, where $y := (y_0, \ldots, y_T)$ and $z = (z_0, \ldots, z_T)$ are the primal-dual optimizers of (13) at all time-steps. Through Algorithm 5, we can include any of the optimization variables in the cost and still manage to efficiently compute the gradient of the objective by storing the conservative Jacobians $\mathcal{J}_{y_t}(p)$ and $\mathcal{J}_{z_t}(p)$ and then applying the Leibniz rule

$$\mathcal{J}_{\mathcal{C}}(p) = \mathcal{J}_{\mathcal{C},\bar{x}}(p)\mathcal{J}_{\bar{x}}(p) + \mathcal{J}_{\mathcal{C},y}(p)\mathcal{J}_y(p) + \mathcal{J}_{\mathcal{C},z}(p)\mathcal{J}_z(p).$$

The conservative Jacobians $\mathcal{J}_y$ and $\mathcal{J}_z$ are already available as a by-product of Algorithm 2.

To ensure that Lemma 5 is still applicable, we only require $\mathcal{C}$ to be path-differentiable jointly in its arguments. Under this condition, the results of Theorem 2 still hold. Note that the class of path-differentiable functions is quite large and comprises a large selection of nonconvex functions.

D. Dealing With Infeasibility

So far, we have not considered the situation where (7) becomes infeasible. This can happen frequently in practice since the gradient-based optimization scheme modifies the behavior of the MPC map without guaranteeing recursive feasibility. There is, however, a simple procedure that can be used to recover from infeasible scenarios. The modification comprises two steps: first, we modify (7) to ensure its feasibility, then we change the cost function $\mathcal{C}$ to ensure that p minimizes constraint violations.

To ensure that (7) is always feasible, we introduce the slack variables ϵ_t and soften the state constraints [28] (the input constraints can always be satisfied)

$$\begin{aligned} \underset{x_t, u_t, \epsilon_t}{\min.} \quad & \mathcal{P}_\epsilon^{\mathrm{MPC}}(\epsilon_t) + \|x_{N|t}\|_P^2 + \sum_{k=0}^{N-1} \|x_{k|t}\|_{Q_x}^2 + \|u_{k|t}\|_{R_u}^2 \\ \text{s.t.} \quad & x_{k+1|t} = A_t x_{k|t} + B_t u_{k|t} + c_t,\ x_{0|t} = \bar{x}_t \\ & H_x x_{k|t} \leq h_x + \epsilon_{k|t},\ H_u u_{k|t} \leq h_u \\ & \epsilon_{k|t} \geq 0,\ k \in \mathbb{Z}_{[0,N-1]} \\ & H_{x,N} x_{N|t} \leq h_{x,N} + \epsilon_{k|N}. \end{aligned} \tag{28}$$

To avoid unnecessary constraint violation, we penalize nonzero values of ϵ_t with the penalty function $\mathcal{P}_\epsilon^{\mathrm{MPC}}(\epsilon) = c_1\|\epsilon\|_2^2 + c_2\|\epsilon\|_1$, with $c_1, c_2 > 0$. If c_2 is large enough, one can prove that $\mathcal{P}_\epsilon^{\mathrm{MPC}}$ is an exact penalty function.

Lemma 6 (See [28, Thm. 1]): Problem (28) has the same solution as (7) as long as (7) admits a solution and $c_2 > \|\lambda\|_\infty$, where λ are the multipliers associated to the inequality constraints affecting the state in (7).

To ensure that, if possible, p is chosen to have no constraint violations in a closed loop, we introduce a penalty function in the objective of (4)

$$\begin{aligned} \underset{p, \bar{x}, y, \epsilon}{\text{minimize}} \quad & \sum_{t=0}^{T} \|\bar{x}_t\|_{Q_x}^2 + \mathcal{P}_\epsilon(\epsilon) \\ \text{subject to} \quad & (y_t, u_{0|t}, \epsilon_t) \sim \mathrm{MPC}(\bar{x}_t, y_{t-1}, p) \\ & \bar{x}_{t+1} = f(\bar{x}_t, u_{0|t}),\ \bar{x}_0,\ y_{-1} \text{ given} \\ & t \in \mathbb{Z}_{[0,T]} \end{aligned} \tag{29}$$

where $\epsilon := (\epsilon_0, \ldots, \epsilon_{T-1})$ and $\mathcal{P}_\epsilon(\epsilon) = c_3\|\epsilon\|_1$ for some $c_3 > 0$. The introduction of $\mathcal{P}_\epsilon$ should ensure that the solution of (29) satisfies $\epsilon_t = 0$ for all $t \in \mathbb{Z}_{[0,T-1]}$. If this is the case, the optimizers x_t and u_t of each MPC problem (28) satisfy the nominal constraints (2), thus ensuring that $\bar{x}$ and $\bar{u}$ do too.

With some reformulation, we can equivalently write (29) as

$$\underset{p}{\text{minimize}} \quad \mathcal{C}(p) + \mathcal{P}_\epsilon(\epsilon(p)) \tag{30}$$

where $\mathcal{C}$ and ϵ are locally Lipschitz definable functions of p, and ϵ is the function that maps p to the value of ϵ that solves (29). Since closed-loop constraint satisfaction is equivalent to $\epsilon(p) = 0$, the goal is to obtain a solution of

$$\underset{p}{\text{minimize}}\ \mathcal{C}(p) \quad \text{s.t. } \epsilon(p) = 0. \tag{31}$$

Under certain conditions on $\mathcal{P}_\epsilon$ and on the nature of the minimizers of (31), we can prove that (30) and (31) are equivalent, in which case $\mathcal{P}_\epsilon$ is an *exact penalty function*. For this, we need the following definition.

Definition 3: Let p^* be such that $\epsilon(p^*) = 0$. Problem (31) is *calm* at p^* if there exists some $\bar{\alpha} \geq 0$ and some $\epsilon > 0$ such that for all (p, u) with $\|p - p^*\| \leq \epsilon$ and $\epsilon(p) = u$, we have

$$\mathcal{C}(p) + \bar{\alpha}\|u\| \geq \mathcal{C}(p^*).$$

Calmness is a rather weak regularity condition that is verified in many situations. In finite dimensions, it holds for a dense subset of the perturbations [29, Prop. 2.1]. For calm minimizers of (30), we have the following.

Proposition 2 (See [29, Thm. 2.1]): The set of local minima p^* of (31) for which (31) is calm at p^* coincide with the local minima of (30) provided that the penalty parameter c_3 is chosen at least as large as the calmness modulus.

Generally, it may be challenging to obtain an accurate estimate of the calmness module. Nevertheless, for practical purposes, a large enough value of c_3 typically produces the desired effect $\epsilon = 0$.

With this in mind, we can use Algorithm 4 replacing $\mathcal{C}$ with $\mathcal{C} + \mathcal{P}_\epsilon$ and the MPC problem (7) with (28). If c_2 and c_3 are sufficiently large, and under the calmness assumption of Proposition 2, we can guarantee convergence to a local minimizer of the problem with constraints (31). Combining Lemma 6 and Proposition 2 yields the following.

Theorem 3: Under Assumptions 1–3, let p^* be the optimal parameter obtained with Algorithm 3 applied to (30). If (31) is calm at p^* and c_2 and c_3 are sufficiently large, then the MPC controller given in (7) (without constraint relaxation) is recursively feasible for the dynamics (22), and the closed-loop constraints (2) are satisfied for all $t \in \mathbb{Z}_{[0,T]}$.

VII. Simulation Example

All simulations are done in CasADi [30] with the active set solver DAQP [31] on a laptop with 32 GB of RAM and an Intel Core processor i7-1165G7 @ 2.80 GHz. The code is available and open source.[1] Average offline computation times for each BP-MPC iterations (including the simulation of T time-steps and the computation of the conservative Jacobians) are 499.130 and 205.486 ms for the examples in Sections VII-A and VII-B, respectively.

[1][Online]. Available: https://github.com/RiccardoZuliani98/BP-MPC_improving_closed_loop_MPC_performance

A. Input-Constrained Example

We begin by deploying our optimization scheme to solve problem (9) for the continuous time pendulum on cart system of [32]

$$\ddot{x}(t) = \frac{m\mu g \sin(\phi) - \mu\cos(\phi)(u + \mu\dot{\phi}^2 \sin(\phi))}{mJ - \mu^2\cos(\phi)^2} \tag{32a}$$

$$\ddot{\phi} = \frac{J(u + \mu\dot{\phi}^2 \sin(\phi)) - \mu^2 g \sin(\phi)\cos(\phi)}{mJ - \mu^2\cos(\phi)^2} \tag{32b}$$

where x and $\dot{x}$ are the position and velocity of the cart, respectively, and ϕ and $\dot{\phi}$ are the angular position and velocity of the pendulum, respectively. To fulfill Assumption 3, one can restrict the domain of (32) to an arbitrarily large interval and set the functions equal to zero outside. The goal is to steer the system to the upright equilibrium position $\bar{x} = (0, 0, 0, 0)$ starting from $\bar{x}_0 = (0, 0, -\pi, 0)$ (i.e., pendulum down). For the time being, we only consider the input constraints $\bar{u}(t) \in [-4, 4]$ and postpone state constraints to Section VII-B. We discretize the nonlinear ODE (32) using Runge–Kutta 4 with a sampling time of 0.015 s. The closed-loop objective is to minimize

$$\mathcal{C}(\bar{x}, \bar{u}) = \sum_{t=0}^{170} \|\bar{x}_t\|^2_{Q_x} + 10^{-6}\|\bar{u}_t\|^2 \tag{33}$$

with $Q_x = \text{diag}(100, 1, 100, 1)$. The MPC (7) utilizes the same input constraints and state cost matrix Q_x, and an input and terminal cost are parameterized as

$$R_u = p_0^2 + 10^{-6},\ P = \tilde{P}\tilde{P}^\top + 10^{-8}I,\ \tilde{P} = \begin{bmatrix} p_1 & 0 & 0 & 0 \\ p_2 & p_3 & 0 & 0 \\ p_4 & p_5 & p_6 & 0 \\ p_7 & p_8 & p_9 & p_{10} \end{bmatrix}.$$

We set $p = (p_0, \ldots, p_{10})$, initialized with $p_0 = 0$ and P equal to the solution of the discrete-time Algebraic Riccati equation (computed on the linearized dynamics at the origin). Note that this choice of P and R ensures that $P \succ 0$, and $R \succ 0$ for all p. We use the linearization strategy of Section VI-A to obtain linear dynamics for (7). We choose a short horizon of $N = 11$. In Algorithm 4, we set

$$\alpha_k = \frac{\rho\log(k+1)}{(k+1)^\eta}$$

which fulfills the assumptions in Theorem 2 for any $\eta \in (0.5, 1]$ and $\rho > 0$. Through manual tuning, we chose $\rho = 5 \cdot 10^{-4}$ and $\eta = 0.51$.

Fig. 1 shows the evolution of the relative difference between the closed-loop cost attained by applying Algorithm 4 and the globally optimal cost for problem (4) for different numbers of iterations. Here, the global optimum is obtained by solving (4) directly using the nonlinear solver IPOPT [33]. Note that this coincides with the solution of a nonlinear MPC problem with

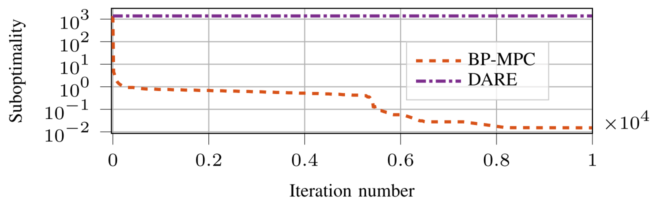

Fig. 1. Suboptimality between closed-loop cost and optimal cost.

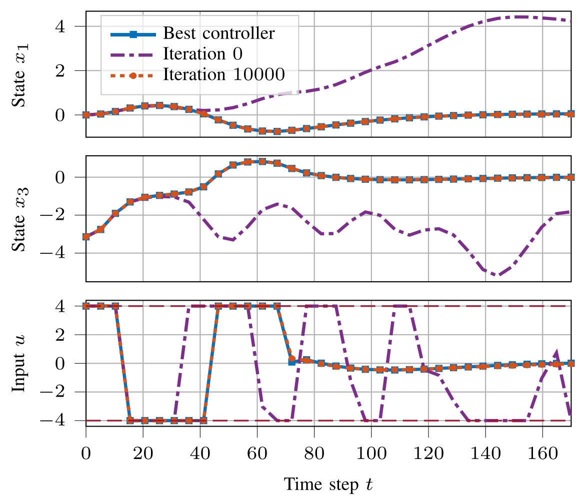

Fig. 2. Comparison of closed-loop state and input trajectories.

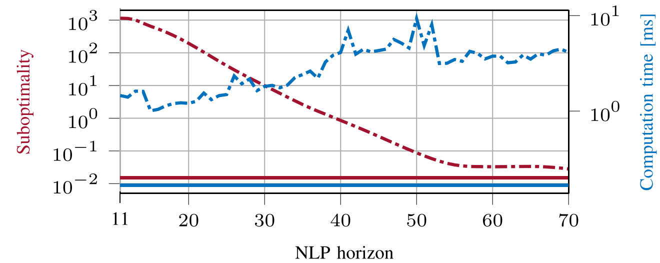

Fig. 3. Comparison of the relative suboptimality and the worst-case computation times of an NMPC with different horizon lengths (dashed lines), and our scheme with fixed horizon $N = 11$ (solid lines).

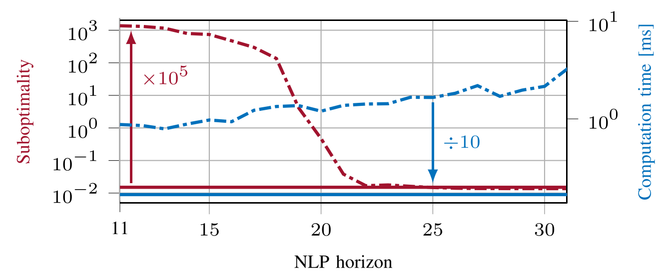

Fig. 4. Comparison of the relative suboptimality and the worst-case computation times of an NMPC with terminal cost and different horizon lengths (dashed lines), and our scheme with fixed horizon $N = 11$ (solid lines).

no terminal cost and a control horizon greater than or equal to 170 time steps, after which the system reaches the origin. The suboptimality is negligible (less than 0.01%) after only a few iterations. The figure additionally shows that our method achieves a lower cost compared to an MPC with $R_u = 10^{-6}$ and a fixed terminal cost derived from solving the discrete time algebraic Riccati equation (DARE), where A and B are chosen as the linearized dynamics at the origin.

Fig. 2 shows the closed-loop trajectories of the horizontal and angular position and of the input under different control policies. We note that the trajectory obtained after 10 000 iterations of Algorithm 4 effectively coincides with the optimal one, whereas the one obtained by using p^0 in (7) differs substantially.

For this simulation example, the linearization procedure of Section VI-A is crucial. Using a fixed linear model, obtained by linearizing the dynamics at the origin, results in an MPC controller that is not able to stabilize the system given the same horizon $N = 11$. This highlights the importance of choosing a sufficiently accurate linear prediction model for the MPC problem. In our experience, the linearization procedure outlined in Section VI-A suffices in most cases.

In Fig. 3, we compare the performance of our scheme against a nonlinear MPC controller (NMPC) with different control horizons. The NMPC is implemented using the SQP method offered by Acados [34] with the solver HPIPM. At every time-step, we warmstart the next NMPC using the solution obtained in the previous time step. At the initial time step, the warmstart trajectory is obtained by solving the NMPC problem with horizon T (similar results can be obtained with shorter horizons). The initial warmstarting, added to make the comparison with our method more fair, is crucial to ensure the fast convergence of the solver and to avoid numerical failures. As the horizon N grows, the suboptimality of the nonlinear controller decreases; however, it never reaches the performance of our controller, which utilizes a fixed horizon $N = 11$. At the same time, the worst-case computation time needed to solve the NMPC problem grows significantly.

Tuning the NMPC can significantly improve its performance. This is showcased in Fig. 4, where we added the terminal cost $x_{N|t}^\top P x_{N|t}$ (with P chosen as the solution of the algebraic Riccati equation for the linearized dynamics at the origin) to the NMPC. In this case, the NMPC with horizon $N = 25$ or greater outperforms our scheme. For $N = 25$, however, the NMPC requires about 10 times more computation time in the worst-case scenario compared to the worst-case scenario of our scheme. If the horizon of the NMPC is chosen equal to ours (i.e., $N = 11$), then our scheme attains a cost that is 10^5 smaller compared to the NMPC.

One might argue that BP-MPC is unnecessary when the system dynamics (32) are known and noise free. The optimal performance can more efficiently be obtained by solving a nonlinear trajectory optimization problem with a horizon larger than 170 and applying the optimal input $u_t^{\texttt{t\_opt}}$ open loop. This choice, however, is very fragile against process noise. Fig. 5 shows the closed-loop cost of our scheme (applied in receding horizon) and that of $u_t^{\texttt{t\_opt}}$ (applied in open loop), assuming that the dynamics (32) are affected by a stochastic additive noise sampled

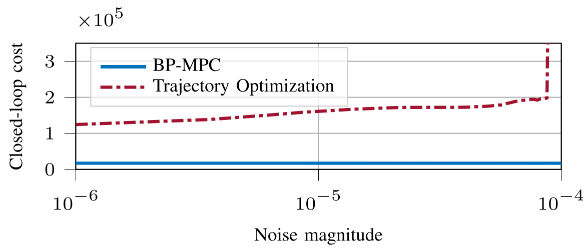

Fig. 5. Closed-loop median performance (over 1000 random noise samples) of nonlinear trajectory optimization (feedforward) and BP-MPC (receding horizon) with different noise magnitudes.

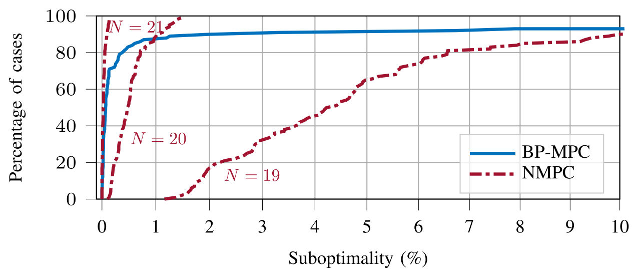

Fig. 6. Percentage suboptimality of the tuned MPC scheme (with horizon 11) and NMPC schemes with variable horizon for a set of 1000 initial conditions.

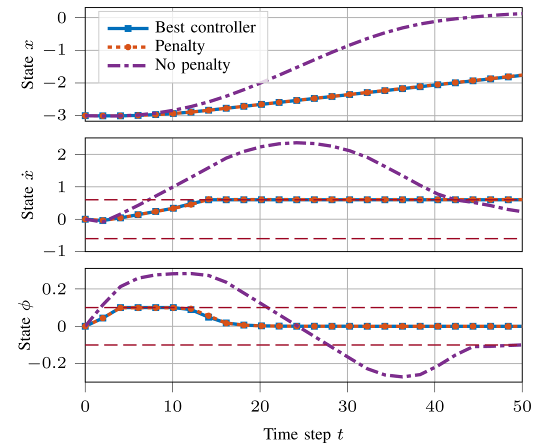

Fig. 7. Comparison of closed-loop state trajectories $x(t)$, $\dot{x}(t)$, and $\dot{\phi}(t)$ under different control policies.

TABLE I
COMPARISON OF CLOSED-LOOP COST AND CONSTRAINT VIOLATION

	Closed-loop cost	Constraint violation
DARE	377.391	0.713
BP-MPC (penalty)	390.824	0
BP-MPC (no penalty)	180.979	43.829
Best achievable	380.244	0

uniformly from the set $\{0\} \times [0, w_{\max}] \times [0, w_{\max}] \times \{0\}$, for different values of $w_{\max}$. Our scheme compensates for the noise, maintaining good performance.

Fig. 6 shows the percentage suboptimality (the ratio between the closed-loop cost attained by the controller and the best achievable cost) of our controller (blue line) averaged over 1000 different initial conditions sampled from the set $\bar{x}_0 + [\omega_1, \omega_2, 0, 0]$, with ω_1, ω_2 sampled uniformly from the interval $[-0.02, 0.02]$. We can see that the tuned MPC (with horizon 11) performs well for about 90% of all samples, but achieves a much larger cost (suboptimality $> 10\%$) in about 10% of the cases. In addition, our scheme outperforms an NMPC with horizon $N = 19$ (with terminal cost set to the solution of the DARE) and outperforms an NMPC with $N = 20$ in about 90% of the cases. Note, however, that an NMPC with horizon $N = 21$ performs strictly better than our controller. This indicates that the optimized MPC may not generalize well to all unseen initial conditions. The question of how to further robustify the tuning algorithm is a promising direction for future work.

B. Input and State-Constrained Example

In this section, we consider the same dynamics (32), discretized using RK4 with a sampling time of 0.05, but this time with initial condition $x(0) = (-3, 0, 0, 0)$ (corresponding to the pendulum being upright and the cart at -3 meters from the origin). In this case, the goal is to steer the system to the origin while satisfying the constraints $\dot{x}(t) \in [-0.6, 0.6]$, $\phi(t) \in [-0.1, 0.1]$, $\dot{\phi}(t) \in [-0.6, 0.6]$, and $u(t) \in [-0.9, 0.9]$. We choose the MPC horizon $N = 6$, with $T = 120$, and closed-loop cost $Q_x = \mathrm{diag}(1, 0.01, 1, 0.1)$, $R_u = 0.01$. Moreover, we use the same choice of p as in Section VII-A with the same initialization.

This new control task is very challenging from a safety perspective, as the controller needs to reduce the aggressiveness of the control action to avoid violating the tight constraints. We, therefore, utilize the soft-constrained MPC described in (28) with penalty parameters $c_1 = 15$ and $c_2 = 15$ and apply the optimization scheme with the objective function in (30), with $P_\epsilon(\epsilon) = 601^\top \epsilon + 40\epsilon^\top \epsilon$ (where $\epsilon := (\epsilon_1, \epsilon_2, \ldots, \epsilon_T)$ contains the slack variables of all the optimization problems, each of which spans N time-steps, and $\mathbf{1}$ is the vector of all ones). The results can be seen in Fig. 7, where the constraints are satisfied and $\epsilon = 0$. The effect of a penalty on the constraint violation induces the optimization algorithm to favor values of p that maintain small constraint violations. This does not happen if $c_3 = c_4 = 0$, as evidenced by Fig. 7 (note that in this case we still penalize the slacks within each MPC problem, i.e., $c_1 = c_2 = 15$). The closed-loop cost and constraint violations of the BP-MPC and the MPC with fixed terminal cost (equal to the solution of the DARE) are summarized in Table I.

C. Comparison With [10]

For completeness, we compare our method with [10] on the example presented in Section VII-A, using the same cost parameterization and a fully nonlinear MPC. Using open-source code,[2] we test both a quasi-Newton and a gradient-based update. However, despite extensive manual tuning of the stepsizes, both update methods failed to achieve performance comparable to our method (the best observed cost is 37 404.04, compared to 17 147.9 of our method).

The situation changes if we consider the example in Section VII-B. For a fair comparison, we use a fixed linear prediction model for both methods (obtained by linearizing the dynamics at the origin). In this case, the method in [10] converges rapidly to the optimal performance in a smaller number of iterations than our method. This is due to the fundamental difference in the two update schemes: Gros and Zanon [10] used a quasi-Newton scheme, and it is performing one update every time-step within each iteration and requires 1 iteration to converge (hence, 120 parameter updates are necessary to reach optimal performance); our scheme, on the other hand, uses a gradient-based scheme and only updates once per iteration, requiring three iterations in total.

This comparison empirically suggests that, in scenarios where the dynamics are highly nonlinear (e.g., the example of Section VII-A), our method can outperform the method in [10], achieving optimal performance despite using a simpler MPC architecture. However, in scenarios where the dynamics are linear or mildly nonlinear (e.g., the example of Section VII-B), the method in [10] can achieve similar performance with fewer iterations, thanks to the quasi-Newton update scheme and the possibility of updating the parameters at each time-step of every iteration. In addition, it is important to note that, unlike [10], our method comes with convergence guarantees.

VIII. Conclusion

We proposed a backpropagation algorithm to optimally design an MPC scheme to maximize closed-loop performance. The cost and the constraints in the MPC can depend on the current state of the system, as well as on past solutions of previous MPC problems. We employed conservative Jacobians to compute the sensitivity of the closed-loop trajectory with respect to variations of the design parameter. Leveraging a nonsmooth version of the implicit function theorem, we derived sufficient conditions under which the gradient-based optimization procedure converges to a critical point of the problem. Further, we extended our framework to cases where the MPC problem becomes infeasible, using nonsmooth penalty functions and derived conditions under which the closed-loop is guaranteed to converge to a safe solution.

Current work focuses on deploying our optimization scheme on more realistic real-life examples and extending it to scenarios where the system dynamics are only partially known and/or affected by stochastic noise.

Future work will also investigate the use of more advanced optimization techniques, such as second-order or accelerated methods, to improve the convergence rate of the optimization procedure and to obtain better convergence guarantees.

[2][Online]. Available: https://github.com/FilippoAiraldi/mpc-reinforcement-learning

Appendix

A. Obtaining (10) From (9)

Let $\bar{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ be defined as $\bar{f}(\bar{x}_t, p) = f(\bar{x}_t, \mathrm{MPC}(\bar{x}_t, p))$ and $\bar{f}^n := \bar{f} \circ \bar{f}^{n-1}$, with $\bar{f}^1 := \bar{f}$. Then, the entire closed-loop state trajectory $\{\bar{x}_t\}_{t=0}^T$ can be written for all $t \in \mathbb{Z}_{[1,T]}$ as $\bar{x}_t = \bar{f}^t(\bar{x}_0, p)$. We define, with a slight abuse of notation, $\bar{x} : \mathbb{R}^{n_p} \to \mathbb{R}^{(T+1)n_x}$ as $\bar{x}(p) := (\bar{x}_0, \bar{f}(\bar{x}_0, p), \bar{f}^2(\bar{x}_0, p), \ldots, \bar{f}^T(\bar{x}_0, p))$, where we omitted the dependence of $\bar{x}$ on $\bar{x}_0$ for simplicity. We can write the constraints in (9) compactly as $\mathcal{H}(p) \leq 0$, where $\mathcal{H} : \mathbb{R}^p \to \mathbb{R}^{(T+1)n_x}$ is defined as $\mathcal{H}(p) := \bar{H}_x \bar{x}(p) - \bar{h}_x$, with $\bar{H}_x := \mathrm{diag}(H_x, \ldots, H_x)$ and $\bar{h}_x := (h_x, \ldots, h_x)$.

B. Deriving the Dual of the QP (13)

In this section, we derive the dual problem (14) of the QP (13). To do so, consider the Lagrangian function $\mathscr{L}$ of (13), given by

$$\mathscr{L}(y, \lambda, \mu) = \frac{1}{2} y^\top Q y + q^\top y + \mu^\top (F y - \phi) + \lambda^\top (G y - g)$$

where $\lambda \geq 0$ and μ are the Lagrange multipliers of (13). The Lagrange dual function d can be obtained by minimizing $\mathscr{L}$ with respect to its first argument

$$d(\lambda, \mu) = \inf_y \frac{1}{2} y^\top Q y + q^\top y + \mu^\top (F y - \phi) + \lambda^\top (G y - g).$$

Since $\mathscr{L}$ is a convex function of x, we can find its infimum by setting $\nabla_y \mathscr{L}(y, \lambda, \mu)$ to zero and solving for y

$$\nabla_y \mathscr{L}(y, \lambda, \mu) = Q y + q + F^\top \mu + G^\top \lambda = 0. \tag{34}$$

Note that since Q is positive definite, it is invertible and its inverse is also positive definite. Therefore, the only value of y solving (34) is given by

$$y = -Q^{-1}(F^\top \mu + G^\top \lambda + q). \tag{35}$$

Substituting this value of y into $\mathscr{L}$, we get

$$d(\lambda, \mu) = d(z) = -\frac{1}{2} z^\top H z - h^\top z - c$$

where $z = (\lambda, \mu)$, $c := -1/2 q^\top Q^{-1} q$ is a constant independent of z, and

$$H := \begin{bmatrix} G Q^{-1} G^\top & G Q^{-1} F^\top \\ F Q^{-1} G^\top & F Q^{-1} F^\top \end{bmatrix}, \; h := \begin{bmatrix} G Q^{-1} q + g \\ F Q^{-1} q + \phi \end{bmatrix}. \tag{36}$$

The Lagrange dual problem aims at maximizing d over z with the constraint $\lambda \geq 0$. Equivalently, we can minimize $-d$ subject to the same constraint

$$\begin{aligned} \text{minimize}_z \quad & \frac{1}{2} z^\top H z + h^\top z + c \\ \text{subject to} \quad & E z \geq 0 \end{aligned}$$

where $E = [I \; 0]$ is the matrix satisfying $E z = \lambda$.

C. Proof of Theorem 1

By [35, Thm. 1], if LICQ holds for a convex optimization problem like (13), then the set of multipliers is a singleton. Hence, $z(p)$ is unique since it is a multiplier for (13).

We first show that the expressions in (20) represent the elements of the conservative Jacobian $\mathcal{J}_{\mathcal{F}}(z, \bar{p})$. Note that

$$\mathcal{J}_{\mathcal{F},z}(z, \bar{p}) = J_{P_C} \mathcal{J}_{[z-\gamma(H(\bar{p})z+h(\bar{p}))],z}(z, \bar{p}) - I \tag{37}$$

$$= J_{P_C}(I - \gamma H(\bar{p})) - I \tag{38}$$

where $C = \{z : Ez \geq 0\} = \mathbb{R}^{n_{\text{in}}}_{\geq 0} \times \mathbb{R}^{n_{\text{eq}}}$ and $J_{P_C} \in \mathcal{J}_{P_C}(\bar{z})$ with

$$\bar{z} = (\bar{\lambda}, \bar{\mu}) := z - \gamma(H(\bar{p})z + h(\bar{p})) \tag{39}$$

and we have used the fact that $z - \gamma(H(\bar{p})z - h(\bar{p}))$ is differentiable in z with Jacobian equal to $I - \gamma H(\bar{p})$. Next, we have

$$\begin{aligned}\mathcal{J}_{\mathcal{F},\bar{p}}(z, \bar{p}) &= J_{P_C} \mathcal{J}_{[z-\gamma(H(\bar{p})z+h(\bar{p}))],\bar{p}}(z, \bar{p}) \\ &= -\gamma J_{P_C}(Az + B)\end{aligned}$$

where $A \in \mathcal{J}_H(\bar{p})$, $B \in \mathcal{J}_h(\bar{p})$, and we have used the fact that H and h are path-differentiable as a by-product of Lemma 5. This proves that (20a) and (20b) are the elements of the conservative Jacobian $\mathcal{J}_{\mathcal{F}}(z, \bar{p})$.

The conservative Jacobian of P_C can be easily computed by recognizing from (18) that P_C is composed by a number of scalar projectors to the nonnegative orthant and a number of scalar identity functions. The identity functions have unitary Jacobian, whereas the projectors, whose explicit representation is given in (19), are convex and path differentiable (since they are Lipschitz and convex). Therefore, from [14, Corollary 2], we conclude that their subdifferential (in the sense of convex analysis) is a conservative field. This means that we can choose for each $i \in \mathbb{Z}_{[1,n_{\text{in}}]}$

$$\mathcal{J}_{P_{\mathbb{R}_{\geq 0}}}(\lambda_i) = \begin{cases} 0, & \text{if } \lambda_i < 0 \\ 1, & \text{if } \lambda_i > 0 \\ [0, 1], & \text{if } \lambda_i = 0. \end{cases}$$

We can then retrieve the conservative Jacobian of P_C by combining the conservative Jacobians of all the projectors

$$\mathcal{J}_{P_C}(\bar{z}) = \operatorname{diag}(\mathcal{J}_{P_{\mathbb{R}_{\geq 0}}}(\bar{\lambda}_1), \ldots, \mathcal{J}_{P_{\mathbb{R}_{\geq 0}}}(\bar{\lambda}_{n_{\text{in}}}), 1, \ldots, 1).$$

Assuming, for the time being, that every element of $\mathcal{J}_{\mathcal{F},z}(z(\bar{p}), \bar{p})$ is invertible, where $z(\bar{p})$ is the solution of (14), we immediately have that z is path-differentiable with conservative Jacobian $\mathcal{J}_z(\bar{p})$, and $-U^{-1}V \in \mathcal{J}_z(\bar{p})$ for every $[U\, V] \in \mathcal{J}_{\mathcal{F}}(z, \bar{p})$, thanks to Lemma 3. We now only need to prove that for every $[U\, V] \in \mathcal{J}_{\mathcal{F}}(z, \bar{p})$, U is invertible.

Let $\mathcal{I} \subset \mathbb{Z}_{[1,n_{\text{in}}]}$ denote the active inequality constraints. Remembering that $\bar{\lambda}_i$ is defined in (39), and that $\lambda_i = P_{\mathbb{R}_{\geq 0}}[\bar{\lambda}_i]$, we now prove the following.

1) $\bar{\lambda}_i > 0$ if $i \in \mathcal{I}$ with $\lambda_i > 0$ (active constraint).
2) $\bar{\lambda}_i < 0$ if $i \notin \mathcal{I}$ with $\lambda_i = 0$ (inactive constraint).
3) $\bar{\lambda}_i = 0$ if $i \in \mathcal{I}$ with $\lambda_i = 0$ (weakly active constraint).

Here, 1) is obvious, since if $\bar{\lambda}_i > 0$, then $\lambda_i = \max\{0, \bar{\lambda}_i\} = \bar{\lambda}_i$. To prove 2), suppose $\mathbf{r}(Gy, i) < g_i$ with $i \notin \mathcal{I}$. Then, since $y = -Q^{-1}(F^\top \mu + G^\top \lambda + q)$, we have

$$\mathbf{r}\left(-GQ^{-1}F^\top \mu - GQ^{-1}G^\top \lambda - GQ^{-1}q, i\right) < g_i. \tag{40}$$

From the definition of $\bar{\lambda}_i$ in (39), we have

$$\begin{aligned}\bar{\lambda}_i \ &= \ \lambda_i - \gamma \mathbf{r}(H_{11}\lambda + H_{12}\mu + h_1, i) \\ &= \lambda_i + \gamma \mathbf{r}(-GQ^{-1}G^\top \lambda - GQ^{-1}F^\top \mu - GQ^{-1}q, i) - \gamma g_i \\ &\overset{(40)}{<} \lambda_i + \gamma g_i - \gamma g_i = \lambda_i = 0\end{aligned}$$

since $i \notin \mathcal{I}$. This proves 2).

To prove 3), consider the case where $i \in \mathcal{I}$ and $\lambda_i = 0$. Then, (40) becomes

$$\mathbf{r}(-GQ^{-1}F^\top \mu - GQ^{-1}G^\top \lambda - GQ^{-1}q, i) = g_i \tag{41}$$

and similarly to before, we have

$$\begin{aligned}\bar{\lambda}_i \ &= \ \lambda_i + \gamma \mathbf{r}(-GQ^{-1}G^\top \lambda - GQ^{-1}F^\top \mu - GQ^{-1}q, i) - \gamma g_i \\ &\overset{(41)}{=} \lambda_i + \gamma g_i - \gamma g_i = \lambda_i = 0.\end{aligned}$$

We let n_{i}, n_{wa}, and n_{a} denote the number of inactive, weakly active, and strongly active inequality constraints, respectively. Without loss of generality, we can assume that $J_{P_C} = \operatorname{diag}(J_{\text{i}}, J_{\text{wa}}, J_{\text{a}}, I)$, where $J_{\text{i}} = \operatorname{diag}(0, \ldots, 0)$ is associated with inactive constraints, $J_{\text{wa}} = \operatorname{diag}(\beta_1, \ldots, \beta_{n_{\text{wa}}})$ is associated with weakly active constraints (with $\beta_i \in [0, 1]$), and $J_{\text{a}} = I$ is associated with (strongly) active constraints. Similarly, let $G = (G_{\text{i}}, G_{\text{wa}}, G_{\text{a}})$ and $v = (v_{\text{i}}, v_{\text{wa}}, v_{\text{a}}, v_{\text{eq}})$, where v_{eq} is associated with the equality constraints.

Assume now, for the sake of contradiction that there exists $v \in \mathbb{R}^{n_{\text{in}}+n_{\text{eq}}}$, $v \neq 0$, satisfying $Uv = 0$. This condition can be written using (38) as

$$J_{P_C}[I - \gamma H(\bar{p})]v - v = 0 \tag{42}$$

or equivalently as in (43) shown at the top of the next page. Since the top left part of the J_{P_C} matrix is full of zeros, the condition for v_{i} is $v_{\text{i}} = 0$, and we can, therefore, rewrite (43) as in (44), shown at the top of the next page. Next, let $\tilde{G}_{\text{wa}} := \sqrt{\gamma} G_{\text{wa}} L$, $\tilde{G}_{\text{a}} := \sqrt{\gamma} G_{\text{a}} L$, and $\tilde{F} := \sqrt{\gamma} F_{\text{wa}} L$, where $Q^{-1} = LL^\top$ is the Cholesky decomposition of Q^{-1}. With this notation (44) can be rewritten as in (45), shown at the top of the next page. We can assume that J_{wa} is invertible, as if any $\beta_i = 0$, we get the condition $\mathbf{r}(v_{\text{wa}}, i) = 0$ and we can treat the constraint the same way we did with the inactive constraints. By multiplying both matrices in (45) by $\operatorname{diag}(J_{\text{wa}}^{-1}, I, I)$, we obtain (46), shown at the top of the next page.

Note that $\mathcal{A}_1$ is negative semidefinite since J_{wa}^{-1} is a diagonal matrix with entries $1/\beta_i \in [1, +\infty)$, where $+\infty$ is not included, and $-\mathcal{A}_2$ is also negative definite since $\mathcal{A}_2 = V^\top V$, where $V^\top = (\tilde{G}_{\text{wa}}, \tilde{G}_{\text{a}}, \tilde{F})$ is full row rank because of Assumption 1 and L is invertible. Hence $\mathcal{A}_1 - \mathcal{A}_2$ is negative definite and invertible, and the only way to verify (42) is to have $v = 0$. We conclude that every U is invertible by contradiction.

$$\begin{bmatrix} 0 & & & \\ & J_{\mathrm{wa}} & & \\ & & I & \\ & & & I \end{bmatrix} \left[I - \gamma \begin{bmatrix} G_{\mathrm{i}}Q^{-1}G_{\mathrm{i}}^\top & G_{\mathrm{i}}Q^{-1}G_{\mathrm{wa}}^\top & G_{\mathrm{i}}Q^{-1}G_{\mathrm{a}} & G_{\mathrm{i}}Q^{-1}F^\top \\ G_{\mathrm{wa}}Q^{-1}G_{\mathrm{i}}^\top & G_{\mathrm{wa}}Q^{-1}G_{\mathrm{wa}}^\top & G_{\mathrm{wa}}Q^{-1}G_{\mathrm{a}} & G_{\mathrm{wa}}Q^{-1}F^\top \\ G_{\mathrm{a}}Q^{-1}G_{\mathrm{i}}^\top & G_{\mathrm{a}}Q^{-1}G_{\mathrm{wa}}^\top & G_{\mathrm{a}}Q^{-1}G_{\mathrm{a}} & G_{\mathrm{a}}Q^{-1}F^\top \\ FQ^{-1}G_{\mathrm{i}}^\top & FQ^{-1}G_{\mathrm{wa}}^\top & FQ^{-1}G_{\mathrm{a}} & FQ^{-1}F^\top \end{bmatrix} \right] \begin{bmatrix} v_{\mathrm{i}} \\ v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{i}} \\ v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} \tag{43}$$

$$\begin{bmatrix} J_{\mathrm{wa}} & & \\ & I & \\ & & I \end{bmatrix} \left[I - \gamma \begin{bmatrix} G_{\mathrm{wa}}Q^{-1}G_{\mathrm{wa}}^\top & G_{\mathrm{wa}}Q^{-1}G_{\mathrm{a}} & G_{\mathrm{wa}}Q^{-1}F^\top \\ G_{\mathrm{a}}Q^{-1}G_{\mathrm{wa}}^\top & G_{\mathrm{a}}Q^{-1}G_{\mathrm{a}} & G_{\mathrm{a}}Q^{-1}F^\top \\ FQ^{-1}G_{\mathrm{wa}}^\top & FQ^{-1}G_{\mathrm{a}} & FQ^{-1}F^\top \end{bmatrix} \right] \begin{bmatrix} v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} = \begin{bmatrix} v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} \tag{44}$$

$$\begin{bmatrix} J_{\mathrm{wa}} - I & & \\ & 0 & \\ & & 0 \end{bmatrix} \begin{bmatrix} v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} - \begin{bmatrix} J_{\mathrm{wa}} & & \\ & I & \\ & & I \end{bmatrix} \begin{bmatrix} \tilde{G}_{\mathrm{wa}}\tilde{G}_{\mathrm{wa}}^\top & \tilde{G}_{\mathrm{wa}}\tilde{G}_{\mathrm{a}} & \tilde{G}_{\mathrm{wa}}\tilde{F}^\top \\ \tilde{G}_{\mathrm{a}}\tilde{G}_{\mathrm{wa}}^\top & \tilde{G}_{\mathrm{a}}\tilde{G}_{\mathrm{a}} & \tilde{G}_{\mathrm{a}}\tilde{F}^\top \\ \tilde{F}\tilde{G}_{\mathrm{wa}}^\top & \tilde{F}\tilde{G}_{\mathrm{a}} & \tilde{F}\tilde{F}^\top \end{bmatrix} \begin{bmatrix} v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{45}$$

$$\underbrace{\begin{bmatrix} I - J_{\mathrm{wa}}^{-1} & & \\ & 0 & \\ & & 0 \end{bmatrix}}_{:=\mathcal{A}_1} \begin{bmatrix} v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} - \underbrace{\begin{bmatrix} \tilde{G}_{\mathrm{wa}}\tilde{G}_{\mathrm{wa}}^\top & \tilde{G}_{\mathrm{wa}}\tilde{G}_{\mathrm{a}} & \tilde{G}_{\mathrm{wa}}\tilde{F}^\top \\ \tilde{G}_{\mathrm{a}}\tilde{G}_{\mathrm{wa}}^\top & \tilde{G}_{\mathrm{a}}\tilde{G}_{\mathrm{a}} & \tilde{G}_{\mathrm{a}}\tilde{F}^\top \\ \tilde{F}\tilde{G}_{\mathrm{wa}}^\top & \tilde{F}\tilde{G}_{\mathrm{a}} & \tilde{F}\tilde{F}^\top \end{bmatrix}}_{:=\mathcal{A}_2} \begin{bmatrix} v_{\mathrm{wa}} \\ v_{\mathrm{a}} \\ v_{\mathrm{eq}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{46}$$

References

[1] H. Chen and F. Allgöwer, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205–1217, 1998.

[2] S. D. Cairano and A. Bemporad, "Model predictive control tuning by controller matching," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 185–190, Jan. 2010.

[3] R. Soloperto, J. Köhler, and F. Allgöwer, "Augmenting MPC schemes with active learning: Intuitive tuning and guaranteed performance," *IEEE Control Syst. Lett.*, vol. 4, no. 3, pp. 713–718, Jul. 2020.

[4] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*. PMLR, 2014, pp. 387–395.

[5] B. Amos and J. Z. Kolter, "OptNet: Differentiable optimization as a layer in neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 136–145.

[6] A. L. Dontchev, R. T. Rockafellar, and R. T. Rockafellar, *Implicit Functions and Solution Mappings: A View From Variational Analysis*. Berlin, Germany: Springer, 2009.

[7] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8299–8310.

[8] A. Oshin and E. A. Theodorou, "Differentiable robust model predictive control," 2023, *arXiv:2308.08426*.

[9] W. S. Cortez, J. Drgona, D. Vrabie, and M. Halappanavar, "Robust differentiable predictive control with safety guarantees: A predictive safety filter approach," 2023, *arXiv:2311.08496*.

[10] S. Gros and M. Zanon, "Data-driven economic NMPC using reinforcement learning," *IEEE Trans. Autom. Control*, vol. 65, no. 2, pp. 636–648, Feb. 2020.

[11] M. Zanon and S. Gros, "Safe reinforcement learning using robust MPC," *IEEE Trans. Autom. Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021.

[12] M. Zanon, V. Kungurtsev, and S. Gros, "Reinforcement learning based on real-time iteration NMPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5213–5218, 2020.

[13] E. M. Bednarczuk and K. E. Rutkowski, "On Lipschitz continuity of projections onto polyhedral moving sets," *Appl. Math. Optim.*, vol. 84, no. 2, pp. 2147–2175, 2021.

[14] J. Bolte and E. Pauwels, "Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning," *Math. Program.*, vol. 188, pp. 19–51, 2021.

[15] D. Davis, D. Drusvyatskiy, S. Kakade, and J. D. Lee, "Stochastic subgradient method converges on tame functions," *Foundations Comput. Math.*, vol. 20, no. 1, pp. 119–154, 2020.

[16] J. Bolte, T. Le, E. Pauwels, and T. Silveti-Falls, "Nonsmooth implicit differentiation for machine-learning and optimization," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2021, pp. 13537–13549.

[17] A. Agrawal, S. Barratt, S. Boyd, and B. Stellato, "Learning convex optimization control policies," in *Proc. Learn. Dyn. Control*, 2020, pp. 361–373.

[18] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and J. Z. Kolter, "Differentiable convex optimization layers," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2019, pp. 9562–9574.

[19] J. Bolte, E. Pauwels, and A. Silveti-Falls, "Differentiating nonsmooth solutions to parametric monotone inclusion problems," *SIAM J. Optim.*, vol. 34, no. 1, pp. 71–97, 2024.

[20] R. Zuliani, E. C. Balta, and J. Lygeros, "Closed-loop performance optimization of model predictive control with robustness guarantees," 2024, *arXiv:2403.04655*.

[21] M. Coste, *Introduction to O-Minimal Geometry*. Rennes, France: Institut de recherche mathématique de Rennes, 1999.

[22] J. Bolte, T. Le, E. Pauwels, and A. Silveti-Falls, "Nonsmooth implicit differentiation for machine learning and optimization," 2021, *arXiv:2106.04350*. [Online]. Available: https://arxiv.org/abs/2106.04350

[23] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[24] A. Beck, *First-Order Methods in Optimization*, Philadelphia, PA, USA: SIAM, 2017.

[25] H. H. Bauschke, P. L. Combettes, H. H. Bauschke, and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Berlin, Germany: Springer, 2017.

[26] M. Coste, *An Introduction to Semialgebraic Geometry*. Rennes, France: Institut de recherche mathématique de Rennes, 2002.

[27] A. Aboudonia, A. Eichler, and J. Lygeros, "Distributed model predictive control with asymmetric adaptive terminal sets for the regulation of large-scale systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6899–6904, 2020.

[28] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," in *Proc. Control 2000 Conf.*, 2000.

[29] J. V. Burke, "An exact penalization viewpoint of constrained optimization," *SIAM J. Control Optim.*, vol. 29, no. 4, pp. 968–998, 1991.

[30] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Computation*, vol. 11, pp. 1–36, 2019.

[31] D. Arnström, A. Bemporad, and D. Axehill, "A dual active-set solver for embedded quadratic programming using recursive LDL^T updates," *IEEE Trans. Autom. Control*, vol. 67, no. 8, pp. 4362–4369, Aug. 2022.

[32] K. Guemghar, B. Srinivasan, P. Mullhaupt, and D. Bonvin, "Predictive control of fast unstable and nonminimum-phase nonlinear systems," in *Proc. 2002 Amer. Control Conf.*, 2002, pp. 4764–4769.

[33] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: An integrating framework for enterprise-wide dynamic optimization," *Comput. Chem. Eng.*, vol. 33, no. 3, pp. 575–582, 2009.

[34] R. Verschueren et al., "Acados—A modular open-source framework for fast embedded optimal control," *Math. Program. Computation*, vol. 14, no. 1, pp. 147–183, 2022.

[35] G. Wachsmuth, "On LICQ and the uniqueness of lagrange multipliers," *Operations Res. Lett.*, vol. 41, no. 1, pp. 78–80, 2013.

Riccardo Zuliani (Graduate Student Member, IEEE) received the B.S. degree in mechatronics from the University of Padua, Padua, Italy, and the M.S. degree in robotics, systems, and control from ETH Zürich, Zürich, Switzerland, where he is currently working toward the Ph.D. degree in control engineering with the Automatic Control Laboratory (IfA).

His research interests include model predictive control, differentiable optimization, iterative learning control, and additive manufacturing.

Efe C. Balta (Member, IEEE) received the B.S. degree in manufacturing engineering from the Faculty of Mechanical Engineering, Istanbul Technical University, Istanbul, Türkiye, in 2016, and the M.S. and Ph.D. degrees in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2018 and 2021, respectively.

Between 2021 and 2023, he was a Postdoctoral Researcher with the Automatic Control Laboratory (IfA), ETH Zürich, Zürich, Switzerland. Since September 2023, he has been leading the Control and Automation research group at inspire AG. He is also a Guest Senior Scientist with IfA. His research interests include control theory, optimization, statistical learning, robotics, cyber-physical systems, and additive manufacturing.

John Lygeros (Fellow, IEEE) received the B.Eng. degree in electrical engineering and the M.Sc. degree in systems control from the Imperial College of Science Technology and Medicine, London, U.K., in 1990 and 1991, respectively, and the Ph.D. degree in electrical engineering and computer sciences department from the University of California, Berkeley, Berkeley, CA, USA, in 1996.

During 1996–2000, he held a series of research appointments with the National Automated Highway Systems Consortium, Berkeley, CA, the Laboratory for Computer Science, MIT, Cambridge, MA, USA, and the Electrical Engineering and Computer Sciences Department, University of California, Berkeley. Between 2000 and 2003, he was a University Lecturer with the Department of Engineering, University of Cambridge, Cambridge, U.K., and a Fellow of Churchill College. Between 2003 and 2006, he was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Patras, Patras, Greece. In July 2006, he joined the Automatic Control Laboratory, ETH Zürich, Zürich, Switzerland, first as an Associate Professor and then has been a Full Professor since January 2010. Since 2009, he has been the Head of the Automatic Control Laboratory. Between 2015 and 2018, he was the Head of the Department of Information Technology and Electrical Engineering. His research interests include modeling, analysis, and control of hierarchical, hybrid, and stochastic systems, with applications to biochemical networks, transportation systems, energy systems, and industrial processes.

Dr. Lygeros is a Member of the IET and the Technical Chamber of Greece. Between 2013 and 2023, he was the Vice President for Finances and a Council Member of the International Federation of Automatic Control (IFAC) as well as the Board of the IFAC Foundation. Since 2020, has been the Director of the National Centre of Competence in Research "Dependable Ubiquitous Automation" (NCCR Automation).