AC4MPC: Actor-Critic Reinforcement Learning for Guiding Model Predictive Control

Rudolf Reiter, Andrea Ghezzi, Katrin Baumgärtner, Jasper Hoffmann, Robert D. McAllister, and Moritz Diehl, *Member, IEEE*

***Abstract*—Nonlinear model predictive control (MPC) and reinforcement learning (RL) are two powerful control strategies with complementary advantages. This work shows how actor-critic RL techniques can be leveraged to improve the performance of MPC. The RL critic is used as an approximation of the optimal value function, and an actor rollout provides an initial guess for the primal variables of the MPC. A parallel control architecture is proposed where each MPC instance is solved twice for different initial guesses. Besides the actor rollout initialization, a shifted initialization from the previous solution is used. The control actions from the lowest-cost trajectory are applied to the system at each time step. We provide some theoretical justification of the proposed algorithm by establishing that the discounted closed-loop cost is upper-bounded by the discounted closed-loop cost of the original RL actor plus an error term that depends on the (sub)optimality of the RL actor and the accuracy of the critic. These results do not require globally optimal solutions and indicate that larger horizons mitigate the effect of errors in the critic approximation. The proposed algorithm is intended for applications where standard methods to construct terminal costs or constraints for MPC are impractical. The approach is demonstrated in an illustrative toy example and an autonomous driving overtaking scenario.**

***Index Terms*—Dynamic programming (DP), model predictive control (MPC), reinforcement learning (RL).**

I. Introduction

BOTH nonlinear model predictive control (MPC) and reinforcement learning (RL) are techniques to obtain optimal policies for optimal control problems (OCPs) [1], [2]. MPC minimizes the cost of a simulated environmental model online [1]. The optimization framework offers an intuitive and effective approach for designing nonlinear controllers suitable for a wide range of applications [3]. For instance, it allows direct optimization over the desired cost function or specific constraints. However, for constrained systems with fast dynamics or scarce computational resources, the computational demands associated with solving the optimization problem within the available sampling time are often prohibitive, limiting the adoption of MPC in many applications. A common way to decrease the computation time is to shorten the optimization horizon and approximate the remaining horizon by a terminal value function [1], [4], [5]. A shorter online optimization horizon reduces the number of decision variables and, therefore, the computation time. While standard methods exist to construct this terminal cost (and terminal constraint), these methods require an available, desirable, and reachable (stabilizable) steady state for the system [1], [5]. However, more general control objectives, such as the autonomous driving overtaking scenario considered in Section V-C, do not meet this requirement and cannot rely on standard methods.

Another challenge with nonlinear MPC arises if the optimization problem is solved by direct methods, which formulate it as an nonlinear program (NLP) [6]. These methods require an initial guess close to the global or a sufficiently good, local optimal solution to avoid getting stuck in a substantially suboptimal solution. Sufficiently good initial guesses reduce the number of iterations required for the optimization algorithm to converge.

RL refers to a collection of algorithms that aim at learning optimal policies for OCPs by principles of dynamic programming (DP) and Monte Carlo sampling [2], [7], [8]. The optimal policy, also referred to as *actor*, and an optimal value function, also referred to as *critic*, are approximated by parameterized functions and trained during interaction with the environment. Intrinsic to all RL algorithms is the goal of obtaining globally optimal policies and value functions via interactions with the environment. RL typically yields policies with low accuracy but close to global solutions, in contrast to MPC, which finds high-accuracy local solutions. The limitations of RL algorithms stem from the potentially high-dimensional state and action spaces, the limited number of samples, and the limited expressiveness of the neural networks (NNs).

Remarkably, the RL properties are nearly orthogonal to those of MPC [9]. We propose the algorithm actor-critic RL for guiding model predictive control (AC4MPC) that combines these complementary advantages. AC4MPC aims at obtaining globally optimal policies by using trained NNs of actor-critic RL algorithms to construct a terminal value function and a policy rollout to provide an initial guess for MPC. To obtain fast online computation times, a framework is proposed to

Received 5 December 2024; revised 3 July 2025; accepted 18 September 2025. This work was supported in part by Deutsche Forschungsgemeinschaft (DFG) through Research Unit Forschungsgruppe (FOR) 2401 under Project 424107692 and in part by the European Union (EU) through Embedded Learning and Optimization for the Next Generation of Smart Industrial Control Systems (ELO-X) under Grant 953348. Recommended by Associate Editor D. Raimondo. *(Corresponding author: Rudolf Reiter.)*

Rudolf Reiter is with the Robotics and Perception Group, Department of Informatics, University of Zurich, 8006 Zurich, Switzerland (e-mail: rudolf.reiter@imtek.uni-freiburg.de).

Andrea Ghezzi and Katrin Baumgärtner are with the Department of Microsystems Engineering (IMTEK), University of Freiburg, 79110 Freiburg, Germany (e-mail: andrea.ghezzi@imtek.unifreiburg.de; katrin.baumgaertner@imtek.uni-freiburg.de).

Jasper Hoffmann is with the Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany (e-mail: hofmaja@informatik.uni-freiburg.de).

Robert D. McAllister is with Delft Center for Systems and Control, Delft University of Technology, 2628 Delft, The Netherlands (e-mail: r.d.mcallister@tudelft.nl).

Moritz Diehl is with the Department of Microsystems Engineering and the Department of Mathematics, University of Freiburg, 79110 Freiburg, Germany (e-mail: moritz.diehl@imtek.uni-freiburg.de).

Digital Object Identifier 10.1109/TCST.2025.3620521

augment the real-time iteration (RTI) scheme [10] with a parallel optimizer and evaluate whether this parallel solution exhibits a lower cost. Evaluating the parallel solutions for their predicted performance at each iteration is nontrivial since it may be an intermediate solution of the RTI scheme. Moreover, we utilize the actor and critic networks for an auxiliary evaluation control law, terminal rollout, and terminal value function to rank the trajectories based on their predicted costs and select the lowest-cost control for each iteration.

A. Related Work

Due to their complementary advantages, MPC and RL have been previously combined in various ways. To promote sample efficiency and safety, Amos et al. [11], Gros and Zanon [12], [13], Romero et al. [14], and Reiter et al. [15] use MPC together with NNs within the RL policy. These methods do not address the challenges of MPC warm-starts and terminal cost approximations.

Similar to [16] and [17], the presented approach flips the paradigm of [18], [19], and [20], which uses MPC as an expert to warm-start the training of an actor-critic RL algorithm. In fact, we assume a well-trained actor-critic RL to warm-start the online optimization of an MPC, thereby improving the overall performance.

Allocating initial guesses by external modules such as NNs and using approximations of the terminal value function have been studied in various works. For instance, in [21], warm-starts for the active set are used, Sambharya et al. [22] use warm-starts quadratic programs (QPs), and Masti and Bemporad [23] and Marcucci and Tedrake [24] use trained NNs to warm-start mixed-integer solvers. Alboni et al. [16], Grandesso et al. [17], Mansard et al. [25], Qu et al. [26], and Chen et al. [27] use a trained NN to warm-start MPC. However, using external warm-starts in each iteration may conflict with the RTI scheme [10]. Compared to [16] and [17], this article focuses on merging the opposing paradigms of reinitializations close to "better" local minima and the RTI scheme that initializes the optimization solver with the previous trajectory.

Shen and Borrelli [28] use RL on a coarse discrete state space to provide approximate motion plans for multiple vehicles tracked thereafter by distributed MPC.

Abdufattokhov et al. [29] approximate the optimal value function for Markov decision process (MDP) related to regulation problems where a quadratic terminal value function is obtained by supervised learning. Zhong et al. [4], Beckenbach et al. [30], Beckenbach and Streif [31], and Moreno-Mora et al. [32] use approximate DP or Q-learning, respectively, to approximate the value function for MPC and Deits et al. [33] use combinatorial optimization solver evaluations to approximate the value function related to a mixed-integer problem. Nonetheless, Beckenbach et al. [30] and Beckenbach and Streif [31] require a specific structure of the cost function. Similar to the proposed approach, Karnchanachari et al. [34] learn a value function as part of an MPC policy within an actor-critic method. Karnchanachari et al. [34] state relevant practical considerations when using sequential quadratic programming (SQP) with NNs. However, Karnchanachari et al. [34] do not utilize the actor, nor do it employ parallel computations or evaluations.

For regulation [35], [36], [37] or economic MPC problems [38], a stabilizing control law can be used to construct an approximate terminal value function.

Bertsekas [39] summarizes several fundamental concepts used within this work, i.e., suboptimal control, explicit value function approximations, and rollouts as implicit approximations. AC4MPC can be seen as a specific suboptimal control algorithm to approximate the optimal policy and value function.

B. Contribution

The contributions of this work are the following:

1) derivation of a control strategy, namely, AC4MPC, that combines MPC and RL to improve the overall performance and omits local optima;
2) theoretical bounds for the closed-loop cost of the proposed AC4MPC control strategy, indicating potential performance improvements relative to the RL actor for either optimal or suboptimal (locally optimal) solutions to the AC4MPC optimization problem;
3) derivation of a real-time capable algorithm based on AC4MPC and RTI, referred to as AC4MPC-RTI;
4) evaluation of AC4MPC-RTI on a realistic autonomous driving task.

The proposed algorithm is designed for a general class of control problems, focusing on applications where steady-state targets are not available and standard methods for constructing terminal cost or constraints for MPC are not feasible.

C. Outline

The remainder of this article is structured as follows. In Section II, we repeat the pivotal concepts of MPC and actor-critic RL. In Section III, the main algorithm, AC4MPC, is introduced, and its theoretical properties are derived. The method is, furthermore, adapted to yield a real-time capable version, AC4MPC-RTI, in Section IV. In Section V, the performance on an illustrative example and a more realistic automated driving (AD) example are evaluated. We conclude and discuss this article in Section VI.

II. Preliminaries

This section introduces the problem setup and essential concepts from both RL and MPC. The index j is used for a closed-loop time step, and index k refers to time steps in a prediction. The natural numbers are $\mathbb{N} = \{0, 1, \ldots\}$, and we use the definition $\mathbb{N}_N = \{x \in \mathbb{N} | x \leq N\}$ and $\mathbb{N}_{[n,N]} = \{x \in \mathbb{N}_N | n \leq x\}$. The vector of "all ones" is $\mathbf{1}$ with suitable dimensions. Throughout this work, deterministic environments are considered.

The state $s \in \mathbb{S} \subseteq \mathbb{R}^{n_s}$ and the control $u \in \mathbb{U} \subseteq \mathbb{R}^{n_u}$ are related to the dynamic discrete-time Markovian environment with the continuous transition function

$$s_{j+1} = F\left(s_j, u_j\right), \quad F : \mathbb{S} \times \mathbb{U} \to \mathbb{S}.$$

Note that $\mathbb{S}$ is the domain/range of the state space, not a desired state constraint. The objective is formulated in terms of minimizing a continuous, nonnegative stage cost $c(s, u) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}_{\geq 0}$. With a discount factor $\gamma \in (0, 1]$, the value function of a control law or policy $\pi(s) : \mathbb{S} \to \mathbb{U}$ is

$$J_\pi(s) := \sum_{k=0}^{\infty} \gamma^k c(s_k, u_k)$$
$$s_0 = s, \quad s_{k+1} = F(s_k, u_k), \quad u_k = \pi(s_k) \tag{1}$$

and the OCP that defines the optimal cost $J^*(s)$ for a given state s can be stated by

$$J^*(s) := \min_\pi J_\pi(s).$$

The optimal policy π^* is defined such that $\pi^*(s) = \arg\min_\pi J_\pi(s)$ for all $s \in \mathbb{S}$. The optimal Q-function is directly related to the optimal value function by

$$Q^*(s, u) := c(s, u) + \gamma J^*(F(s, u)). \tag{2}$$

We include constraints into the cost function, i.e., we rewrite hard constraints via L1 penalties. Particularly, a priority over a nominal cost $c_0(s, u)$ is given to satisfying equality constraints $g(s, u) = 0$ with $g(s, u) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$ and constraints $h(s, u) \geq 0$ with $h(s, u) : \mathbb{R}^{n_s} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_h}$ within the cost formulation

$$c(s_k, u_k) = c_0(s_k, u_k) + w_g^\top |g(s_k, u_k)| + w_h^\top \min(h(s_k, u_k), 0). \tag{3}$$

For sufficiently large weights $w_g \in \mathbb{R}^{n_g}$ and $w_h \in \mathbb{R}^{n_h}$, the optimal solution is equivalent to the solution of the constrained problem [40]. We consider a general class of nonnegative stage costs $c(s, u)$ that characterize the performance of the controller. Thus, we do not assume that there exists a desirable and reachable steady state for the system. As a result, achieving and maintaining $c(s, u) = 0$ may not be possible, and $\gamma < 1$ may be required to ensure that $J^*(s)$ is finite for all $s \in \mathbb{S}$.

A. Reinforcement Learning

This work is based on actor-critic RL algorithms to approximate $\pi^*(s)$, $J^*(s)$, and/or $Q^*(s, u)$ with parameterized functions $\hat{\pi}(s)$, $\hat{J}(s)$, and/or $\hat{Q}(s, u)$. Within RL, one can distinguish between on-policy methods, such as proximal policy optimization (PPO) [41], which collect samples with the currently learned policy $\hat{\pi}$ before each update, and off-policy methods, such as soft actor-critic (SAC) [42], which use data generated from policies unrelated to the currently learned policy. In the following, we utilize both actor-critic policy types, i.e., SAC and PPO. The value function obtained by SAC is typically of the type $\hat{Q}(s, u)$, and from PPO, it is $\hat{J}(s)$.

In the tabular setting, comprising discrete states and controls without function approximation, the convergence of $\hat{\pi}(s)$, $\hat{J}(s)$, and $\hat{Q}(s, u)$ toward their optimal counterparts $\pi^*(s)$, $J^*(s)$, and $Q^*(s, u)$ can be shown for both SAC [43] with vanishing entropy term and PPO [44]. For continuous state and control spaces, approximation error bounds are often restricted to linear function approximation, excluding nonlinear functions like NNs [2]. However, in practice, the estimates often converge toward the optimal policy $\pi^*(s)$ and optimal value functions $J^*(s)$ or $Q^*(s, u)$.

B. Nonlinear Mpc

MPC approximates the infinite horizon cost function in (1) via a finite horizon $N \in \mathbb{N}$ with $N \geq 1$ and a terminal cost $V_f : \mathbb{S} \to \mathbb{R}_{\geq 0}$, stated as

$$V_N(s, \mathbf{u}) := \sum_{k=0}^{N-1} \gamma^k c(s_k, u_k) + \gamma^N V_f(s_N) \tag{4}$$

in which $\mathbf{u} = (u_0, u_1, \ldots, u_{N-1})$, $s_{k+1} = F(s_k, u_k)$, and $s_0 = s$. The terminal cost $V_f(s)$ is typically designed to approximate the value function $J_\pi(s)$ for a policy $\pi(s)$ that asymptotically stabilizes the nominal system or achieves a sufficient performance objective. Then, the MPC optimization problem is

$$V_N^0(s) := \min_{\mathbf{u} \in \mathbb{U}^N} V_N(s, \mathbf{u}). \tag{5}$$

The following assumption ensures that solutions for (5) exist.

Assumption 1 (Continuity and Compactness): The functions $F : \mathbb{S} \times \mathbb{U} \to \mathbb{S}$, $c : \mathbb{S} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$, and $V_f : \mathbb{S} \to \mathbb{R}_{\geq 0}$ are continuous. The set $\mathbb{S}$ is closed, and $\mathbb{U}$ is compact.

In Section III, we leverage the concept of terminal costs and associated theoretical results to devise the proposed method.

In order to compute the solution of (5), we adopt a direct approach, specifically direct multiple shooting [6], yielding the following problem formulation:

$$\min_{\mathbf{s}, \mathbf{u}} \sum_{k=0}^{N-1} \gamma^k c(s_k, u_k) + \gamma^N V_f(s_N)$$
$$\text{s.t.} \begin{cases} s_0 = s_j \\ s_{k+1} = F(s_k, u_k), \quad u_k \in \mathbb{U}, \ k \in \mathbb{N}_{N-1}. \end{cases} \tag{6}$$

Let $\mathbf{s} = (s_0, \ldots, s_N)$ be the vector that collects the state along the prediction horizon. We include $\mathbf{s}$ among the optimization variables and a continuity condition for the system dynamics at each step of the control horizon. Enlarging the dimension of the NLP with the variables in $\mathbf{s}$ makes (6) sparse and structured. These properties enhance numerical stability and improve convergence. A favorable numerical method for solving (6) is SQP [45]. It enables the effective warm-starting of the primal variables $\mathbf{s}$ and $\mathbf{u}$. Iteratively converging schemes, such as the RTI scheme [10], can deal with fast sampling times and constrained memory of embedded devices.

Precisely, SQP attains the solution of the given NLP by iteratively solving QPs obtained by linearizing the nonlinear constraints in (6) and computing a quadratic approximation of the, potentially nonlinear, cost function. Thus, the convergence of an SQP algorithm to a minimizer of the NLP (6) requires the solution of potentially several QPs.

One can mitigate this burden by adopting the RTI scheme, which performs only one, or in general M, SQP iterations per sampling time. Intuitively, converging toward the minimizer of (6) takes place over consecutive time steps. In every closed-loop iteration, the previous solution is shifted to provide the initial guess for the new OCP. Note that within RTI, we may apply a control action to the system that stems from an SQP iteration, which is not yet fully converged to the optimum of (6). Hence, the RTI solution may violate the nonlinear constraint of (6), e.g., the resulting trajectory may not be dynamically feasible. This observation will be critical in Section IV-B to evaluate the cost of an infeasible trajectory.

If a reachable and fixed steady-state target is available, we can typically design terminal costs or constraints for NMPC using standard methods, even if the stage cost is not positive-definite with respect to this steady-state target, i.e., an economic NMPC problem [5]. However, these standard methods and associated theoretical guarantees are not easily extended to applications without available, reachable, or desirable steady-state targets. For example, the autonomous driving experiment considered in Section V-C requires the controller to execute an overtaking maneuver. In this application, there is no desired steady state. The maneuver is instead characterized by a performance objective defined by the stage cost. Constructing terminal costs or constraints via standard methods is therefore not possible. In this article, we are primarily interested in this more general class of problems for which terminal costs/constraints cannot be constructed via standard methods.

Without terminal costs/constraints, NMPC still guarantees suitable performance bounds if a sufficiently long horizon is used, globally optimal solutions to each NMPC problem are obtained, and the system satisfies a turnpike/dissipativity assumption, which is typically difficult to verify [46] and [47]. In fact, even NMPC with quadratic, positive-definite stage costs may not satisfy this property or guarantee asymptotic stability for any finite horizon [48]. In the discounted case ($\gamma < 1$), worst-case performance bounds can be derived if the stage cost admits a finite upper bound.

In Section III, we propose a method that utilizes RL to provide both a terminal cost and an initial guess for the NMPC optimization problem, thereby offering an alternative means to ensure suitable performance for NMPC without requiring standard methods to design a terminal cost or constraint. In contrast to the results available for NMPC without a terminal cost or constraint, we establish a performance guarantee that holds for any horizon length and does not require globally optimal solutions to the NMPC optimization problem. We also show that this bound improves with increasing horizon N, demonstrating the potential for improved performance relative to the RL actor for large N (if the actor is not already optimal).

III. Actor and Critic Models for Nonlinear MPC

Notice that solving (6) in each step to the global optimum, using a perfectly estimated value function $V_f(s) \equiv J^*(s)$ and applying the first control, yields the optimal policy $\pi^*(s)$. This follows directly from the definitions (1) and (2). In general, none of the optimal functions $\pi^*(s)$, $J^*(s)$, or $Q^*(s,u)$ are available. Instead, AC4MPC uses an actor model $\hat{\pi}(s)$ and a critic model $\hat{J}(s)$ or $\hat{Q}(s,u)$ to improve the performance of MPC. The trained actor and critic NNs are obtained by methods described in Section II-A. The algorithm, which is this article's main contribution, is summarized in Algorithm 1 and permits both globally optimal and suboptimal solutions to the MPC optimization problem, with the closed-loop performance of both the globally optimal and suboptimal algorithms discussed in Sections III-B and III-C, respectively.

A. Basic Ac4Mpc Algorithm Description

In AC4MPC, the terminal cost for the standard MPC formulation is defined by either the approximate value function $\hat{J}(s)$, e.g., obtained by PPO, or Q-value function $\hat{Q}(s,\hat{\pi}(s))$, e.g., obtained by SAC. Since these estimated value functions are not exact, including an additional rollout of the actor $\hat{\pi}(s)$ can improve the estimate of the true value function for this actor policy [39]. For a rollout of $R \in \mathbb{N}$ and estimated value function $\hat{J}(\cdot)$, we define the terminal cost as

$$V_f(s) := \sum_{i=0}^{R-1} \gamma^i c(s_k, \hat{\pi}(s_k)) + \gamma^R \hat{J}(s_R) \tag{7}$$

in which $s_{k+1} = F(s_k, \hat{\pi}(s_k))$ and $s_0 = s$. For an estimated Q-value function, we simply replace $\hat{J}(s)$ by $\hat{Q}(s,\hat{\pi}(s))$. This rollout aims to better approximate the value function for the actor $\hat{\pi}(s)$. With this terminal cost, the MPC objective function becomes

$$V_{N,R}(s,\mathbf{u}) = \sum_{k=0}^{N-1} \gamma^k c(s_k,u_k) + \sum_{k=N}^{N+R-1} \gamma^k c(s_k,\hat{\pi}(s_k)) + \gamma^{N+R}\hat{J}\left(s_{N+R}\right) \tag{8}$$

in which

$$s_{k+1} = \begin{cases} F(s_k,u_k), & k \in \mathbb{N}_{[0,N-1]} \\ F(s_k,\hat{\pi}(s_k)), & k \in \mathbb{N}_{[N,N+R-1]}. \end{cases}$$

Thus, the first N inputs u_k are free variables, while the following R inputs are fixed by the actor $\hat{\pi}(\cdot)$.

In addition to the rollout, the actor $\hat{\pi}(s)$ provides an initial trajectory of states and controls for the AC4MPC optimization problem. Specifically, we define the simulated state and input trajectory from an initial state $s \in \mathbb{S}$ as $\hat{\Phi}_N(s;\hat{\pi}(\cdot)) = (\hat{s}_0, \hat{s}_1, \dots, \hat{s}_N)$ and $\hat{\Psi}_N(s;\hat{\pi}(\cdot)) = (\hat{u}_0, \hat{u}_1, \dots, \hat{u}_{N-1})$ in which

$$\hat{s}_{k+1} = F(\hat{s}_k, \hat{u}_k), \quad \hat{u}_k = \hat{\pi}(\hat{s}_k), \quad \hat{s}_0 = s. \tag{9}$$

B. Optimal Solution

First, we consider a control policy based on the (global) optimum of the following minimization problem:

$$V^0_{N,R}(s) := \min_{\mathbf{u}\in\mathbb{U}^N} V_{N,R}(s,\mathbf{u})$$
$$\mathbf{u}^0_{N,R}(s) := \arg\min_{\mathbf{u}\in\mathbb{U}^N} V_{N,R}(s,\mathbf{u}).$$

The first input of the solution to this optimization problem, then, defines the control policy

$$\kappa^0_{N,R}(s) := u^0(0;s)$$

in which $\mathbf{u}^0_{N,R}(s) = (u^0_0(s), u^0_1(s), \dots, u^0_{N-1}(s))$. With this policy, the closed-loop system is

$$s_{j+1} = F\left(s_j, u_j\right), \quad u_j = \kappa^0_{N,R}\left(s_j\right). \tag{10}$$

We are interested in the closed-loop performance of the optimal AC4MPC control policy defined by

$$\mathcal{J}_T^0(s_0) := \sum_{j=0}^{T-1} \gamma^j c\left(s_j, u_j\right) \quad \text{s.t.} \quad (10)$$

relative to the closed-loop performance of the actor $\hat{\pi}(s_0)$ defined by $J_{\hat{\pi}}(s_0)$.

The inherent inaccuracy of the critic, due to both approximation error and finite training data, is captured via the Bellman error of $\hat{J}(s)$ concerning the actor $\hat{\pi}$. To establish the desired guarantees, we require bounds on the (one-sided) Bellman error of the critic and the critic's overestimation of the value function $J_{\hat{\pi}}(s)$ for all $s \in \mathbb{S}$.

Assumption 2 (Bellman Error) The actor $\hat{\pi} : \mathbb{S} \to \mathbb{U}$ is continuous. The critic $\hat{J} : \mathbb{S} \to \mathbb{R}_{\geq 0}$ is continuous and nonnegative, and there exist $\varepsilon \geq 0$ and $\delta \geq 0$ such that for all $s \in \mathbb{S}$

$$\gamma \hat{J}(F(s, \hat{\pi}(s))) \leq \hat{J}(s) - c(s, \hat{\pi}(s)) + \delta \quad (11)$$

$$\hat{J}(s) \leq J_{\hat{\pi}}(s) + \varepsilon. \quad (12)$$

Note that Assumption 2 does not require that a reachable, zero-cost, steady-state pair $(s_s, u_s) \in \mathbb{S} \times \mathbb{U}$ exists ($s_s = F(s_s, u_s)$ and $c(s_s, u_s) = 0$), nor do we require that such a steady state is rendered asymptotically stable by the potentially suboptimal actor $\hat{\pi}(s)$. This fact is important as the RL actor may not ensure $\hat{\pi}(s_s) = u_s$ due to approximation errors. Moreover, in some applications, the optimal policy may not involve stabilizing a specific steady state (see Section V-C). Continuity of $\hat{\pi}(s)$ and $\hat{J}(s)$ ensures that $V_f(s)$, defined in (7), is continuous and therefore $V_{N,R}^0(s)$ and $\mathbf{u}_{N,R}^0(s)$ exist.

The policy evaluation step in actor-critic methods aims to minimize the Bellman error for the actor $\hat{\pi}(s)$, thereby minimizing δ and ε [2]. For continuous state–action spaces, we cannot tightly compute δ and ε, but estimates of these constants can be obtained by sampling.[1] However, we expect the resulting guarantees to be conservative, as is typical for nonlinear systems, and the specific bounds are, therefore, not expected to be useful in a quantitative sense. Instead, the subsequent theoretical results are primarily intended to provide theoretical justification for the observed performance and behavior of the proposed algorithm.

We begin with the following cost decrease inequality.

Lemma 1: If Assumptions 1 and 2 hold, then

$$\gamma V_{N,R}^0\left(s^+\right) \leq V_{N,R}^0(s) - c\left(s, \kappa_{N,R}^0(s)\right) + \gamma^{N+R}\delta \quad (13)$$

in which $s^+ = F(s, \kappa_{N,R}^0(s))$ for all $s \in \mathbb{S}$.

Proof: For any $s \in \mathbb{S}$, define the optimal open-loop input trajectory $(u_0^0, u_1^0, \ldots, u_{N-1}^0) = \mathbf{u}_{N,R}^0(s)$ and the open-loop state trajectory as

$$s_{k+1}^0 = \begin{cases} F\left(s_k^0, u_k^0\right), & k \in \mathbb{N}_{[0,N-1]} \\ F\left(s_k^0, \hat{\pi}\left(s_k^0\right)\right), & k \in \mathbb{N}_{[N,N+R-1]}. \end{cases}$$

For $s^+ = F(s, \kappa_{N,R}^0(s))$, we define the candidate trajectory

$$\tilde{\mathbf{u}}^+ := \left(u_1^0(s), u_2^0(s), \ldots, u_{N-1}^0(s), \hat{\pi}\left(s_N^0\right)\right).$$

[1]For each sample $s^i \in \mathbb{S}$, we can evaluate $\delta_i := \gamma\hat{J}(F(s, \hat{\pi}(s^i))) - \hat{J}(s^i) + c(s^i, \hat{\pi}(s^i))$ and simulate the actor $\hat{\pi}(s)$, for sufficiently long time, to estimate $J_{\hat{\pi}}(s^i)$ and the error $\varepsilon_i := \hat{J}(s^i) - J_{\hat{\pi}}(s^i)$. This yields $\delta = \max_i \delta_i$ and $\varepsilon = \max_i \varepsilon_i$.

We, therefore, have from (8) that

$$\begin{aligned} &\gamma V_{N,R}\left(s^+, \tilde{\mathbf{u}}^+\right) \\ &\quad = V_{N,R}^0(s) - c\left(s, \kappa_{N,R}^0(s)\right) \\ &\quad\quad + \gamma^{N+R}\left(\gamma\hat{J}\left(s_{N+R+1}\right) + c\left(s_{N+R}^0, \hat{\pi}\left(s_{N+R}^0\right)\right) - \hat{J}\left(s_{N+R}^0\right)\right) \end{aligned}$$

in which $s_{N+R+1} = F(s_{N+R}^0, \hat{\pi}(s_{N+R}^0))$. By rearranging (11), we have that

$$\gamma\hat{J}\left(s_{N+R+1}\right) + c\left(s_{N+R}^0, \hat{\pi}\left(s_{N+R}^0\right)\right) - \hat{J}\left(s_{N+R}^0\right) \leq \delta$$

and therefore

$$\gamma V_{N,R}\left(s^+, \tilde{\mathbf{u}}^+\right) \leq V_{N,R}^0(s) - c\left(s, \kappa_{N,R}^0(s)\right) + \gamma^{N+R}\delta.$$

By optimality, $V_{N,R}^0(s^+) \leq V_{N,R}(s^+, \tilde{\mathbf{u}}^+)$ and we, therefore, have (13).□

If $\gamma < 1$, (13) indicates that large horizons N and rollouts R in AC4MPC reduce the effect of the Bellman error in the critic $\hat{J}(s)$. At the other extreme, AC4MPC with $N = 1$ and $R = 0$ is equivalent to one value iteration of the critic $\hat{J}(s)$. These observations are consistent with results for ℓ-step lookahead algorithms in DP (see [49]). The key contribution of Lemma 1 is that this result also applies to the *closed-loop* performance.

For a general class of nonlinear systems and stage costs, (13) is tight because the candidate trajectory may be the solution to the optimization problem, i.e., $\mathbf{u}^0(s^+) = \tilde{\mathbf{u}}^+$. If the critic provides an overestimate of stagewise cost decrease for all $s \in \mathbb{S}$, i.e., $\delta = 0$ in (11), then (13) is equivalent to the ideal cost decrease guarantee for MPC. Moreover, if $c(s, u)$ is a (continuous) positive-definite function with respect to a reachable steady-state target, and $\gamma = 1$, then (20) is equivalent to the cost decrease condition required for $V_N(\cdot)$ to be a (practical) Lyapunov function for the closed-loop system. The stability of the origin, then, follows from standard assumptions about the continuity of $V_N^0(\cdot)$ at the origin [1, s. 2.4.2]. In general, however, $\delta > 0$ in Assumption 2 does not guarantee asymptotic stability of the origin.

While the undiscounted problem is important in the context of steady-state tracking control, we focus on a more general setting. Thus, we permit unreachable setpoints and general performance objectives without specifying, or assuming the existence of a reachable steady-state target. In this setting, $\gamma < 1$ is typically required to ensure that $J^*(s)$ and $J_{\hat{\pi}}(s)$ are finite for all $s \in \mathbb{S}$, as is typical in approximate DP and RL. Given this more general setting, the subsequent guarantees are necessarily more conservative than a typical steady-state tracking result. Closed-loop performance bounds are provided, but asymptotic stability of a steady state is not necessarily the desired or achieved outcome.

For $\gamma \in (0, 1)$, we establish the following bound for the infinite horizon closed-loop performance.

Theorem 1: If Assumptions 1 and 2 hold and $\gamma \in (0, 1)$, then

$$\limsup_{T\to\infty} \mathcal{J}_T^0(s_0) \leq J_{\hat{\pi}}(s_0) - \sigma_{N,R}(s_0) + \gamma^{N+R}\varepsilon + \frac{\gamma^{N+R}\delta}{1-\gamma} \quad (14)$$

for all $s_0 \in \mathbb{S}$ in which

$$\sigma_{N,R}(s_0) := V_{N,R}(s_0, \hat{\mathbf{u}}) - V_{N,R}^0(s_0), \quad \hat{\mathbf{u}} = \hat{\Psi}_N(s_0; \hat{\pi}(\cdot)).$$

Proof: For any $s_0 \in \mathbb{S}$, let $\hat{u}_k = \hat{\pi}(\hat{s}_k)$ and $\hat{x}_{k+1} = F(\hat{s}_k, \hat{u}_k)$ for $k \in \mathbb{N}_{[0,N+R-1]}$. Let $\hat{\mathbf{u}} = (\hat{u}_0, \ldots, \hat{u}_{N+1})$ and note that $\hat{\mathbf{u}} = \hat{\Psi}_N(s_0; \hat{\pi}(\cdot))$. We have from Assumption 2 that

$$\begin{aligned} V_{N,R}(s_0, \hat{\mathbf{u}}) &= \sum_{k=0}^{N+R-1} \gamma^k \ell(\hat{s}_k, \hat{u}_k) + \gamma^{N+R} \hat{J}(\hat{s}_{N+R}) \\ &\le \sum_{k=0}^{N+R-1} \gamma^k \ell(\hat{s}_k, \hat{u}_k) + \gamma^{N+R} J_{\hat{\pi}}(\hat{s}_{N+R}) + \gamma^{N+R}\varepsilon \\ &\le J_{\hat{\pi}}(s_0) + \gamma^{N+R}\varepsilon. \end{aligned} \tag{15}$$

Using this inequality with the definition of $\sigma_{N,R}(s_0)$ gives

$$V^0_{N,R}(s_0) \le J_{\hat{\pi}}(s_0) + \gamma^{N+R}\varepsilon - \sigma_{N,R}(s_0). \tag{16}$$

For the closed-loop system starting from $s_0 \in \mathbb{S}$, we have from (13) that

$$c\left(s_j, \kappa^0_N(s_j)\right) \le V^0_{N,R}(s_j) - \gamma V^0_{N,R}(s_{j+1}) + \gamma^{N+R}\delta$$

for all $j \ge 0$. By repeated application of this bound, we have

$$\mathcal{J}^0_T(s_0) \le V^0_{N,R}(s_0) - \gamma^T V^0_{N,R}(s_T) + \gamma^{N+R} \sum_{j=0}^{T-1} \gamma^j \delta.$$

Note that $V^0_{N,R}(s_T)$ is nonnegative because $c(\cdot)$ and $\hat{J}(\cdot)$ are nonnegative. In the limit as $T \to \infty$, we have

$$\limsup_{T\to\infty} \mathcal{J}^0_T(s_0) \le V^0_{N,R}(s_0) + \frac{\gamma^{N+R}\delta}{1-\gamma}.$$

By applying (16) to upper bound $V^0_{N,R}(s_0)$, we have the desired bound in (14) for all $s_0 \in \mathbb{S}$.□

The bound in Theorem 1 demonstrates the tradeoffs associated with using the proposed algorithm relative to the RL actor. The function $\sigma_{N,R}(s_0)$ is defined by the cost improvement of the optimal MPC solution relative to the actor rollout over the same horizon N, with a terminal cost defined by the horizon R. Note that $\sigma_{N,R}(s_0)$ is nonnegative by the definition of optimality. This value is easily computed during the implementation of ACMPC, and larger values indicate a greater potential for improvement in the closed-loop performance of the proposed method relative to the RL actor. However, this bound is degraded by possible inaccuracies in the critic defined by the constants δ and ε in Assumption 2. These inaccuracies are mitigated by increasing N or R.

Remark 1 (Comparison With RL) If the RL actor is nearly optimal for all states $s_0 \in \mathbb{S}$, quantified by the value of $\sigma_{N,R}(s_0)$, then the proposed framework can only provide minor improvements while also introducing the possibility of underperforming the RL actor due to inaccuracies in the critic. We do not recommend the proposed method over the RL actor in this case. Alternatively, if the RL actor is suboptimal, the proposed framework offers the potential to significantly improve the RL actor's performance, despite possible inaccuracies in the critic.

Remark 2 (Comparison With NMPC): The proposed approach offers an alternative method for constructing a suitable terminal cost in applications where standard methods are not feasible. While NMPC offers some performance guarantees without a terminal cost, these methods require a sufficiently long horizon (that increases computational demand) and dissipativity/turnpike properties that can be difficult to verify a priori [46], [47], [50]. By contrast, Theorem 1 provides a performance guarantee for *all* values of $N \ge 1$. We note that if $c(s,u)$ is bounded, NMPC without a terminal cost ($\hat{J}(s) = 0$ and $R = 0$) also obtains the guarantee in Theorem 1, but with the worst possible value of $\delta := \max_{s\in\mathbb{S},u\in\mathbb{U}} c(s,u)$. For a sufficiently poor actor or critic, using NMPC without a terminal cost is likely preferable to the proposed method. Alternatively, if an actor with good performance and an accurate critic are available, then the proposed method can significantly outperform NMPC without a terminal cost, as demonstrated in subsequent experiments.

Remark 3 (N Versus R): Note that the potential improvement defined by the function $\sigma_{N,R}(s_0)$ is primarily controlled by the MPC horizon N, as additional decision variables u in the optimization problem provide additional flexibility in improving the optimal cost. Thus, the practical benefits of increasing N or R are based on the (sub)optimality of the actor [characterized by $\sigma_{N,R}(s_0)$] relative to the accuracy of the critic (characterized by ε and δ). For suboptimal actors, larger values of N are expected to increase the value of $\sigma_{N,R}(s_0)$ and therefore improve the bound in (14). For inaccurate critics, increasing R is preferable as it mitigates this inaccuracy without increasing the number of decision variables in the optimization problem.

If the actor is suboptimal within the NMPC horizon ($\sigma_{N,R}(s_0) > 0$), then we can demonstrate guaranteed improvement with respect to this actor for sufficiently large $N + R$.

Corollary 1: If Assumptions 1 and 2 hold and $\gamma \in (0,1)$, then for any $s_0 \in \mathbb{S}$ such that $\sigma_{N,R}(s_0) > 0$, there exists sufficiently large $N \ge 1$ and $R \ge 0$, such that

$$\limsup_{T\to\infty} \mathcal{J}^0_T(s_0) < J_{\hat{\pi}}(s_0).$$

Corollary 1 provides a justification for the observed benefits of the proposed method. If performance worse than the RL actor is observed, Corollary 1 indicates that increasing the horizon length can fix this issue. In the subsequent experiments, we do not explicitly verify that N and R are sufficient to guarantee the result in Corollary 1, but nonetheless empirically observe this result with modest horizons.

C. Suboptimal Solutions

In practice, however, globally optimal solutions to the NMPC or AC4MPC minimization problem may not be tractable due to computational constraints and local minima. Thus, *suboptimal* (locally optimal) solutions are often used instead. We, therefore, consider a formulation of the AC4MPC algorithm and control policy that does not rely on globally optimal solutions to the minimization problem. We describe the closed-loop iterations of AC4MPC below and in Algorithm 1. Note that for NMPC without terminal costs/constraints, no guarantees are available for the suboptimal solutions permitted in Algorithm 1 or encountered in practice.

At the first time step, we roll out the actor $\hat{\pi}$ from the current state measurement s and use this input trajectory as an initial guess for the optimization problem. The AC4MPC algorithm

Algorithm 1 AC4MPC—Closed-Loop Iterations

```
input : Policy $\hat{\pi}(\cdot)$, value function $Q(\cdot)$ or $J(\cdot)$
1  MPC ← formulation (6) with $V_f(\cdot)$ defined in (7)
     with $\hat{J}(\cdot)$ or $\hat{J}(\cdot) = \hat{Q}(\cdot, \hat{\pi}(\cdot))$;
2  for $j \leftarrow 0$ to $\infty$ do
3      $s$ ← state measurement;
4      policy rollout $\hat{\mathbf{u}} \leftarrow \hat{\Psi}_N(s; \hat{\pi}(\cdot))$;
5      if $j == 0$ then
6          $\tilde{\mathbf{u}} \leftarrow \hat{\mathbf{u}}$;
7      initialize MPC ← $\tilde{\mathbf{u}}$;
8      $\mathbf{u}$ ← solve MPC;
9      if $V_N(s, \hat{\mathbf{u}}) \leq V_N(s, \mathbf{u})$ then
10         $\mathbf{u} \leftarrow \hat{\mathbf{u}}$;
11     apply $u \leftarrow \mathbf{u}[0]$ to the system;
12     shifting $\tilde{\mathbf{u}} \leftarrow \zeta(s, \mathbf{u}; \hat{\pi}(\cdot))$;
13
```

ensures that the computed input trajectory is better (no worse) than this initial guess, i.e., produces a lower value of $V_N(\cdot)$. After the first time step, the subsequent initial guess is obtained by shifting the most recent iterate and using the actor $\hat{\pi}(s)$ to provide an initial guess only for the very last control. Let $\mathbf{u} = (u_0, u_1, \ldots, u_{N-1})$ denote the input trajectory computed by AC4MPC for the current state $s = s_0$. Then, we define a shifted input trajectory at the subsequent time step as

$$\tilde{\mathbf{u}}^+ = \zeta(s, \mathbf{u}; \hat{\pi}(\cdot)) := (u_1, \ldots, u_{N-1}, \hat{\pi}(s_N)). \tag{17}$$

The AC4MPC algorithm, then, selects a better policy (no worse) than $\tilde{\mathbf{u}}$, i.e., produces a lower value of $V_N(\cdot)$. Thus, the AC4MPC algorithm (Algorithm 1) implicitly defines a function $K_N : \mathbb{S} \times \mathbb{U}^N \to \mathbb{U}^N$, which satisfies

$$K_N(s, \tilde{\mathbf{u}}) \in \left\{\mathbf{u} \in \mathbb{U}^N \mid V_N(s, \mathbf{u}) \leq V_N(s, \tilde{\mathbf{u}})\right\}. \tag{18}$$

Note that the global optimum of the AC4MPC optimization problem satisfies the requirements of $K_N(s, \tilde{\mathbf{u}})$. The control policy $\kappa_N : \mathbb{S} \times \mathbb{U}^N \to \mathbb{U}$ defined by AC4MPC is the first input in the trajectory defined by $K_N(s, \tilde{\mathbf{u}})$, i.e.,

$$\kappa_N(s, \tilde{\mathbf{u}}) := u_0 \quad \text{with} \quad (u_0, u_1, \ldots, u_{N-1}) = K_N(s, \mathbf{u}).$$

With this control policy, we obtain the closed-loop system-optimizer dynamics

$$\begin{aligned} s_{j+1} &= F(s_j, u_j), \quad u_j = \kappa_N(s_j, \tilde{\mathbf{u}}_j) \\ \tilde{\mathbf{u}}_{j+1} &= \zeta(s_j, K_N(s_j, \tilde{\mathbf{u}}_j); \hat{\pi}(\cdot)). \end{aligned} \tag{19}$$

Note that both the state s_j and the initial guess $\tilde{\mathbf{u}}_j$ evolve according to autonomous dynamics defined by AC4MPC. Algorithm 1 provides a simple example of an AC4MPC algorithm that satisfies the requirements in (18). Different conceptual parts are highlighted in color and aligned with the associated parts in Section IV, i.e., a policy rollout (red), the initialization of the MPC (yellow), obtaining the solution of the MPC (blue), evaluating different trajectories (green), and shifting and simulating the last control (purple). Note that the solution to the MPC problem in Algorithm 1 (i.e., *solve* MPC) does not need to be a global optimum.

Lemma 2 (Cost Decrease): If Assumptions 1 and 2 hold, then

$$\gamma V_N\left(s^+, K_N\left(s^+, \tilde{\mathbf{u}}^+\right)\right) - V_N(s, K_N(s, \tilde{\mathbf{u}})) \leq -c(s, \kappa_N(s, \tilde{\mathbf{u}})) + \gamma^{N+R}\delta \tag{20}$$

in which $s^+ = F(s, \kappa_N(s, \tilde{\mathbf{u}}))$ and $\tilde{\mathbf{u}}^+ = \zeta(s, K_N(s, \tilde{\mathbf{u}}); \hat{\pi}(\cdot))$ for all $s \in \mathbb{S}$ and $\tilde{\mathbf{u}} \in \mathbb{U}^N$.

Proof: We proceed similar to the proof of Lemma 1. For any $s \in \mathbb{S}$ and $\tilde{\mathbf{u}} \in \mathbb{U}^N$, let $s^+ = F(s, \kappa_N(s, \tilde{\mathbf{u}}))$ and $\tilde{\mathbf{u}}^+ = \zeta(s, K_N(s, \tilde{\mathbf{u}}); \hat{\pi}(\cdot))$. Let s_k denote the open-loop state at time $k \in \mathbb{N}_N$ for given $s_0 = s$ and the input trajectory $\mathbf{u} = (u_0, u_1, \ldots, u_{N-1}) = K_N(s, \tilde{\mathbf{u}})$. Let $s_{k+1} = F(s_k, \hat{\pi}(s_k))$ for all $k \in \{N, \ldots, N+R-1\}$. From the definition of $\tilde{\mathbf{u}}^+$ and (8) that, we have

$$\begin{aligned} &\gamma V_N\left(s^+, \tilde{\mathbf{u}}^+\right) - V_N(s, K_N(s, \tilde{\mathbf{u}})) \\ &\quad = -c(s, \kappa_N(s, \tilde{\mathbf{u}})) \\ &\quad\quad + \gamma^{N+R}\left(\gamma\hat{J}\left(s_{N+R+1}\right) + c\left(s_N, \hat{\pi}\left(s_{N+R}\right)\right) - \hat{J}\left(s_{N+R}\right)\right). \end{aligned}$$

By rearranging (11), we have that

$$\gamma\hat{J}\left(s_{N+R+1}\right) + c\left(s_{N+R}, \hat{\pi}\left(s_{N+R}\right)\right) - \hat{J}\left(s_{N+R}\right) \leq \delta$$

and therefore

$$\gamma V_N\left(s^+, \tilde{\mathbf{u}}^+\right) - V_N(s, K_N(s, \tilde{\mathbf{u}})) \leq -c(s, \kappa_N(s, \tilde{\mathbf{u}})) + \gamma^{N+R}\delta.$$

From the definition of $K_N(\cdot)$, we have

$$V_N\left(s^+, K_N\left(s^+, \tilde{\mathbf{u}}^+\right)\right) \leq V_N\left(s^+, \tilde{\mathbf{u}}^+\right)$$

and combining these equations gives (20).□

For the suboptimal AC4MPC control policy, we see that Lemma 2 provides effectively the same result as Lemma 1. We again consider the closed-loop performance of this policy

$$\mathcal{J}_T(s_0) := \sum_{j=0}^{T-1} \gamma^j c\left(s_j, u_j\right) \quad \text{s.t.} \quad (19).$$

For this suboptimal solution, computed via Algorithm 1 and satisfying (18), we establish a closed-loop performance bound in the subsequent theorem (Theorem 2). Note that the bound in Theorem 2 is nearly identical to that of Theorem 1, with the exception that the newly defined function $\tilde{\sigma}_{N,R}(s_0)$ depends on the (possibly) *suboptimal* solution $K_N(s_0, \hat{\mathbf{u}})$ from Algorithm 1.

Theorem 2 (Performance): If Assumptions 1 and 2 hold and $\gamma \in (0, 1)$, then

$$\limsup_{T\to\infty} \mathcal{J}_T(s_0) \leq J_{\hat{\pi}}(s_0) - \tilde{\sigma}_{N,R}(s_0) + \gamma^{N+R}\varepsilon + \frac{\gamma^{N+R}\delta}{1-\gamma} \tag{21}$$

for all $s_0 \in \mathbb{S}$ in which $\hat{\mathbf{u}} = \hat{\Psi}_N(s_0; \hat{\pi}(\cdot))$ and

$$\tilde{\sigma}_{N,R}(s_0) := V_{N,R}(s_0, \hat{\mathbf{u}}) - V_{N,R}(s_0, K_N(s_0, \hat{\mathbf{u}})). \tag{22}$$

Proof: For any $s_0 \in \mathbb{S}$, let $\hat{u}_k = \hat{\pi}(\hat{s}_k)$ and $\hat{x}_{k+1} = F(\hat{s}_k, \hat{u}_k)$ for $k \in \{0, 1, \ldots, N+R-1\}$. Let $\hat{\mathbf{u}} = (\hat{u}_0, \ldots, \hat{u}_{N+1})$ and note that $\hat{\mathbf{u}} = \hat{\Psi}_N(s_0; \hat{\pi}(\cdot))$. We have from Assumption 2 that

$$V_{N,R}(s_0, \hat{\mathbf{u}}) = \sum_{k=0}^{N+R-1} \gamma^k \ell(\hat{s}_k, \hat{u}_k) + \gamma^{N+R}\hat{J}\left(\hat{s}_{N+R}\right)$$

$$\leq \sum_{k=0}^{N+R-1} \gamma^k \ell(\hat{s}_k, \hat{u}_k) + \gamma^{N+R} J_{\hat{\pi}}\left(\hat{s}_{N+R}\right) + \gamma^{N+R}\varepsilon$$
$$\leq J_{\hat{\pi}}(s_0) + \gamma^{N+R}\varepsilon. \quad (23)$$

Using this inequality with the definition of $\tilde{\sigma}_{N,R}(s_0)$ gives

$$V_{N,R}(s_0, K_N(s_0, \hat{\mathbf{u}})) \leq J_{\hat{\pi}}(s_0) + \gamma^{N+R}\varepsilon - \tilde{\sigma}_{N,R}(s_0). \quad (24)$$

From Lemma 2 and the same arguments as in the proof of Theorem 1, the closed-loop system starting from $s_0 \in \mathbb{S}$ and $\tilde{\mathbf{u}}_0 = \hat{\Psi}_N(s_0; \hat{\pi}(\cdot))$ satisfies

$$\mathcal{J}_T(s_0) \leq V_{N,R}(s_0, K_N(s_0, \tilde{\mathbf{u}}_0))$$
$$- \gamma^T V_{N,R}(s_T, K_N(s_T, \tilde{\mathbf{u}}_T)) + \gamma^{N+R} \sum_{j=0}^{T-1} \gamma^j \delta.$$

Note that $V_{N,R}(\cdot)$ is nonnengative because $c(\cdot)$ and $\hat{J}(\cdot)$ are nonnegative. In the limit as $T \to \infty$, we have

$$\mathcal{J}_T(s_0) \leq V_N(s_0, K_N(s_0, \tilde{\mathbf{u}}_0)) + \frac{\gamma^{N+R}\delta}{1-\gamma}.$$

By applying (24), we have (21).□

These observations are again consistent with results for ℓ-step lookahead algorithms in DP (see [49]). We note, however, that DP typically assumes that a globally optimal solution is obtained for the ℓ-step lookahead minimization. Since we are permitting suboptimal solutions in the AC4MPC algorithm, the best guarantee we obtain is that the closed-loop performance is bounded relative to the closed-loop performance of the actor used in the AC4MPC algorithm. In economic MPC, similar results are obtained with respect to a periodic reference trajectory that is used to construct the terminal cost and constraint [5], [51].

Theorem 2 provides a performance guarantee for any horizon length (N and R) and does not require globally optimal solutions to the proposed optimization problem, a further improvement relative to Theorem 1 and the discussion in Remark 2. In contrast to the optimal solution and Theorem 1, this suboptimal algorithm and Theorem 2 do not cover typical implementations of NMPC without a terminal cost, even if $c(s, u)$ is bounded, because there is no actor available to generate the warm-start. A large cost improvement relative to the RL actor, as defined by $\tilde{\sigma}_{N,R}(s_0)$, is still important to justify the use of AC4MPC instead of simply implementing the RL actor, but global optimality is not required to achieve this improvement. Remark 1 still holds and Remark 3 also applies to $\tilde{\sigma}_{N,R}(s_0)$.

Moreover, we emphasize that the bound in (21) is conservative. In practice, we expect AC4MPC with moderate horizon lengths N and rollout lengths R to outperform the RL actor, as demonstrated in the subsequent experiments. Stronger guarantees may be possible if we strengthen the assumptions on the stage cost and system, e.g., strict dissipativity or turnpike properties [52].

IV. Multiple Shooting and RTIs for AC4Mpc

So far, AC4MPC was defined conceptually as a single shooting formulation without a practical algorithm to solve the MPC problem (5). In the following, we propose AC4MPC-RTI, a practical extension to AC4MPC that significantly reduces online computation time.

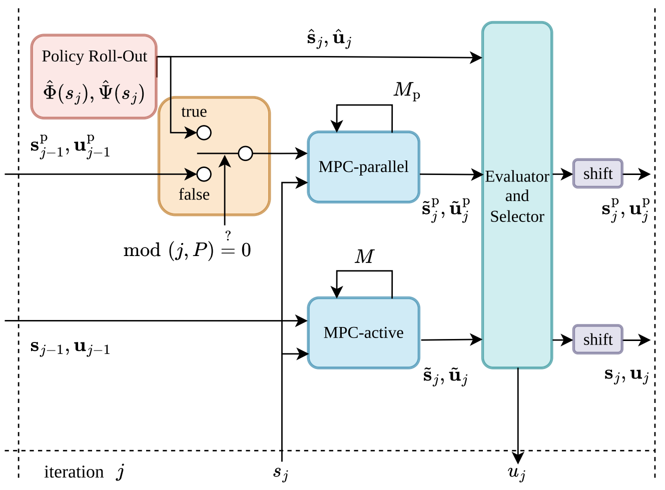
Policy Roll-Out
$\hat{\Phi}(s_j), \hat{\Psi}(s_j)$
$\hat{\mathbf{s}}_j, \hat{\mathbf{u}}_j$
$\mathbf{s}^{\mathrm{p}}_{j-1}, \mathbf{u}^{\mathrm{p}}_{j-1}$
true
false
$\operatorname{mod}(j, P) \overset{?}{=} 0$
M_{p}
MPC-parallel
$\tilde{\mathbf{s}}^{\mathrm{p}}_j, \tilde{\mathbf{u}}^{\mathrm{p}}_j$
M
MPC-active
$\mathbf{s}_{j-1}, \mathbf{u}_{j-1}$
$\tilde{\mathbf{s}}_j, \tilde{\mathbf{u}}_j$
Evaluator and Selector
shift
$\mathbf{s}^{\mathrm{p}}_j, \mathbf{u}^{\mathrm{p}}_j$
shift
$\mathbf{s}_j, \mathbf{u}_j$
iteration j
s_j
u_j

Fig. 1. Sequential algorithm sketch of AC4MPC-RTI. In each iteration, the actor policy is rolled out to obtain a control and state trajectory (red). After each P iteration, the parallel MPC is initialized with the policy rollout (yellow) and, otherwise, by the shifted previous MPC solution. The active MPC is initialized with the lowest-cost trajectory, which can be either the shifted solution from its last iteration, the parallel MPC trajectory, or the policy rollout. The cost is provided by the proposed evaluation algorithm "ac4eval" (green).

Particularly, we propose to use the RTI scheme [10] and multiple shooting [6], which create additional challenges for the algorithm. The solution never fully converges within the RTI scheme. The local optimum is instead tracked over several time steps, as discussed in Section II-B. Therefore, an initial guess provided by a policy rollout may only obtain a lower cost after several QP steps. In fact, the solution obtained after each QP step may even be infeasible for the nonlinear system dynamics due to the multiple shooting formulation, i.e., $F(s, u) - s^+ \neq 0$. The cost of an infeasible trajectory cannot be directly evaluated.

To be compatible with the RTI scheme, AC4MPC-RTI is extended by the following: 1) maintaining a state trajectory $\mathbf{s}$ beside control trajectory $\mathbf{u}$; 2) *allowing* trajectories to converge over multiple controller iterations by maintaining two MPC instances and reinitializing only one of them at all P iterations; and 3) adapting an evaluation algorithm that tackles the challenging cost prediction of a usually infeasible, multiple shooting trajectory. The basic algorithmic parts of AC4MPC-RTI are aligned with AC4MPC, see colored boxes in Algorithms 1 and 2 and Fig. 1.

A parallel RTI iteration scheme for AC4MPC-RTI with different initialization strategies that address 1) and 2) is described in Section IV-A. The evaluation algorithm related to 3) is described in Section IV-B. A schematic overview of AC4MPC-RTI is shown in Fig. 1 and Algorithm 2.

The proposed algorithm is generalizable to several parallel policy rollouts, which could be obtained by different NNs, see *a mixture of experts* [53]. For clarity of exposition, we regard only one rollout in the following.

Algorithm 2 AC4MPC-RTI Closed-Loop Iterations

```
input : Policy $\hat{\pi}(\cdot)$, value function $\hat{Q}(\cdot)$ or $\hat{J}(\cdot)$,
        max. SQP iterations $M, M_{\mathrm{p}}$, re-ini.
        period $P$, correction par. $\alpha$
1  MPC-active ← MPC (6) with $V_f \leftarrow \hat{J}$ or $\hat{Q}$;
2  MPC-parallel ← MPC (6) with $V_f \leftarrow \hat{J}$ or $\hat{Q}$;
3  for $j \leftarrow 0$ to $\infty$ do
4      $s \leftarrow$ state measurement;
5      policy rollout
         $(\hat{\mathbf{s}}, \hat{\mathbf{u}}) \leftarrow (\hat{\Phi}_N(s; \hat{\pi}(\cdot)), \hat{\Psi}_N(s; \hat{\pi}(\cdot)))$;
6      if $(j \mod P) == 0$ then
7          $(\mathbf{s}^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}}) \leftarrow (\hat{\mathbf{s}}, \hat{\mathbf{u}})$;
8          if $j == 0$ then
9              $(\mathbf{s}, \mathbf{u}) \leftarrow (\hat{\mathbf{s}}, \hat{\mathbf{u}})$;
10     initialize MPC-active ← $(\mathbf{s}, \mathbf{u})$;
11     initialize MPC-parallel ← $(\mathbf{s}^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}})$;
12     $M$ SQP iter. for MPC-active;
13     $M_{\mathrm{p}}$ SQP iter. for MPC-parallel;
14     obtain solution $(\mathbf{s}, \mathbf{u}) \leftarrow$ MPC-active;
15     obtain solution $(\mathbf{s}^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}}) \leftarrow$ MPC-parallel;
16     if $\mathrm{ac4eval}(\mathbf{s}^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}}) \leq \mathrm{ac4eval}(\mathbf{s}, \mathbf{u})$ then
17         $(\mathbf{s}, \mathbf{u}) \leftarrow (\mathbf{s}^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}})$
18     if $\mathrm{ac4eval}(\hat{\mathbf{s}}, \hat{\mathbf{u}}) \leq \mathrm{ac4eval}(\mathbf{s}, \mathbf{u})$ then
19         $(\mathbf{s}, \mathbf{u}) \leftarrow (\hat{\mathbf{s}}, \hat{\mathbf{u}})$
20     apply $u \leftarrow \mathbf{u}[0]$ to the system;
21     shifting $(\mathbf{s}, \mathbf{u}) \leftarrow \xi(\mathbf{s}; \hat{\pi}(\cdot)),\ \zeta(s, \mathbf{u}; \hat{\pi}(\cdot))$;
22     shifting $(\mathbf{s}^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}}) \leftarrow \xi(\mathbf{s}^{\mathrm{p}}; \hat{\pi}(\cdot)),\ \zeta(s^{\mathrm{p}}, \mathbf{u}^{\mathrm{p}}; \hat{\pi}(\cdot))$;
23
```

A. Parallelization

In AC4MPC-RTI, two MPC instances and the policy rollout are evaluated in each time step. In the *parallel MPC*, the candidate trajectories $\hat{\mathbf{s}}_j = \hat{\Phi}_N(s_j; \hat{\pi}(\cdot))$ and $\hat{\mathbf{u}}_j = \hat{\Psi}_N(s_j; \hat{\pi}(\cdot))$ obtained from the policy rollout are used as an initial guess for an MPC (6) (see Fig. 1). The RTI scheme, then, performs M SQP iterations starting from this initial guess. We use RTI over several closed-loop time steps j to allow the solver to converge. Particularly, the actor does not initialize the MPC solver in each iteration j but instead in all $P \in \mathbb{N}^+$ time steps, where $\mathrm{mod}(j, P) = 0$. If $\mathrm{mod}(j, P) \neq 0$, the initial guess for the parallel MPC solver is obtained by *shifting*. The active solver uses the RTI scheme with the previous shifted solution, where the previous solution is the lowest-cost solution of either the active solver, the parallel solver, or the pure actor rollout. Shifting, as described previously for AC4MPC, shifts the primal variable of the MPC problem and simulates the system with the actor for the very last initial state, i.e., the controls are shifted by $\zeta(s, \mathbf{u}; \hat{\pi}(\cdot))$, and the states $\mathbf{s}$ are shifted by

$$\xi(\mathbf{s}; \hat{\pi}(\cdot)) := (s_1, \ldots, s_N, F(s_N, \hat{\pi}(s_N))).$$

If the trajectory $(\hat{\mathbf{s}}_j, \hat{\mathbf{u}}_j)$ obtained from the policy rollout or the parallel optimized trajectory $(\tilde{\mathbf{s}}^p_j, \tilde{\mathbf{u}}^p_j)$ is superior in terms of the evaluated cost (see Section IV-B), the related states are used to initialize the active solver in the next iteration. Given that the evaluated cost of the trajectory $(\tilde{\mathbf{s}}_j, \tilde{\mathbf{u}}_j)$ obtained from the active solver is lowest, the active solver is not reinitialized with any policy, rather RTIs, or generally, M SQP iterations are performed with successively starting at the shifted previous solution. This guarantees that AC4MPC-RTI performs at least as well as an MPC formulation using RTI, while also incurring the computational burden of parallel policy evaluations.

B. Evaluation

After each iteration, the candidates $(\hat{\mathbf{s}}_j, \hat{\mathbf{u}}_j)$ obtained from the policy rollout, the parallel sequentially optimized rollouts $(\tilde{\mathbf{s}}^p_j, \tilde{\mathbf{u}}^p_j)$, and the trajectory of the active solver $(\tilde{\mathbf{s}}_j, \tilde{\mathbf{u}}_j)$ are evaluated and ranked among their lowest predicted cost. Evaluating the expected closed-loop cost of the optimization problem defined by (4) solved by multiple-shooting and RTIs is nontrivial due to the following.

First, the problem can only be evaluated on a finite horizon. To approximate the infinite horizon, the critic is used in the evaluator to approximate the infinite horizon cost, such as in the MPC formulation (6).

Second, evaluating the expected closed-loop cost of a multiple-shooting scheme using RTIs is challenging because the dynamics constraints might not be satisfied within the SQP iterations, i.e., the trajectory exhibits gaps [54].

Within globalization strategies of optimization algorithms for multiple-shooting formulations, these gaps are typically combined with the objective via a merit function to obtain a single evaluation criterion. These merit functions need large exact penalties to *outweigh* the other objectives [45]. The merit function serves the purpose of *closing the gaps* over iteration but is not suited to evaluate the expected closed-loop cost due to the somewhat arbitrary choice of weights, given that they are large enough and lead to a numerically stable optimization algorithm. Additionally, with open gaps, the trajectory is not dynamically feasible and, thus, unsuitable for evaluation.

A straightforward method to obtain a feasible trajectory would involve using the controls $\mathbf{u}$ to simulate the system $F(\cdot)$ forward, starting from the current state s. Trivially, the trajectory would be feasible. However, for unstable systems, the obtained state trajectory may differ significantly from the multiple-shooting trajectory $\mathbf{s}$, potentially resulting in a high evaluation cost. Since the open-loop trajectory is recomputed based on the state feedback in each step, the control law would stabilize the obtained closed-loop trajectory. Therefore, simulating the control law for evaluating infeasible trajectories also yields a better cost prediction.

In the following, a feasibility projection method is proposed to evaluate any trajectory $(\mathbf{s}, \mathbf{u})$ of length N that utilizes the actor policy as a correcting control law associated with open gaps. The method involves a homotopy parameter $\alpha \in [0, 1]$ that scales the impact of the correction law. We use an auxiliary control law

$$\bar{s}_{k+1} = F(\bar{s}_k, \bar{u}_k), \quad \bar{u}_k = u_k + \alpha(\hat{\pi}(\bar{s}_k) - \hat{\pi}(s_k)) \tag{25}$$

to simulate the system forward to obtain the simulated controls $\bar{\mathbf{u}} = [\bar{u}_0, \ldots, \bar{u}_{N-1}]$ and states $\bar{\mathbf{s}} = [\bar{s}_0, \ldots, \bar{s}_N]$. A parameter of $\alpha = 0$ would correspond to an open-loop forward simulation

Algorithm 3 ac4eval($\cdot$)

```
input     : Trajectory $\mathbf{s} \in \mathbb{R}^{n_s \times N}, \mathbf{u} \in \mathbb{R}^{n_s \times (N-1)}$
parameter: Policy $\hat{\pi}(\cdot)$, value function $\hat{Q}(\cdot)$ or $\hat{J}(\cdot)$,
           correction parameter $\alpha \in [0,1]$,
           evaluation rollout length $R$
1  Initialize cost $c_\mathrm{r} \leftarrow 0$ ;
2  Initialize state, control $\bar{s}_0 = s_0, \bar{u}_0 = u$ ;
3  for $k \leftarrow 0$ to $N-1$ do
4      get aux. control $\bar{u}_k = u_k + \alpha(\hat{\pi}(\bar{s}_k) - \hat{\pi}(s_k))$;
5      update cost $c_\mathrm{r} \leftarrow c_\mathrm{r} + c(\bar{s}_k, \bar{u}_k)$;
6      simulate system $\bar{s}_{k+1} = f(\bar{s}_k, \bar{u}_k)$;
7  for $k \leftarrow N$ to $N+R$ do
8      update cost $c_\mathrm{r} \leftarrow c_\mathrm{r} + c(\bar{s}_k, \hat{\pi}(\bar{s}_k))$;
9      policy rollout $\bar{s}_{k+1} = f(\bar{s}_k, \hat{\pi}(\bar{s}_k))$;
10 if $\hat{J}(\cdot)$ then
11     terminal cost $c_\mathrm{r} \leftarrow c_\mathrm{r} + \hat{J}(\bar{s}_{N+R})$;
12 else
13     terminal cost $c_\mathrm{r} \leftarrow c_\mathrm{r} + \hat{Q}(\bar{s}_{N+R}, \hat{\pi}(\bar{s}_{N+R}))$;
14 return accumulated cost $c_\mathrm{r}$
```

without feedback. Notably, the auxiliary state trajectory $\tilde{\mathbf{s}}$ obtained from the control law defined in (25) would only differ from the SQP solution of the states $\hat{\mathbf{s}}$ if the states $\hat{\mathbf{s}}$ were infeasible with respect to the dynamics function.

Along the lines of [39] and as discussed in Section III, the value function is approximated by a rollout of the actor policy for R steps at the final state $\bar{s}_N$ to obtain $\bar{s}_{N+1}, \ldots, \bar{s}_{N+R}$ and $\bar{u}_N, \ldots, \bar{u}_{N+R-1}$ and the final critic value at $\bar{s}_{N+R}$ (see Algorithm 3).

C. Parameterization

Besides numerical parameters of the NLP used within the MPC and hyperparameters for the RL training, several parameters specific to AC4MPC-RTI need to be chosen, i.e., the reinitialization parameter $P \in \mathbb{N}_{\geq 1}$, the correction parameter $\alpha \in [0, 1]$, and the evaluation rollout length $R \in \mathbb{N}$. Suppose the system is considered unstable, and the learned policy stabilizes the system empirically. In that case, the parameter α should be closer to 1 to use a policy that stabilizes the potentially open gaps of the shooting nodes. If the learned terminal value function is poor quality, the evaluation rollout length R should be increased. This enables the AC4MPC-RTI algorithm to more effectively evaluate which candidate trajectory yields the lowest open-loop cost.

V. Experiments

Section V highlights the properties and performance of the proposed algorithms in experiments. In Section V-B, the properties of AC4MPC are illustrated on a low-dimensional example. In Section V-C, AC4MPC-RTI is evaluated in a more realistic scenario of time-optimally overtaking vehicles. First, in Section V-A, we discuss some important implementation issues when using NNs within an MPC.

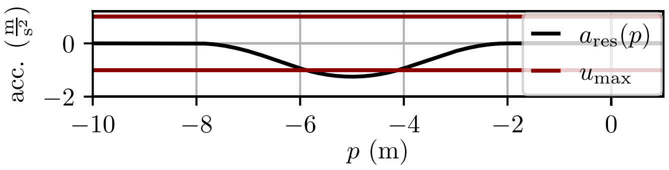

Fig. 2. Acceleration acting on the 1-D vehicle due to a snowy slope and the maximum input acceleration in the *snow hill* environment.

A. Using NNs Within Mpc

Although MPC solvers, such as acaods [55], are capable of solving nonlinear and nonconvex programs, the expected performance depends to a major extent on the local smoothness of the model. NNs may contradict local smoothness requirements, e.g., rectified linear unit (ReLU) networks are not even continuously differentiable. Therefore, the proposed AC4MPC and AC4MPC-RTI algorithms require smooth activation functions, such as tanh-activation functions, which we use in the following experiments.

For AC4MPC, the interior point algorithm ipopt [56] is used to solve the optimization problem to a local optimum. For the AC4MPC-RTI algorithm, we use SQP iterations with the RTI scheme and Gauss–Newton Hessian approximations for the stage costs and the constraints due to their favorable numerical properties [10]. We use the interior point QP solver HPIPM [57] for the QP-subproblems. For the terminal value function, which is an NN in the proposed algorithm, we set the Hessian matrix in the QP subproblems to a diagonal matrix with small entries and only compute first-order derivatives since this increased the numerical robustness in the performed experiments. In the AD example, the nonlinearity of the critic was occasionally preventing the solver from converging. Therefore, the influence of the critic was diminished by multiplying it in the terminal value function by a factor $0 \leq \beta \leq 1$.

For the RTI solver, we used acados [55] and the learning framework L4CasADi [58] to interface Pytorch models. The actor and critic networks were trained using stable-baselines-3 [59].

B. Illustrative Example

To shed light on the fundamental properties of AC4MPC, an illustrative *snow hill* environment is introduced. The environment models a point-mass vehicle with position p and velocity v, with $\dot{p} = v$ and the state $s = [p, v]^\top$. The vehicle moves in one dimension and has to climb a *snowy hill*, which is modeled by an acceleration shown in Fig. 2. The vehicle can be controlled by a bounded acceleration $|u| \leq 1$ m/s^2, leading to the model equation $\ddot{p} = \dot{v} = u + a_\mathrm{res}(p)$. The dynamics are discretized by an RK4 integrator and a discretization time of $t_d = 0.1$ s to yield the discrete-time system $s_{k+1} = F(s_k, u_k)$. Notably, the vehicle must first move away from the hill to gain enough speed to climb the slope. Using the initial state $\hat{s}$, the discrete-time *snow hill* environment OCP is

$$\min_{\substack{s_0,\ldots,s_{N_\mathrm{sim}},\\ u_0,\ldots,u_{N_\mathrm{sim}-1}}} \sum_{k=0}^{N_\mathrm{sim}} \sqrt{s_k^\top Q s_k + 1} + \sum_{k=0}^{N_\mathrm{sim}-1} u_k^\top R u_k$$

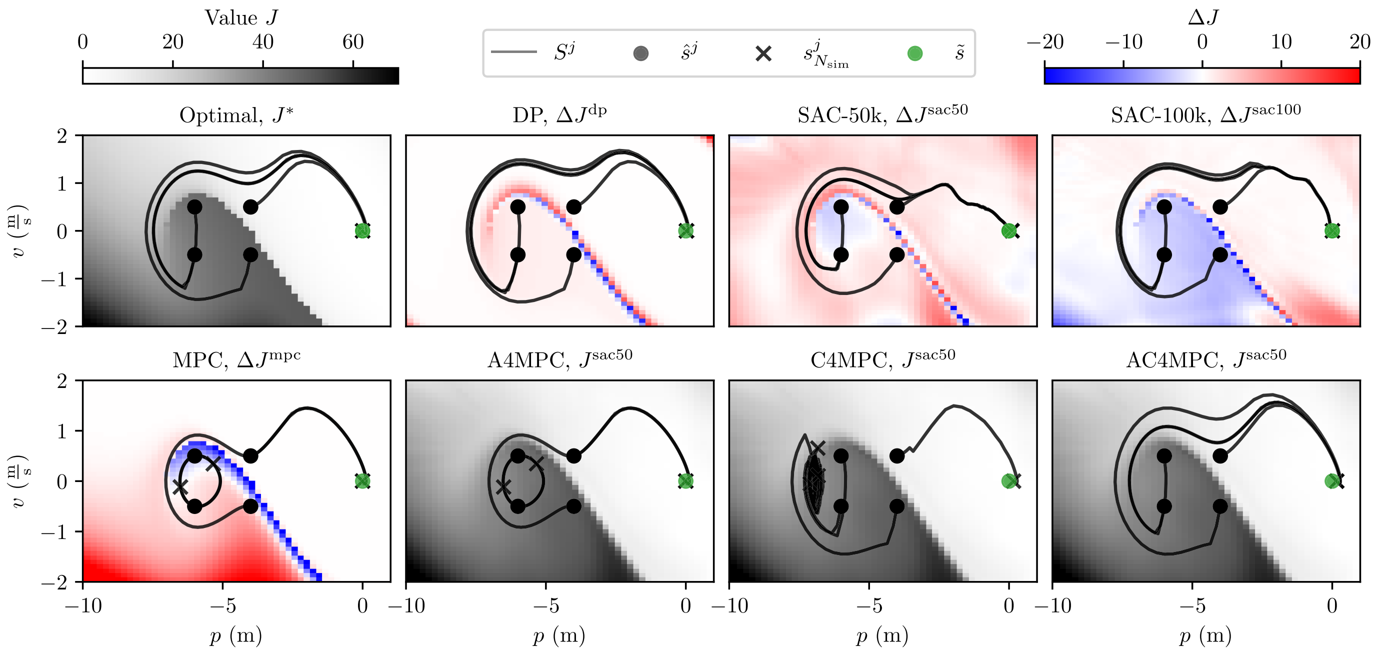

Fig. 3. Comparison of closed-loop trajectories S^j and relevant value functions for different control algorithms applied to the snow hill environment. Optimization solvers of any algorithm were solved toward full local convergence. Closed-loop trajectories are evaluated for four different starting states $\hat{s}^j$, simulating the system for 20 s, following the related policy. The goal state $\tilde{s} = [0, 0]^\top$ can only be reached by certain algorithm variants. The first plot shows the ground-truth value function J^* and trajectories obtained by MPC with a sufficiently long horizon to reach the goal state. The next plot shows trajectories obtained by DP, which are only close to optimal due to the discretization error. The difference $\Delta J^{\mathrm{dp}} = J^* - J^{\mathrm{dp}}$ of the DP value function J^{dp} to the optimal counterpart is shown. The next two plots show simulated trajectories using the actor, as well as the critic value function difference ΔJ^{sac} to the optimal one of SAC RL after $5 \cdot 10^4$ and 10^5 iterations, respectively. The bottom-left plot shows the nominal MPC evaluation by initializing the trajectories at the current state. The value function J^{mpc} corresponds to the open-loop values computed by the NMPC, and we plot $\Delta J^{\mathrm{mpc}} = J^* - J^{\mathrm{mpc}}$. Next, we show the A4MPC, which uses the actor obtained by SAC to initialize each MPC closed-loop iteration but no terminal value function, C4MPC, which uses the last state as the initial guess and the critic of the SAC as terminal value function, and AC4MPC, which uses both. The value functions plotted for A4MPC, C4MPC, and AC4MPC correspond to the SAC critic value J^{sac50}, which is used directly in C4MPC and AC4MPC as the terminal value function, and the related policy rollout is used in A4MPC and AC4MPC-RTI.

$$\text{s.t. } s_0 = \hat{s}, \quad |u_k| \le 1, \quad s_{k+1} = F(s_k, u_k), \quad k \in \mathbb{N}_{N_{\mathrm{sim}}-1}.$$

In the following, different control approaches for the *snow hill* environment and related to AC4MPC are qualitatively compared via samples of closed-loop trajectories and their value functions, as shown in Fig. 3. Furthermore, a quantitative comparison of the obtained closed-loop cost for RL variants, MPC, AC4MPC, and AC4MPC-RTI is given in Fig. 4. For all experiments, we simulate for $N_{\mathrm{sim}} = 200$ steps.

First, the “ground-truth” value function J^* and the policy are obtained by solving the OCP as NLP and fixing the final state to the goal state $\tilde{s} = [0, 0]^\top$ (see top-left plot in Fig. 3). Distinct globally optimal trajectories are shown for different starting states $\hat{s}$. Note that by fixing the final state and using an interior point solver ipopt [56], the solver always converged.

Second, the value function J^{dp} and policy are obtained by DP within a discretization of $\Delta s = [0.05\ \mathrm{m}, 0.05\ \mathrm{m/s}]^\top$ and $\Delta u = 0.01\ \mathrm{m/s}^2$, between $v_{\mathrm{eval}} = [-3, 3]$ m/s and $p_{\mathrm{eval}} = [-12, 4]$ m. DP yields nearly optimal trajectories despite the state discretization error. In Fig. 3, the difference to the optimal value function $\Delta J^{\mathrm{dp}} = J^* - J^{\mathrm{dp}}$ is shown, in addition to example trajectories obtained by following the DP solution at each grid cell.

Moreover, the policy obtained by SAC after $5 \cdot 10^4$ and 10^5 iterations and the critic function J^{sac50} and J^{sac100}, respectively, are evaluated. For both the actor and the critic, feed-forward NNs with two layers of size 256 with tanh-activation functions are used. Notably, in SAC, a Q-value function $Q(s, u)$ is part of the algorithm. The regular value function is obtained by minimizing over the input u in each state. In Fig. 3, it can be verified that the value function is approximated up to a small error, and the optimal policy drives the trajectories suboptimally to the goal state $\tilde{s}$.

Thereafter, the nominal MPC is evaluated using a terminal cost equal to the stage cost and a horizon of $N_{\mathrm{mpc}} = 20$. The MPC is initialized at the current state and solved with the ipopt [56] solver toward convergence in each iteration. Fig. 3 reveals that MPC gets occasionally stuck in local minima and can barely reach the goal state. This is due to the missing terminal value function and initial guesses that lead to poor local minima. Moreover, the horizon is too short to add a terminal constraint for the goal state.

Finally, AC4MPC is evaluated with two ablations. In the ablation named A4MPC, the actor is used to initialize the MPC, but without a terminal value function. In the ablation C4MPC, the critics J^{sac50} and J^{sac100} are used as terminal value functions for the MPC. In C4MPC, the current state is used to initialize the primal variables of the MPC. In Fig. 3 and the performance comparison in Fig. 4, it can be verified that superior performance can only be achieved by using both the actor and the critic, as in AC4MPC. In this example, the AC4MPC outperforms all other variants, including DP in closed-loop performance.

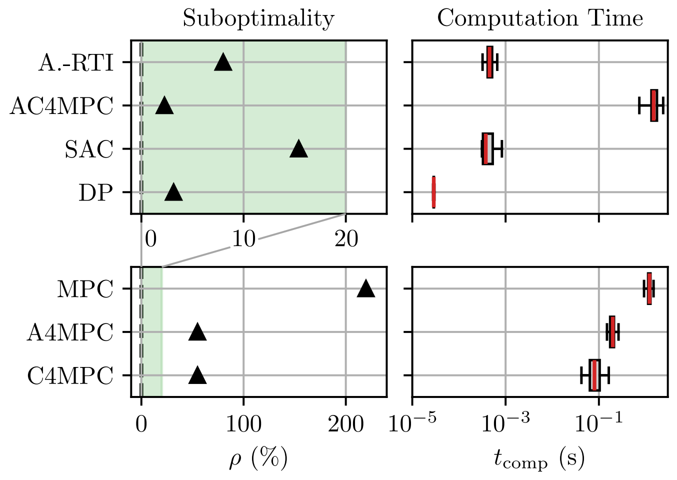

Fig. 4. Comparison of average suboptimality $\rho = (J^{(\cdot)} - J^*)/J^*$ evaluated for closed-loop accumulated costs, corresponding to the closed-loop value functions, for different control algorithms on the *snow hill* environment. The proposed AC4MPC algorithm outperforms all other approaches, including DP. In this example, using either the critic (C4MPC) or the actor (A4MPC) alone leads to high costs, yielding only slight improvements over the nominal MPC. Using AC4MPC incurs a high computational demand due to the need to solve the optimization problem toward convergence in each iteration. AC4MPC-RTI, which only solves one QP instead of the full NLP, significantly reduces the online computation time yet slightly increases the closed-loop cost. The cost of AC4MPC-RTI is considerably lower than the RL SAC cost. The zoomed range in the top plot is highlighted in green.

Finally, in Fig. 4, we quantitatively evaluate AC4MPC-RTI for a horizon of 20 steps and actor and critic networks obtained after 10^5 or $5 \cdot 10^4$ SAC steps, respectively. We used a correction parameter $\alpha = 1$ in Algorithm 3 and an evaluation rollout length $R = 20$. The results in Fig. 4 highlight that AC4MPC-RTI outperforms the corresponding SAC variant. The computation time of AC4MPC-RTI is over two orders of magnitude faster than AC4MPC, yet slower than the SAC policy evaluation. An illustrative single simulation of AC4MPC-RTI, including open-loop planned trajectories, is shown for an initial state $\hat{s}^\top = [-5, -1]$ in Fig. 5. It shows that the solver switches occasionally to the parallel MPC trajectory or the direct policy rollout. The parallel MPC solver is reinitialized in all $P = 5$ steps.

Undoubtedly, the model used in AC4MPC plays a crucial role in evaluating the suboptimality and optimizing the initial guess provided by the actor network. If the model deviates from the true system significantly, the algorithm is expected to perform poorly, as evaluated empirically in Fig. 6 for the *snow hill* environment.

In conclusion, the illustrative example demonstrates that, in general, both the actor and critic approximations may be relevant for the AC4MPC, and that AC4MPC-RTI significantly improves computation time by slightly sacrificing performance. Section V-C gives a more elaborate example of AD using AC4MPC-RTI.

C. Autonomous Driving

The following example considers a practically relevant and more involved autonomous driving scenario with parameters

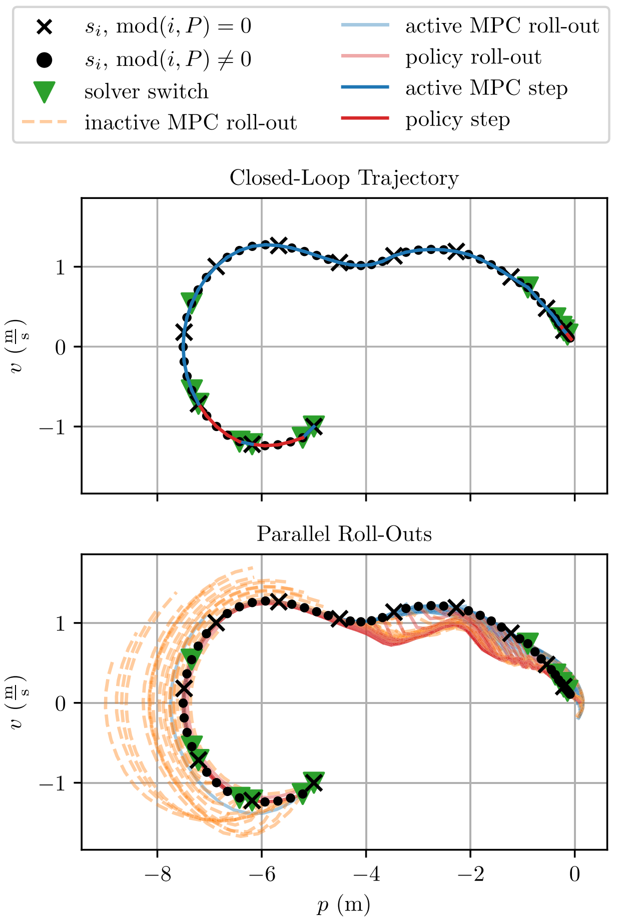

Fig. 5. Phase plot of the AC4MPC-RTI closed-loop trajectory in the *snow hill* environment, starting from the state $s_0 = [-5, -1]^\top$ and ending in the goal state $\tilde{s} = [0, 0]^\top$. At each $P = 5$ iterations (black cross), the parallel MPC is reinitialized using the actor policy. The control corresponding to the lowest-cost trajectory is applied to the system. The top plot shows whether an NMPC control was applied in the current time step (blue) or the proposed RL action (red). Additionally, green triangles indicate if, in the particular time step, the source of the output changed between the NMPC variants or the policy rollout. The bottom plot shows the parallel rollouts of potentially both inactive NMPCs (orange) and the RL rollout (red).

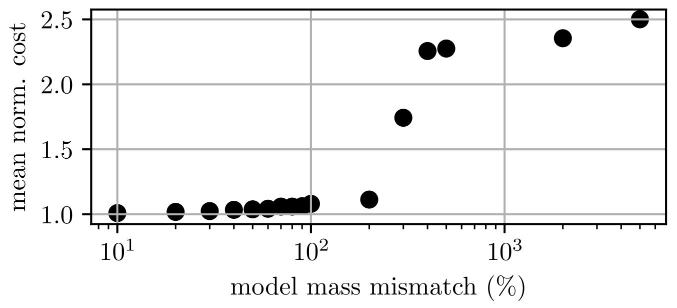

Fig. 6. Robustness of AC4MPC is empirically evaluated in the *snow hill* environment by evaluating the mean performance over 100 closed-loop evaluations with an increasing model-plant mismatch, realized by modifying the mass in the MPC model between 10% and 5000%. Up to a mass mismatch of 200%, the closed-loop cost is barely influenced. However, the performance significantly decreases beyond a mismatch of 200%.

according to the AD framework proposed in [15].[2] The scenario includes a randomized road, i.e., a road that is

[2]AD simulator available at https://github.com/RudolfReiter/vehicle_gym

TABLE I
ONLINE COMPUTATION TIMES (PARALLEL EVALUATION) FOR AD EXAMPLE

approach	*mean (maximum)* computation time in (ms) N=10	N=30	N=60
RL-SAC		0.37 (1.05)	
RL-PPO		0.58 (2.70)	
MPC	0.76 (1.74)	2.53 (3.81)	5.87 (10.15)
AC4MPC-RTI (SAC)	1.12 (1.88)	3.35 (5.47)	7.56 (13.78)
AC4MPC-RTI (PPO)	1.10 (2.86)	3.12 (5.31)	6.30 (8.89)

constructed by randomizing its curvature $\kappa(p_s)$ along the longitudinal position p_s in an interval $[\overline{\kappa}, \underline{\kappa}]$. Two slower surrounding vehicles (SVs) are simulated to follow a reference speed and a curvilinear path at random positions in front of a controlled ego vehicle (EV). All vehicles are simulated as five-state single-track models in the Frenet coordinate frame using ellipsoidal obstacle constraints. The goal of the EV is to overtake the SVs while maintaining a speed limit $\overline{v} = 20$ m/s, considering longitudinal and lateral acceleration constraints $\overline{a}_{\text{lon}} = 3$ m/s^2, $\underline{a}_{\text{lon}} = -12$ m/s, and $\overline{a}_{\text{lat}} = 5$ m/s, respectively, and avoiding collisions. SV trajectories and the related constraints are predicted in each time step assuming constant velocity and a curvilinear motion in the reference frame [15]. We use parameters for the real-world vehicle devbot 2.0 of the competition Roborace [60].

To benchmark comparisons against the proposed AC4MPC-RTI, a nominal MPC that utilizes the RTI scheme is implemented as described in [15]. Moreover, three RL agents are trained using the SAC method with $2 \cdot 10^6$ steps or using PPO with 10^7 steps and different seeds for randomized initial NN weights. The nominal MPC approximates time-optimal driving by avoiding the obstacles, yet without a globalization strategy, i.e., the MPC uses the RTIs purely based on the previous solution. It uses a zero-velocity terminal constraint. For AC4MPC-RTI, the nominal MPC prediction horizons N of $10, 30$, or 60 are compared with a discretization time of $t_d = 0.1$ s, a correction parameter $\alpha = 0$, and a reinitialization parameter $P = 5$. Since, in this example, the primal variables obtained during RTI iterations exhibit only small open gaps, we directly evaluate the multiple shooting trajectory cost, including penalties for open gaps. The SAC and PPO methods learn a critic and actor feed-forward NN of two layers with 256 neurons each and smooth tanh activation functions. The environment state used within this scenario consists of the ego vehicle state, curvature evaluations $\kappa_i = \kappa(p_{s,i})$ with $p_{s,i} = 0, 10, 30, 70, 100, 150$, and 200 m lookahead distance of the current position, and the SV states. In this example, the policy rollout is not evaluated without optimizer iterations, i.e., lines 18 and 19 in Algorithm 2 do not apply.

The algorithms are simulated in 100 random episodes with equal seeds among the approaches. The final closed-loop cost, as defined within the MPC and RL cost functions, is summed for each episode and compared in Fig. 7.

The comparison reveals that the RL policy performs similar to the MPC policy for a prediction horizon of $N = 10$. For a prediction horizon of $N = 30$, the MPC significantly

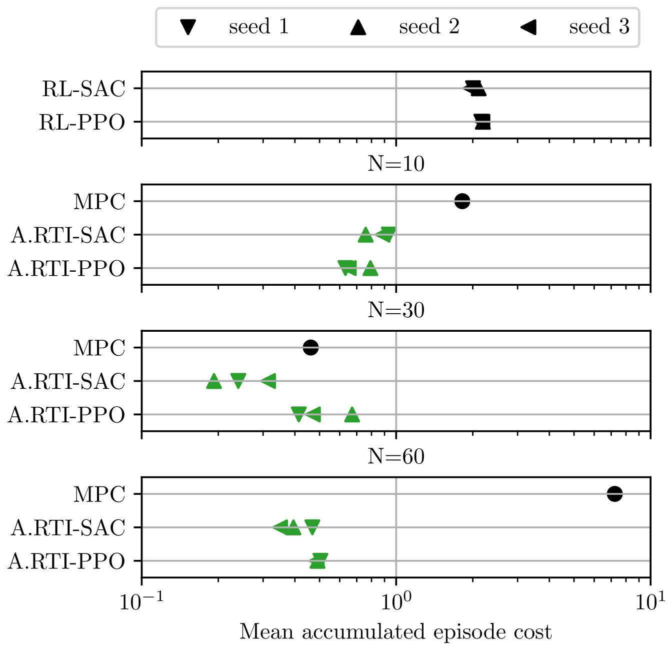

Fig. 7. Accumulated mean episode cost of the AD example with different prediction horizons N for various control algorithms. Three different training seeds were used for algorithms that include NNs. The poor performance of MPC with a large horizon ($N = 60$) stems from the increased sensitivity to the initialization. Due to the initialization strategy of AC4MPC-RTI, the closed-loop cost is much lower compared to MPC for long horizons.

outperforms the RL agents. For longer prediction horizons of $N = 60$, the MPC gets occasionally stuck in local minima created by the obstacle and boundary constraints. This leads to a high closed-loop cost and a worse performance than the RL agents, despite the higher computational demand. Table I shows the average and maximum online solution time returned by the compiled acados [55] solver. For AC4MPC-RTI, it computes the maximum computation time over all solvers, i.e., it assumes parallel processing and synchronization after each iteration. Notably, we do not account for other computation times, as these operations are considered to be significantly faster than solving the optimization problem.

The proposed AC4MPC-RTI algorithm outperforms both baseline approaches in terms of closed-loop cost for both short and longer horizons. For short horizons, the critic NN provides a sufficient approximation for the terminal value function, and the actor NN is of minor importance. The primary cost decrease for longer horizons stems from the actor NN that helps escape from local optima. Notably, in this scenario, it was observed that the critic could also worsen the performance of the AC4MPC-RTI approach. In fact, the critic had to be scaled by a factor of 0.1. Otherwise, the MPC solver acados [55] did not converge sufficiently. This highlights that AC4MPC-RTI requires sufficiently well-trained and relatively smooth NNs to achieve the proposed performance improvement. In fact, tuning the critic to numerical stability was the most challenging part of the proposed algorithm in the AD example. Moreover, in this scenario, the rollout length was set to $R = 0$ because an additional rollout for the evaluation did not significantly improve the performance. Finally,

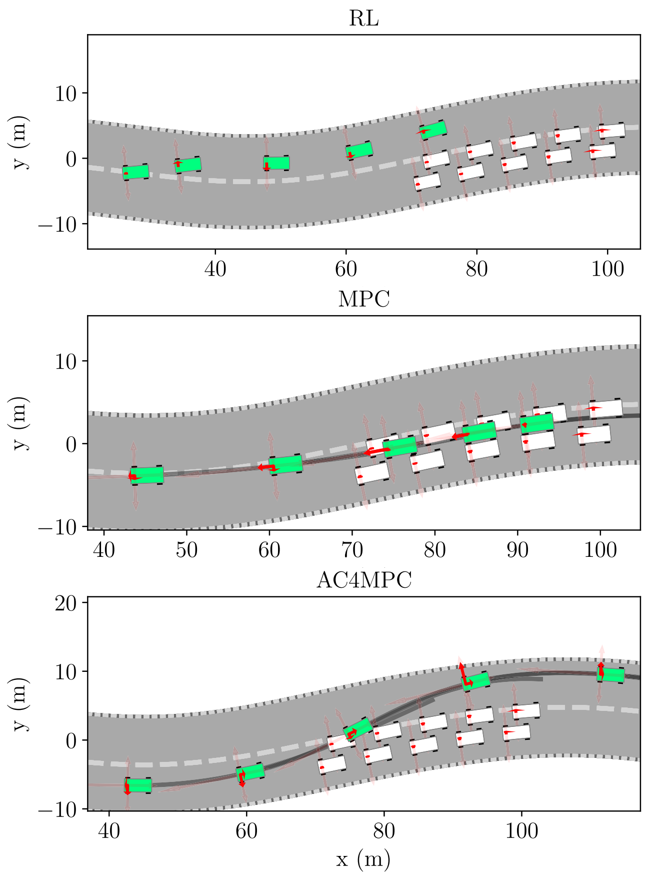

Fig. 8. Snapshots at times $t = 6, 7, 8, 9$, and 10 s for an overtaking maneuver in a randomized scenario of the ego vehicle (green) of two surrounding vehicles (black) for the SAC RL, the MPC, and the AC4MPC-RTI policies. The RL policy lags behind MPC and AC4MPC, while the MPC is stuck in a local minimum behind a leading vehicle. AC4MPC escapes this minimum by a policy rollout and swerves to the right. Red arrows indicate accelerations in the longitudinal and lateral directions in the vehicle coordinate frame. Planned trajectories are plotted in gray.

AC4MPC-RTI exhibits slightly larger online computation times, as shown in Table I.

Exemplary snapshots[3] of the simulation are shown during the critical overtaking maneuver in Fig. 8. The rendering of the simulation reveals that the RL agents progress conservatively and only overtake in the presence of larger gaps. Compared to AC4MPC-RTI, the RL policy does not involve any online optimization, which makes it faster but results in higher costs, given that the MPC model is accurate. As shown in Fig. 8, MPC occasionally gets stuck behind vehicles due to the presence of local minima. AC4MPC-RTI can escape this local minimum due to the critic in the terminal value function and the parallel policy rollouts. Compared to MPC, the AC4MPC-RTI can, therefore, reduce the closed-loop cost at the expense of a slightly larger online computation time.

VI. Conclusion, Discussion, and Outlook

This work proposes a framework that can enhance the performance of nonlinear MPC by utilizing sufficiently well-trained NNs that approximate the optimal policy and optimal value function. Training these networks is the primary goal of RL, and recently, developed software tools (see [61]) offer the possibility of merging these networks with MPC solvers. Under certain assumptions, we have established the theoretical foundation for the proposed improvement in closed-loop performance. Practical, relevant examples provide experimental validation. Notably, the proposed algorithm can be easily parallelized to an ensemble of NNs. The main burden in practical applications is the increased effort to obtain both MPC, necessitating a differentiable model and careful tuning, and RL, requiring a fast simulator. Moreover, the performance of the proposed algorithm depends on the quality of the trained RL networks and the model that approximates the real-world environment. A poorly trained actor may not decrease the overall performance compared to conventional MPC. However, an ill-trained and, hence, highly nonlinear critic used as a terminal value function may lead to numerical instabilities of the optimizer. In this case, the optimization algorithm may fail to converge. We observed such problems in the autonomous driving example of Section V-C. The numerical properties of the value function can be improved by either dedicated optimization problem-solving strategies or by enforcing favorable numerical properties already during the learning, such as in [29]. Fortunately, the proposed algorithm inherits the robustness of MPC to model mismatches [62].

In our particular experiments, the influence of the feasibility parameter α (see Section IV-B) was small. We assume that this is due to the *mildly unstable* systems considered. In the *snow hill* environment and autonomous driving example, the trajectory within MPC has only minor gaps, leading to a negligible influence of the feasibility parameter since it only applies the actor control law for infeasible open gaps. However, we generally expect an increased impact on highly unstable or chaotic systems.

[3]Rendered simulations available: https://rudolfreiter.github.io/ac4mpc_vis/

Acknowledgment

The authors thank Johannes Köhler for his valuable feedback on this work.

References

[1] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed., Madison, WI, USA: Nob Hill Publishing, 2017.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[3] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, "Review on model predictive control: An engineering perspective," *Int. J. Adv. Manuf. Technol.*, vol. 117, nos. 5–6, pp. 1327–1349, 2021.

[4] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, "Value function approximation and model predictive control," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn. (ADPRL)*, Apr. 2013, pp. 100–107.

[5] R. Amrit, J. B. Rawlings, and D. Angeli, "Economic optimization using model predictive control with a terminal cost," *Annu. Rev. Control*, vol. 35, no. 2, pp. 178–186, Dec. 2011.

[6] H. G. Bock and K. J. Plitt, *A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems*. New York, NY, USA: Pergamon, 1984, pp. 242–247.

[7] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[8] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: An overview," in *Proc. 34th IEEE Conf. Decis. control*, Dec. 1995, pp. 560–564.

[9] D. Görges, "Relations between model predictive control and reinforcement learning," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4920–4928, Jul. 2017.

[10] M. Diehl, H. G. Bock, and J. P. Schlöder, "A real-time iteration scheme for nonlinear optimization in optimal feedback control," *SIAM J. Control Optim.*, vol. 43, no. 5, pp. 1714–1736, Jan. 2005.

[11] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 8299–8310.

[12] 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.

[13] S. Gros and M. Zanon, "Reinforcement learning based on MPC and the stochastic policy gradient method," in *Proc. Amer. Control Conf. (ACC)*, May 2021, pp. 1947–1952.

[14] A. Romero, Y. Song, and D. Scaramuzza, "Actor-critic model predictive control," 2023, *arXiv:2306.09852*.

[15] R. Reiter, J. Hoffmann, J. Boedecker, and M. Diehl, "A hierarchical approach for strategic motion planning in autonomous racing," in *Proc. Eur. Control Conf. (ECC)*, 2023, pp. 1–8.

[16] E. Alboni, G. Grandesso, G. P. R. Papini, J. Carpentier, and A. D. Prete, "CACTO-SL: Using sobolev learning to improve continuous actor-critic with trajectory optimization," in *Proc. Learn. Dyn. Control Conf.*, Jun. 2023.

[17] G. Grandesso, E. Alboni, G. P. R. Papini, P. M. Wensing, and A. D. Prete, "CACTO: Continuous actor-critic with trajectory optimization—Towards global optimality," *IEEE Robot. Autom. Lett.*, vol. 8, no. 6, pp. 3318–3325, Jun. 2023.

[18] S. Levine and V. Koltun, "Guided policy search," in *Proc. 30th Int. Conf. Mach. Learn.*, May 2013, pp. 1–9.

[19] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=Byey7n05FQ

[20] H. Wang, S. Lin, and J. Zhang, "Warm-start actor-critic: From approximation error to sub-optimality gap," in *Proc. 40th Int. Conf. Mach. Learn.*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., Jul. 2023, pp. 35989–36019.

[21] M. Klaučo, M. Kalúz, and M. Kvasnica, "Machine learning-based warm starting of active set methods in embedded model predictive control," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 1–8, Jan. 2019.

[22] R. Sambharya, G. Hall, B. Amos, and B. Stellato, "End-to-end learning to warm-start for real-time quadratic optimization," in *Proc. 5th Annu. Learn. Dyn. Control Conf.*, N. Matni, M. Morari, and G. J. Pappas, Eds., Jun. 2022, pp. 220–234.

[23] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 1494–1499.

[24] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2433–2448, Jun. 2021.

[25] N. Mansard, A. DelPrete, M. Geisert, S. Tonneau, and O. Stasse, "Using a memory of motion to efficiently warm-start a nonlinear predictive controller," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 2986–2993.

[26] Y. Qu et al., "RL-driven MPPI: Accelerating online control laws calculation with offline policy," *IEEE Trans. Intell. Vehicles*, vol. 9, no. 2, pp. 1–12, Feb. 2024.

[27] S. W. Chen, T. Wang, N. Atanasov, V. Kumar, and M. Morari, "Large scale model predictive control with neural networks and primal active sets," *Automatica*, vol. 135, Jan. 2022, Art. no. 109947.

[28] X. Shen and F. Borrelli, "Reinforcement learning and distributed model predictive control for conflict resolution in highly constrained spaces," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2023, pp. 1–6.

[29] S. Abdufattokhov, M. Zanon, and A. Bemporad, "Learning Lyapunov terminal costs from data for complexity reduction in nonlinear model predictive control," *Int. J. Robust Nonlinear Control*, vol. 34, no. 13, pp. 8676–8691, Sep. 2024.

[30] L. Beckenbach, P. Osinenko, and S. Streif, "A Q-learning predictive control scheme with guaranteed stability," *Eur. J. Control*, vol. 56, pp. 167–178, Nov. 2020.

[31] L. Beckenbach and S. Streif, "Approximate infinite-horizon predictive control," in *Proc. IEEE 61st Conf. Decis. Control (CDC)*, Dec. 2022, pp. 3711–3717.

[32] F. Moreno-Mora, L. Beckenbach, and S. Streif, "Predictive control with learning-based terminal costs using approximate value iteration," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 3874–3879, 2023.

[33] R. Deits, T. Koolen, and R. Tedrake, "LVIS: Learning from value function intervals for contact-aware robot controllers," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 7762–7768.

[34] N. Karnchanachari, M. de la Iglesia Valls, D. Hoeller, and M. Hutter, "Practical reinforcement learning for MPC: Learning from sparse objectives in under an hour on a real robot," in *Proc. 2nd Conf. Learn. Dyn. Control*, vol. 120, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., 2020, pp. 211–224.

[35] G. De Nicolao, L. Magni, and R. Scattolini, "Stabilizing receding-horizon control of nonlinear time-varying systems," *IEEE Trans. Autom. Control*, vol. 43, no. 7, pp. 1030–1036, Jul. 1998.

[36] M. Diehl, L. Magni, and G. D. Nicolao, "Online NMPC of a looping kite using approximate infinite horizon closed loop costing," in *Proc. IFAC Conf. Control Syst. Design*, vol. 36, 2003, pp. 519–524.

[37] M. Diehl, L. Magni, and G. Nicolao, "Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing," *Annu. Rev. Control*, vol. 28, no. 1, pp. 37–45, 2004.

[38] S. Liu and J. Liu, "Economic model predictive control with extended horizon," *Automatica*, vol. 73, pp. 180–192, Nov. 2016.

[39] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *Eur. J. Control*, vol. 11, nos. 4–5, pp. 310–334, Jan. 2005.

[40] J. N. Richard H. Byrd and R. A. Waltz, "Steering exact penalty methods for nonlinear programming," *Optim. Methods Softw.*, vol. 23, no. 2, pp. 197–213, Apr. 2008.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[42] T. Haarnoja et al., "Soft actor-critic algorithms and applications," 2018, *arXiv:1812.05905*.

[43] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn.*, J. Dy and A. Krause, Eds., Jul. 2018, pp. 1861–1870.

[44] J. G. Kuba, C. S. d. Witt, and J. Foerster, "Mirror learning: A unifying framework of policy optimisation," in *Proc. Int. Conf. Mach. Learn.*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu, and S. Sabato, Eds., Jul. 2022, pp. 7825–7844.

[45] J. Nocedal and S. J. Wright, *Numerical Optimization*. Cham, Switzerland: Springer, 1999.

[46] L. Grüne and S. Pirkelmann, "Economic model predictive control for time-varying system: Performance and stability results," *Optim. Control Appl. Methods*, vol. 41, no. 1, pp. 42–64, Jan. 2020.

[47] T. Faulwasser, L. Grüne, and M. A. Müller, "Economic nonlinear model predictive control," *Found. Trends Syst. Control*, vol. 5, no. 1, pp. 1–98, 2018.

[48] M. A. Müller and K. Worthmann, "Quadratic costs do not always work in MPC," *Automatica*, vol. 82, pp. 269–277, Aug. 2017.

[49] D. P. Bertsekas, *Lessons From AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Belmont, MA, USA: Athena Scientific, 2022.

[50] L. Grune and A. Rantzer, "On the infinite horizon performance of receding horizon controllers," *IEEE Trans. Autom. Control*, vol. 53, no. 9, pp. 2100–2111, Oct. 2008.

[51] D. Angeli, R. Amrit, and J. B. Rawlings, "On average performance and stability of economic model predictive control," *IEEE Trans. Autom. Control*, vol. 57, no. 7, pp. 1615–1626, Jul. 2012.

[52] M. A. Müller and L. Grüne, "Economic model predictive control without terminal constraints for optimal periodic behavior," *Automatica*, vol. 70, pp. 128–139, Aug. 2016.

[53] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.

[54] M. J. Tenny, S. J. Wright, and J. B. Rawlings, "Nonlinear model predictive control via feasibility-perturbed sequential quadratic programming," *Comput. Optim. Appl.*, vol. 28, no. 1, pp. 87–121, Apr. 2004.

[55] R. Verschueren et al., "Acados—A modular open-source framework for fast embedded optimal control," *Math. Program. Comput.*, vol. 14, no. 1, pp. 147–183, Oct. 2021.

[56] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.

[57] G. Frison and M. Diehl, "HPIPM: A high-performance quadratic programming framework for model predictive control," in *Proc. IFAC World Congr.*, Jul. 2020, pp. 6563–6569.

[58] T. Salzmann, E. Kaufmann, J. Arrizabalaga, M. Pavone, D. Scaramuzza, and M. Ryll, “Real-time neural MPC: Deep learning model predictive control for quadrotors and agile robotic platforms,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 4, pp. 2397–2404, Apr. 2023.

[59] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *J. Mach. Learn. Res.*, vol. 22, no. 268, pp. 1–8, 2021.

[60] (2019). *Roborace*. [Online]. Available: https://roborace.com/

[61] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryll, “Learning for CasADi: Data-driven models in numerical optimization,” in *Proc. 6th Annu. Learn. Dyn. Control Conf.*, 2023, pp. 541–553.

[62] G. de Nicolao, L. Magni, and R. Scattolini, “On the robustness of receding-horizon control with terminal constraints,” *IEEE Trans. Autom. Control*, vol. 41, no. 3, pp. 451–453, Mar. 1996.

Rudolf Reiter received the master’s degree in electrical engineering from Graz University of Technology, Graz, Austria, in 2016, and the Ph.D. degree from the University of Freiburg, Freiburg, Germany, in 2024.

From 2016 to 2021, he worked as a Control Systems Research Engineer in the automotive and process measurement industry. He joined the Robotics and Perception Group, University of Zurich, Zürich, Switzerland, in 2025, as a Post-Doctoral Researcher. His research focuses on machine learning and optimization-based control.

Andrea Ghezzi received the master’s degree (cum laude) in automation and control engineering from the Politecnico di Milano, Milan, Italy, in 2020. He is currently pursuing the Ph.D. degree with the University of Freiburg, Freiburg, Germany, under the supervision of Dr. Moritz Diehl.

After his graduation, he worked as a Researcher with the Data Science Research and Development Department, Tenaris, Dalmine, Italy. His research interests focus on numerical optimization and control, particularly on algorithms for mixed-integer nonlinear programming.

Katrin Baumgärtner received the master’s degree in computer science from the University of Freiburg, Freiburg, Germany, in 2019, where she is currently pursuing the Ph.D. degree under the supervision of Dr. Moritz Diehl.

Her research interests are structure-exploiting numerical methods for optimal feedback control and open-source software development.

Jasper Hoffmann received the master’s degree in computer science from the University of Freiburg, Freiburg, Germany, in 2020, where he is currently pursuing the Ph.D. degree under the supervision of Dr. Joschka Bödecker.

His research interests focus on reinforcement learning and optimization-based control, particularly on rare events and combining learning and optimization-based control.

Robert D. McAllister received the Bachelor of Chemical Engineering degree from the University of Delaware, Newark, DE, USA, in 2017, and the Ph.D. degree in chemical engineering from the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2022.

He spent six months as a Post-Doctoral Researcher at Delft Center for Systems and Controls (DCSC), Delft University of Technology (TU Delft), Delft, Netherlands, where he is currently an Assistant Professor. His research interests include model predictive control, stochastic and distributional robustness, and data-driven control methods with applications in energy and agricultural systems.

Moritz Diehl (Member, IEEE) received the dual Diploma degree in mathematics and physics from Heidelberg University, Heidelberg, Germany, and Cambridge University, Cambridge, U.K., in 1999, and the Ph.D. degree in optimization and nonlinear model predictive control from the Interdisciplinary Center for Scientific Computing, Heidelberg University, in 2001.

From 2006 to 2013, he was a Professor with the Department of Electrical Engineering, KU Leuven University, Leuven, Belgium, where he was the Principal Investigator with the Optimization in Engineering Center (OPTEC). In 2013, he moved to the University of Freiburg, Freiburg, Germany, where he heads the Systems Control and Optimization Laboratory, Department of Microsystems Engineering (IMTEK), and is also with the Department of Mathematics. His research interests include optimization and control, spanning from numerical method development to applications in different branches of engineering, with a focus on embedded and renewable energy systems.