

Synthesis of Model Predictive Control and Reinforcement Learning: Survey and Classification

Rudolf Reiter*, Jasper Hoffmann*, Dirk Reinhardt, Florian Messerer, Katrin Baumgärtner, Shambhuraj Sawant, Joschka Boedecker, Moritz Diehl, Sébastien Gros

Abstract—The fields of model predictive control (MPC) and reinforcement learning (RL) consider two successful control techniques for Markov decision processes. Both approaches are derived from similar fundamental principles, and both are widely used in practical applications, including robotics, process control, energy systems, and autonomous driving.

Despite their similarities, MPC and RL follow distinct paradigms that emerged from diverse communities and different requirements. Various technical discrepancies, particularly the role of an environment model as part of the algorithm, lead to methodologies with nearly complementary advantages. Due to their orthogonal benefits, research interest in combination methods has recently increased significantly, leading to a large and growing set of complex ideas leveraging MPC and RL.

This work illuminates the differences, similarities, and fundamentals that allow for different combination algorithms and categorizes existing work accordingly. Particularly, we focus on the versatile actor-critic RL approach as a basis for our categorization and examine how the online optimization approach of MPC can be used to improve the overall closed-loop performance of a policy.

Index Terms—Optimal Control, Model Predictive Control, Reinforcement Learning

I. INTRODUCTION

SOLVING Markov decision processes (MDPs) online is a large and active research domain where different research communities have developed various solution approaches [1], [2], [3]. An MDP can be stated as the problem of computing the optimal policy of an agent interacting with a stochastic environment that minimizes a cost function, possibly over an infinite horizon. Two common approaches for obtaining optimal policies are model predictive control (MPC) and reinforcement learning (RL).

Within MPC, an optimization problem that approximates the MDP is solved online, involving a simulation of an internal

This research was supported by DFG via Research Unit FOR 2401 and project 424107692, by the EU via ELO-X 953348, and by NFR through the project Safe Reinforcement Learning using Model Predictive Control (SARLEM, grant number 300172).

Rudolf Reiter, Florian Messerer, Katrin Baumgärtner and Moritz Diehl are with the Department of Microsystems Engineering (IMTEK), University of Freiburg, 79110 Freiburg, Germany (e-mail: {rudolf.reiter, florian.messerer, katrin.baumgaertner, moritz.diehl}@imtek.uni-freiburg.de).

Jasper Hoffmann and Joschka Boedecker are with the Department of Computer Science, University of Freiburg, 79110 Freiburg, Germany (e-mail: {hofmaja, jboedeck}@informatik.uni-freiburg.de).

Dirk Reinhardt, Shambhuraj Sawant and Sébastien Gros are with the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7034 Trondheim, Norway (e-mail: {dirk.p.reinhardt, shambhuraj.sawant, sebastien.gros}@ntnu.no).

*Equal contribution

prediction model. The optimized controls are applied to the real-world environment in a closed loop at each time step. Historically, MPC has been developed within the field of optimization-based control engineering, driven by the success of optimization algorithms, e.g., linear programming [4]. MPC design leverages domain knowledge to compose mathematical models, often by first principles.

In contrast, within the RL framework, a policy is expressed as a parameterized explicit function of the state. The policy is iteratively improved by interacting with the environment and adapting its parameters related to the observed cost. In general, RL algorithms are not required to use models of the environment, but often, models are used during training for offline simulation.

Both MPC and RL found their way into classical real-world control [5]. The applications differ depending on the availability of training data. MPC is used where measurement data is scarce and expensive, and the environment can be described by optimization-friendly models. On the contrary, RL is successfully implemented in settings where lots of training data can be generated [6], [7].

Besides their sampling efficiency, several further advantages and disadvantages of both methods are nearly orthogonal [8], i.e., weaknesses of either approach are strengths of the other. For instance, RL struggles with safety issues [9], whereas MPC can guarantee constraint satisfaction related to a particular environment model [2]. This motivated many authors to combine the advantages and synthesize novel algorithms that build on both approaches.

This paper's contribution is twofold. We first analyze the properties and orthogonal strengths of MPC and RL. Secondly, we propose a systematic overview of how MPC can be combined with RL and provide an extensive literature overview guided by the proposed classification. This survey reviews practical and theoretical work and concludes with a section on available open-source software.

A. Related Work

Several works exist that shed light on various parts of the huge research fields of control systems, machine learning, and optimization. The author of [8] shows relations between MPC and RL on a conceptual level and includes a detailed discussion for discrete-time linear time-invariant constrained systems with quadratic costs. RL is contextualized from the perspective of control systems in [10] where MPC is mentioned briefly as one particular control technique. The authors of [9] survey methods

for safe learning in robotics and also consider the role of MPC. In particular, the authors show how parameterized MPC formulations can be used to guarantee safety during learning and how safety can be integrated into RL in general. The survey of [11] provides an extensive overview of how machine learning is used for combinatorial optimization. While combinatorial optimization problems have a particular structure related to discrete decision variables, similarities can be observed in how artificial neural networks (NNs) are integrated into an optimization problem.

The author in [3] compares RL and MPC from the point of approximate dynamic programming and proposes a unified mathematical framework. In [12], [13], the RL-based program AlphaZero [14], [15] that plays games such as chess, shogi or go, is cast within an MPC framework. It is argued that the lookahead and rollout-based policy improvement used in online deployment plays a crucial role in enabling the success of the respective algorithms.

A high-level survey on how general machine learning concepts are used within MPC is provided by [16]. The authors of [17] have the same focus on investigating general machine learning used as part of MPC and provide technical details for learning models, policy parameters, optimal terminal value functions, and approximations of iterative online optimization. RL is a minor topic of both surveys [16] and [17]. The survey [17] is well aligned with our proposed framework. For instance, it considers approximating the terminal value function to be treated in another category as closed-loop learning. In fact, our proposed framework can be seen as a complementary work to [3] and [17].

Further works compare MPC and RL for particular applications. Similarly to [17], the authors in [18] compare how MPC and machine learning are combined exclusively for automotive applications. The authors in [19] compare RL and MPC specifically applied to energy management in buildings with a focus on data-driven MPC methods.

B. Notation

The MPC and RL literature use different symbols and notations for identical objects, as shown in the comparison [3]. The notation used within this paper is given in Tab. I. In this survey, mostly RL notation is used to unify the language but with few exceptions and the use of control literature synonyms for established language. For instance, we mostly use “environment” instead of “plant” or “system” but may occasionally write “system” when it is more common in the context. Note that we exclusively use the term MDP that also refers to the conceptually equivalent term discrete-time stochastic optimal control.

Also, within the field of RL notation, ambiguities exist. In this survey, we mostly use RL in order to refer to model-free RL algorithms and include imitation learning (IL) due to their conceptual overlap. MPC that learns a model online is often seen as a variant of model-based RL [20]. Since the proposed synthesis approaches are not limited to model learning, and, moreover, model-based RL includes approaches that learn a model without the intention to be used in an optimization problem, we omit a detailed discussion on model-based RL

For the concatenation $z = [x^\top, y^\top]^\top$ of column vectors x and y , we write $z = (x, y)$. Given set \mathcal{A} and set \mathcal{B} , we denote with $\mathcal{B}^{\mathcal{A}}$ the set of all functions that map from \mathcal{A} to \mathcal{B} . Given a set \mathcal{A} , we denote with $\text{Dist}(\mathcal{A})$ the space of all probability distributions or probability measures over \mathcal{A} . Let p be a probability density of a probability distribution over \mathcal{A} , then the support is defined as $\text{supp}(p) := \{a \in \mathcal{A} \mid p(a) > 0\}$.

C. Overview

On a high level, this paper is structured into (1) an introductory part, (2) a comparison, and (3) a classification scheme and survey of synthesis approaches. An overview is provided in Fig. 1.

The introductory part introduces in Sect. II the general problem setting. Sect. III and IV describe the main concepts behind RL and MPC and discuss how they aim at solving the general problem of Sect. II. Both of these sections are split into a conceptual and an algorithmic part, providing the basis for the remainder of this work.

The comparison in Sect. V highlights practical differences between both approaches and surveys applied comparisons. Experts in the field of MPC and RL may skip Sect. III, IV and potentially the comparison in Sect. V.

The remaining paramount sections are devoted to the classification and review of combinations of both approaches. In Sect. VI, essential concepts are introduced to categorize existing combination variants based on the actor-critic framework of RL. Following the proposed categorization, literature that uses MPC as an expert for training an RL agent is summarized in Sect. VII. The most widespread variants using MPC within the policy are outlined in Sect. VIII, and variants that use the MPC to determine the value function at a particular state are surveyed in Sect. IX. An additional Sect. X focusses on important theoretical results from the field of MPC and RL and is aligned with the previous categorization. Current open-source software is presented in Sect. XI.

The work is concluded and discussed in Sect. XII.

II. PROBLEM SETTING

This section describes the Markov decision process (MDP) framework – a central concept for both RL and MPC.

MDPs provide a framework for modeling a discrete-time stochastic decision process. An MDP is defined over a state space \mathcal{S} , an action space \mathcal{A} , and a stochastic transition model

$$P : \mathcal{S} \times \mathcal{A} \rightarrow \text{Dist}(\mathcal{S}) \quad (1)$$

describing a probability distribution over the next states given a current state and action. A stage cost $l(s, a)$ with $l : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \cup \{\infty\}$ defines the cost of each state-action pair, typically discounted by a factor $\gamma \in (0, 1]$. The MDP [21] is then defined by the 5-tuple

$$\mathcal{M} := (\mathcal{S}, \mathcal{A}, P, l, \gamma). \quad (2)$$

Solving the MDP refers to obtaining actions that minimize the discounted stage cost, which is elaborated in the following.

For solving MDPs we introduce stochastic policies $\pi : \mathcal{S} \rightarrow \text{Dist}(\mathcal{A})$ that map a state to a probability distribution over

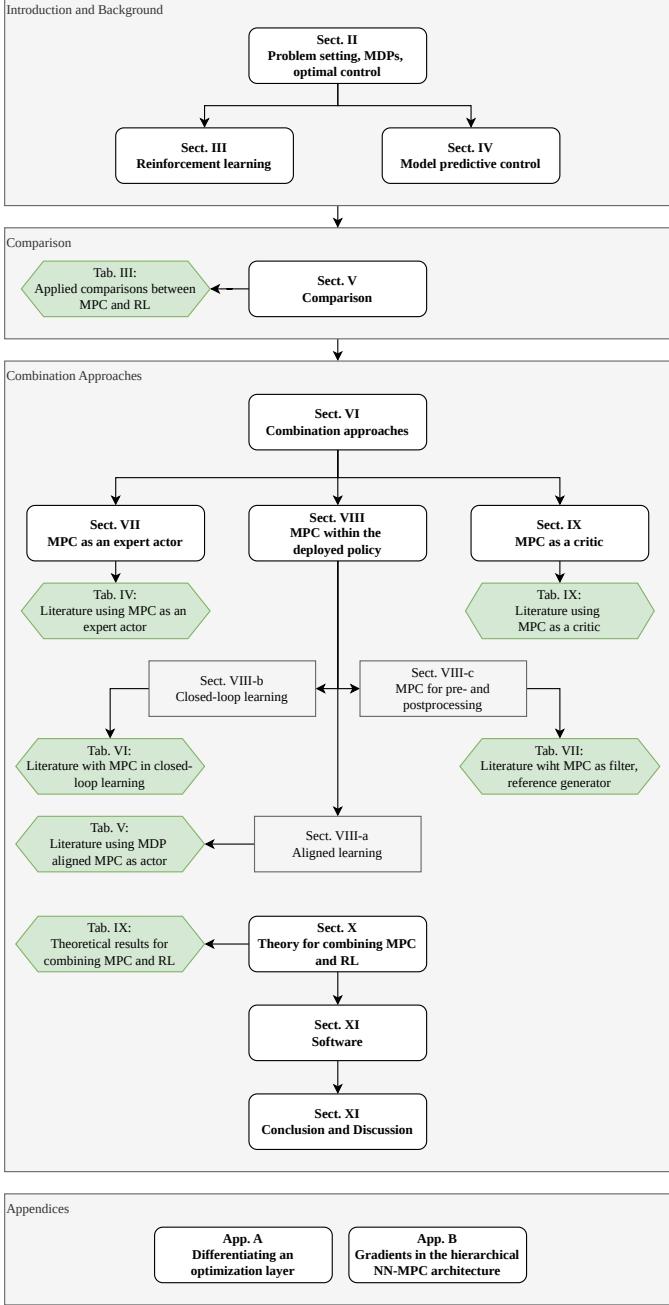


Fig. 1: Paper structure. The main sections are highlighted in gray, subchapters in white, tables are green.

possible actions. The value function $V^\pi : \mathcal{S} \rightarrow \mathbb{R} \cup \{\infty\}$ is the discounted expected future cost starting from state s and following a policy π defined by

$$V^\pi(s) := \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k l(S_k, A_k) \right] \quad (3)$$

$$A_k \sim \pi(\cdot | S_k), \quad S_{k+1} \sim P(\cdot | S_k, A_k), \quad S_0 = s.$$

The state S_k and action A_k describe a sequence of random variables generated by applying the policy π on the MDP.

The aim of solving the MDP is to obtain an optimal policy π^*

TABLE I: Symbols used within this paper.

Name	Symbol
state	s
sampled state	S
planned states within MPC	x
action (control) sampled from policy distribution	a
sampled action	A
planned actions within MPC	u
stochastic model (part of the environment)	$P(s^+ s, a)$
deterministic model approximation within MPC	$f^{\text{MPC}}(x, u)$
stochastic policy (controller)	$\pi(s)$
deterministic policy	$\mu(s)$
parameterized policy	$\pi_\theta(s), \mu_\theta(s)$
cost function (part of the environment)	$l(s, a)$
cost function approximation within MPC	$l^{\text{MPC}}(s, a)$
optimal stochastic policy	$\pi^*(s)$
value function under policy π	V^π
value function under policy π depending on policy parameters θ	$J^\pi(\theta)$
optimal value function	$V^*(s)$
terminal value function approximation under policy π	\bar{V}^π
value function of MPC	$V^{\text{MPC}}(x)$
value function of MPC parameterized by θ	$V_\theta^{\text{MPC}}(x)$
terminal value function of MPC	$\bar{V}^{\text{MPC}}(x)$
terminal value function of MPC parameterized by θ	$\bar{V}_\theta^{\text{MPC}}(x)$
action-value function under policy π	$Q^\pi(s, a)$
optimal action-value function	$Q^*(s, a)$
action-value function of MPC	$Q^{\text{MPC}}(s, a)$
action-value function of MPC parameterized by θ	$Q_\theta^{\text{MPC}}(s, a)$
fixed sampled dataset	\mathcal{D}
data sampled under policy π	\mathcal{D}^π
data sampled under ϵ -greedy policy implicitly defined by Q	\mathcal{D}^{π_Q}
replay buffer with transitions stored during training	$\mathcal{D}^{\text{buffer}}$

that minimizes the expected cost for all states via

$$\pi^* \in \bigcap_{s \in \mathcal{S}} \arg \min_{\pi} V^\pi(s). \quad (4)$$

Solving this equation is often also shortly referred to as solving the MDP. Note that the intersection in (4) is never empty, and there also always exists an optimal deterministic policy μ^* [1] satisfying (4). All optimal policies π^* share the same optimal value function $V^* := V^{\pi^*}$. In addition to the value function in (3), the action-value function $Q^\pi : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \cup \{\infty\}$ of a policy

$$Q^\pi(s, a) := \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k l(S_k, A_k) \right] \quad (5)$$

$$A_k \sim \pi(\cdot | S_k), \quad S_{k+1} \sim P(\cdot | S_k, A_k), \quad S_0 = s, \quad A_0 = a.$$

is the expected value of first applying an action a at the current state s and following the stochastic policy π afterwards.

We define as $V^* := V^{\pi^*}$ and $Q^* := Q^{\pi^*}$ the optimal value and action-value functions. The identities

$$V^*(s) = \min_a Q^*(s, a) \quad (6a)$$

$$\text{supp}(\pi^*(\cdot | s)) \subseteq \arg \min_{a^*} Q^*(s, a^*). \quad (6b)$$

are relating the optimal value function and the optimal policy to the optimal action-value function. In cases with multiple optimal actions for a state s , an optimal policy π^* can be stochastic and randomly selects an action in the set of optimal actions with any probability.

Remark 2.1: The optimal policy for a finite horizon problem is generally time-varying unless the terminal cost is V^* . Time-varying environments can be described in terms of (1) by augmenting the state with an additional clock state.

In the following, we introduce MPC and RL, the two pivotal frameworks of this survey for approximately solving MDPs (4).

III. REINFORCEMENT LEARNING

RL is a powerful approach for solving MDPs using concepts from dynamic programming, Monte Carlo simulation, and stochastic approximation. RL typically involves an agent modeled by a policy iteratively collecting data by interacting with an environment, which could be a simulation model or the real world. The collected data, consisting of state transitions, applied actions, and costs, is then used to iteratively update the policy. In most RL methods, an optimal policy is approximated via estimating the optimal action-value function using temporal difference (TD) learning or by iteratively updating the policy using policy gradient (PG) methods [1], whereas in actor-critic methods both forms are combined.

A. Theoretical Background

In this section, a short theoretical overview of RL is provided, including dynamic programming, temporal difference methods, and policy gradient methods.

1) *Dynamic Programming:* Introduced in [22], dynamic programming (DP) provides the theoretical foundation for many algorithms in RL. DP solves MDPs with known transition models by breaking them into subproblems and using stored solutions to avoid recomputation. DP systematically updates value functions given complete knowledge of the environments MDP. A general DP approach to find the action-value function Q^π of a given policy π can be described by the Bellman operator $T^\pi : \mathbb{R}^{S \times \mathcal{A}} \rightarrow \mathbb{R}^{S \times \mathcal{A}}$,

$$T^\pi Q(s, a) := l(s, a) + \gamma \mathbb{E}_{\substack{S^+ \sim P(\cdot | s, a), \\ A^+ \sim \pi(\cdot | S^+)}} [Q(S^+, A^+)], \quad (7)$$

where S^+ is the next sampled state and A^+ a sampled action. In the space of value functions, the Bellman operator T^π is a contraction mapping for $\gamma < 1$ with respect to the state supremum norm [23], and Q^π is the unique fixed point. In other words, one can start with an arbitrary action-value function $Q \in \mathbb{R}^{S \times \mathcal{A}}$ and iteratively apply T^π to converge to the action-value function Q^π . Determining V^π and Q^π for a fixed policy π is referred to as policy evaluation.

Similar to the Bellman operator T^π , the Bellman optimality operator $T : \mathbb{R}^{S \times \mathcal{A}} \rightarrow \mathbb{R}^{S \times \mathcal{A}}$ is defined as

$$TQ(s, a) := l(s, a) + \gamma \min_{a^+} \mathbb{E}_{S^+ \sim P(\cdot | s, a)} [Q(S^+, a^+)]. \quad (8)$$

Equivalently to T^π , T is a contraction mapping for $\gamma < 1$. Iteratively applying T on an arbitrary action-value function $Q \in$

$\mathbb{R}^{S \times \mathcal{A}}$ converges to the optimal value function Q^* . The resulting method is called value iteration (VI) and, differently to policy evaluation, tries to solve MDPs of (4) implicitly by finding the optimal value function Q^* .

The main drawback of DP methods is the unfavorable scaling to high-dimensional or continuous state and action spaces, which is often referred to as the curse of dimensionality [22]. DP methods require full knowledge of the environment model P , which is a fundamental limitation compared to more generic model-free RL algorithms. Approximate DP (ADP) [24] addresses the curse of dimensionality by using different approximation strategies to extend classical DP, as discussed in the following in the context of RL.

2) *TD Methods:* To avoid the scaling issues of DP, temporal difference (TD) methods [1] introduce two extensions: Firstly, they learn the value functions V^π or Q^π for a policy π and their optimal versions V^* and Q^* with only transition samples utilizing stochastic approximation and without explicitly requiring a transition model P . Secondly, they use an adaptive exploration-exploitation strategy to decide favorable states for which the value function is updated.

In the following, different TD learning algorithms are presented. The simplest policy evaluation method is called TD(0), which estimates the value function V^π for a given policy π . Given a current value function V , an update for a given state S for V is defined by

$$\text{TD(0)} (V \approx V^\pi)$$

$$V(S) \leftarrow V(S) + \alpha \delta, \quad (9a)$$

$$\delta := l(S, A) + \gamma V(S^+) - V(S), \quad (9b)$$

$$\text{with } S \sim \mathcal{D}^\pi, A \sim \pi(\cdot | S), S^+ \sim P(\cdot | S, A), \quad (9c)$$

where \leftarrow denotes overwriting the function V at S , $\alpha > 0$ denotes the learning rate, $A \sim \pi(\cdot | S)$ a sampled action from the policy π at state S and $S^+ \sim P(\cdot | S, A)$ the next state sampled from the stochastic model (1). Furthermore, δ is called the temporal difference, measuring the stochastic difference between the sampled target $l(S, A) + \gamma V(S^+)$ and the current estimate $V(S)$. Finally, the distribution \mathcal{D}^π is a sampling or exploration strategy where the fixed policy π sequentially generates new states by interacting with the environment. For a more technical discussion, see Remark 3.1.

Remark 3.1: The notations $S \sim \mathcal{D}^\pi$ or $(S, A) \sim \mathcal{D}^\pi$ are mathematically not well defined as distributions. Without further elaboration, given an initial state s , an episode is simply run iteratively by applying a policy π :

$$S_0 = s, A_0 \sim \pi(\cdot | S_0), S_1 \sim P(\cdot | S_0, A_0), A_1 \sim \pi(\cdot | S_1), \dots \quad (10)$$

After each time step, an update can then be performed by a TD method. We still use this notation for instructional purposes, especially to highlight which policy was used to generate states and actions.

Similar to DP, under some technical assumptions from stochastic approximation theory, the update scheme converges to the unique fixpoint V^π [23]. For example, one assumption

is that the learning rate α must decrease to zero over time. The update scheme of (9) can be extended to also learn Q^π .

Besides estimating the value function V^π or Q^π for given policies π , a main target of RL algorithms is learning the optimal value functions V^* and Q^* . For instance, the SARSA algorithm [1] is an RL method for approximating the optimal value function Q^* and given by the update rule

SARSA ($Q \approx Q^*$)

$$Q(S, A) \leftarrow Q(S, A) + \alpha \delta, \quad (11a)$$

$$\delta := l(S, A) + \gamma Q(S^+, A^+) - Q(S, A), \quad (11b)$$

$$\text{with } (S, A) \sim \mathcal{D}^{\pi_Q^\epsilon}, S^+ \sim P(\cdot | S, A), A^+ \sim \pi_Q^\epsilon(\cdot | S^+). \quad (11c)$$

The ϵ -greedy policy is implicitly defined by Q via

$$\pi_Q^\epsilon(a | s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a = \arg \min_{a^*} Q(s, a^*), \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{otherwise,} \end{cases} \quad (12)$$

where for notational convenience, it is assumed that there is only one optimal action for a given state s . Note that π_Q^ϵ is used to sample the next action $A^+ \sim \pi_Q^\epsilon$ and to generate the state and action with $D^{\pi_Q^\epsilon}$, where the update is performed.

Another variant for learning the optimal action-value function Q^* is the popular Q-learning method [25], which, instead of sampling an ϵ -greedy action like SARSA, takes the greedy action to build the temporal difference. More explicitly, the update rule is defined via

Q-learning ($Q \approx Q^*$)

$$Q(S, A) \leftarrow Q(S, A) + \alpha \delta, \quad (13a)$$

$$\delta := l(S, A) + \gamma \min_{a^*} Q(S^+, a^*) - Q(S, A), \quad (13b)$$

$$\text{with } (S, A) \sim \mathcal{D}^{\pi_Q^\epsilon}, S^+ \sim P(\cdot | S, A). \quad (13c)$$

Note that Q-learning still uses the ϵ -greedy exploration policy to sample $(S, A) \sim \mathcal{D}^{\pi_Q^\epsilon}$ in (13c), whereas the policy that is used for the temporal difference is the greedy policy. Thus, Q-learning is called an off-policy method, whereas SARSA is an on-policy method as the same ϵ -greedy policy is used for exploration and the temporal difference update. For a more elaborate discussion, see [1].

As discussed, an essential motivation for TD methods is that learning a value function on the whole state space can be intractable, which is one of the major drawbacks of DP. Importantly, typically only a fraction of the state space is encountered under a given policy or an optimal policy. Thus, TD methods often incorporate a trade-off between exploring the state space and exploiting current knowledge using strategies like the ϵ -greedy policy π_Q^ϵ . Yet, for discrete state spaces, convergence to Q^* is only guaranteed if each state and action is seen infinitely often [23].

3) *Policy Gradient Methods:* Different from the previous methods, policy gradient (PG) methods directly optimize a

parameterized policy π_θ that can be used for discrete and continuous action spaces \mathcal{A} . Given a parameterized policy $\pi_\theta : \mathcal{S} \rightarrow \text{Dist}(\mathcal{A})$ and an initial state distribution $\rho_0 \in \text{Distr}(\mathcal{S})$, the goal of PG methods is to find the optimal parameters θ^* that minimize the expected return

$$J^\pi(\theta) := \mathbb{E}_{S \sim \rho_0} [V^{\pi_\theta}(S)] \quad (14a)$$

$$= \frac{1}{1 - \gamma} \mathbb{E}_{S \sim \rho^{\pi_\theta}(s), A \sim \pi_\theta(\cdot | S)} [l(S, A)], \quad (14b)$$

where ρ^π is the discounted visitation frequency defined as follows. Given a policy π_θ , the environment model P and an initial state s , let $p(s \rightarrow s^+, k, \pi_\theta)$ be the probability of reaching state s^+ at time step k by starting from state s following policy π_θ . The normalized discounted visitation frequency is defined by $\rho^{\pi_\theta}(s^+) := (1 - \gamma) \mathbb{E}_{S \sim \rho_0} [\sum_{k=1}^{\infty} \gamma^{k-1} p(S \rightarrow s^+, k, \pi_\theta)]$. It is important to highlight that finding a policy that minimizes the objective of (14) is less restricting as solving the MDP over the full state space as defined in (4). The reasoning is that some parts of the state space might not be reached by the policy, which could be omitted if the support of the initial state distribution ρ_0 is required to cover the whole state space.

In order to find an update direction in which the expected return (14) improves, its gradient $\nabla_\theta J^\pi(\theta)$ is required. Different reformulations of (14) exist that allow one to derive sample estimates of the gradient of the PG objective, $\nabla_\theta J^\pi(\theta)$. First, the stochastic policy-gradient theorem reformulates the policy gradient by

$$\nabla_\theta J^\pi(\theta) = \frac{1}{1 - \gamma} \mathbb{E}_{S \sim \rho^{\pi_\theta}(s), A \sim \pi_\theta(\cdot | S)} [Q^{\pi_\theta}(S, A) \nabla_\theta \log \pi_\theta(A | S)]. \quad (15)$$

building the theoretical foundation of the REINFORCE algorithm [26] or the first actor-critic methods [1]. A derivation is provided in [1].

The second reformulation is called the deterministic policy gradient theorem. Let $\mu_\theta : \mathcal{S} \rightarrow \mathcal{A}$ be a deterministic policy. By differentiating through the expected state-action value function Q^{μ_θ} the deterministic PG (DPG) is obtained by

$$\nabla_\theta J^\pi(\theta) = \frac{1}{1 - \gamma} \mathbb{E}_{S \sim \rho^{\pi_\theta}(s)} \left[\nabla_\theta \mu_\theta(S) \nabla_a Q^{\mu_\theta}(S, a) \Big|_{a=\mu_\theta(S)} \right]. \quad (16)$$

A derivation is provided in [27]. The advantage of the DPG is particularly prominent in high-dimensional action spaces since in the stochastic PG, actions A are sampled to estimate the gradient [27], leading potentially to a higher gradient variance. Thus, the DPG formulation is used in many of the state-of-the-art methods like twin-delayed actor-critic (TD3) [28] and soft actor-critic (SAC) [29]. A concrete algorithm for an actor-critic algorithm using the DPG is provided in section III-B2.

An important distinction between the stochastic PG and the DPG is the ability to handle discrete action spaces. With the stochastic PG, discrete and continuous action spaces can be directly optimized, whereas the DPG requires a differentiable parameterized policy with respect to the parameters. To circumvent this problem, some work extends the DPG to

stochastic policies using relaxation techniques to differentiate through the sampling process of the discrete actions [30].

B. Deep Reinforcement Learning Methods

In the following, an overview of four influential deep RL algorithms, namely deep Q-networks (DQN), deep deterministic PG (DDPG), proximal policy optimization (PPO) and soft actor-critic (SAC) is given.

1) Deep Q-Networks: Function approximators like NNs approximate the action-value function to extend Q-learning to continuous state spaces. One prominent implementation of this is DQN [6], a combination of deep learning and RL. DQN collects transition samples in a buffer $\mathcal{D}^{\text{buffer}}$ and minimizes the mean squared error between the current value function Q_w and the sampled target value by

$$\mathcal{L}_{\text{DQN}}^Q(w) := \mathbb{E}_{(S, A, S^+) \sim \mathcal{D}^{\text{buffer}}} \left[\left(l(S, A) + \gamma \min_{a^*} Q_{\bar{w}}(S^+, a^*) - Q_w(S, A) \right)^2 \right]. \quad (17)$$

For the sample target, a fixed copy \bar{w} of the parameter w is used that is only periodically updated. This stabilizes the training [6]. For an overview of different function approximation methods and their potential instabilities, see [1].

Given a parameterized Q-function Q_w , the resulting update scheme from DQN is given by

DQN ($Q_w \approx Q^*$)

$$w \leftarrow w + \frac{\alpha_w}{B} \sum_{i=1}^B \delta_i \nabla_w Q_w(S_i, A_i), \quad (18a)$$

$$\delta_i := l(S_i, A_i) + \gamma \min_{a^*} Q_{\bar{w}}(S_i^+, a^*) - Q_w(S_i, A_i), \quad (18b)$$

$$\text{with } (S_i, A_i, S_i^+) \sim \mathcal{D}^{\text{buffer}}. \quad (18c)$$

The update rule in (18a) provides a sample-based estimate of the gradient in (17), where B represents the batch size used for the update and α_w a learning rate. Averaging over multiple samples is often called mini-batch training and leads to improved performance and accelerated convergence when training NN [31]. Exploration in DQN is handled again by the ϵ -greedy policy π_Q^ϵ . Differently to the update scheme of Q-learning (13), the buffer $\mathcal{D}^{\text{buffer}}$ stores state and actions that were generated from previous policies derived from Q_w earlier in the training. Similar to Q-learning, DQN is also an off-policy method.

2) Deep Deterministic Policy Gradient: Extending DQN, deep deterministic PG (DDPG) [32] is an off-policy actor-critic method that learns a deterministic policy μ_θ and a value function Q_w in parallel. Both the actor and the critic are NNs. The update rule of DDPG is given by

DDPG ($\mu_\theta \approx \pi^*$)

$$w \xleftarrow{Q} w + \frac{\alpha_w}{B} \sum_{i=1}^B \delta_i \nabla_w Q_w(S_i, A_i), \quad (19a)$$

$$\theta \xleftarrow{\mu} \theta + \frac{\alpha_\theta}{B} \sum_{i=1}^B \nabla_\theta \mu_\theta(S_i) \nabla_a Q_w(S_i, a)|_{a=\mu_\theta(S_i)}, \quad (19b)$$

$$\delta_i := l(S_i, A_i) + \gamma Q_{\bar{w}}(S_i^+, A_i^+) - Q_w(S_i, A_i), \quad (19c)$$

$$\text{with } (S_i, A_i, S_i^+) \sim \mathcal{D}^{\text{buffer}}, A_i^+ = \mu_\theta(S_i^+), \quad (19d)$$

where \xleftarrow{Q} denotes the update for Q and $\xleftarrow{\mu}$ denotes the update for μ . As can be seen from the update scheme, the critic Q_w is used to update the actor μ_θ in (19b), in that sense “criticizing” the actor. Differently to the update rule of DQN (18a), where the greedy action is considered to build the temporal difference, in DDPG, a single evaluation update with respect to the current policy μ_θ is performed. The buffer $\mathcal{D}^{\text{buffer}}$ is filled up over time by using the policy $\mu_\theta + \xi$, where ξ is either an Ornstein-Uhlenbeck process for temporally correlated noise or a Gaussian [32].

3) Proximal-Policy Optimization: A popular on-policy actor-critic method that can be used for discrete and continuous action spaces is proximal policy optimization (PPO) [33]. Prior to each policy and critic update, data in the form of multiple episodes is collected. Since PPO is an on-policy method, transitions generated earlier in the training by outdated policies are discarded. In practice, PPO is often used with high-speed simulation environments, where generating new samples comes with low computational costs. A main advantage of PPO is its ability to prevent drastic parameter updates that could potentially destabilize the training. Similar to trust region methods [34], this is achieved by restricting the policy update.

During training, a stochastic policy π_θ – often a parameterized Gaussian – is used. Assuming a given initial state s_0 , a trajectory is drawn by the forward simulation $S_{k+1} \sim P(\cdot|S_k, A_k)$ and $A_k \sim \pi_\theta(\cdot|S_k)$ until a maximum roll-out length M . Given multiple roll-outs, an estimate $\hat{A}(S_k, A_k)$ of the advantage function defined by $A^\pi(S_k, A_k) := Q^\pi(S_k, A_k) - V^\pi(S_k)$ can be derived, see [35]. Additionally, with the probability ratio $R_k(\theta) := \pi_\theta(A_k|S_k)/\pi_{\bar{\theta}}(A_k|S_k)$, which measures how much the new policy π_θ changes with respect to the current policy $\pi_{\bar{\theta}}$, the PPO clipping objective is defined by

$$J_{\text{CLIP}}^\pi(\theta) := \mathbb{E} \left[\sum_{k=0}^{M-1} \max \left\{ R_k(\theta) \hat{A}(S_k, A_k), \text{clip}(R_k(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}(S_k, A_k) \right\} \right],$$

where the clip function projects the ratio R_k to an interval from $1 - \epsilon$ to $1 + \epsilon$. Note that the clipping objective requires maximization, whereas the original PPO objective involves minimization, as in this work, costs are minimized rather than rewards being maximized.

One of the primary advantages of PPO is its simplicity and ease of implementation [33] compared to previous methods

based on trust region optimization, see [34]. A proof of convergence of PPO to the optimal policy for discrete MDPs is given in [36].

4) *Soft Actor-Critic*: The SAC algorithm [29], [37] is a widely used off-policy actor-critic method that incorporates an entropy bonus for stochastic policies with higher entropy. Using entropy regularization is considered in the framework of maximum-entropy RL [38]. Like DQN and DDPG, SAC optimizes the policy in an off-policy manner collecting transitions encountered during training in a replay buffer, leading to an improved sample efficiency when compared to PPO.

SAC extends the objective (14) by introducing an entropy regularization term, leading to a “soft” policy gradient objective

$$J_{\text{soft}}^{\pi}(\theta) := \frac{1}{1 - \gamma} \mathbb{E}_{S \sim \rho^{\pi_\theta}, A \sim \pi_\theta(\cdot | S)} [l(S, A) + \lambda_{\mathcal{H}} \mathcal{H}(\pi_\theta(\cdot | S))],$$

where \mathcal{H} denotes the entropy of the parameterized policy $\pi_\theta(\cdot | S)$ at the sampled state S . Given a distribution X , the entropy is defined by $\mathcal{H}(X) = \mathbb{E}[-\log(X)]$. The entropy regularization, scaled by the parameter $\lambda_{\mathcal{H}}$, encourages exploration, stabilizes policy training [39] and can lead to more robust policies [40]. Additionally, it has been shown in [41] that for discrete MDPs, the error introduced by the entropy regularization decreases exponentially to the inverse regularization strength, $1/\lambda_{\mathcal{H}}$.

In TD3, twin Q-networks addressing the overestimation bias in Q-value estimation were introduced [28]. Whereas in [29], the fundamentals of SAC are described, [37] introduced an improved version using twin Q-networks and automatic tuning of the entropy regularization with $\lambda_{\mathcal{H}}$. The latter builds the basis for most implementations in current RL software frameworks.

IV. MODEL PREDICTIVE CONTROL

This section introduces MPC, a commonly used framework to obtain policies for continuous MDPs. MPC utilizes a – typically deterministic – model that approximates the true stochastic environment [2], [42]. Starting from the current environment state, MPC uses the internal model to predict how different choices of the planned control trajectory would affect the state trajectory and evaluates the cost associated with this prediction. Usually, evaluating the MPC policy involves solving an optimization problem online to obtain the control input. In this section, we will first give an overview of MPC problem formulations and the relevant considerations, followed by a discussion of algorithms used for finding their solution.

A. MPC Problem Formulations

Solving the MDP optimization problem (4) is, in general, intractable due to several reasons, including the infinite horizon, the optimization over the space of policy functions, and the expectation over nonlinear transformations of stochastic variables. MPC leverages several approximations of (4) in order to derive a computationally tractable optimization problem.

As a first step, the optimal policy is computed only for the current state s and the infinite horizon in (3) and (4) is

approximated by a finite horizon, resulting in the optimization problem

$$\min_{\pi} \mathbb{E} \left[\gamma^N \bar{V}^{\pi}(S_N) + \sum_{k=0}^{N-1} \gamma^k l(S_k, A_k) \right], \quad (20)$$

$$\text{where } S_0 = s, S_{k+1} \sim P(\cdot | S_k, A_k), A_k \sim \pi(\cdot | S_k),$$

where the terminal cost function \bar{V}^{π} is an approximation of the exact value (or cost-to-go) function V^{π} . In the second step, the true stochastic state transition $S_{k+1} \sim P(S_k, A_k)$ is approximated by a simplified model. The most commonly used formulation is nominal MPC, in which a deterministic model is used, i.e., $x_{k+1} = f^{\text{MPC}}(x_k, a_k)$. Hence, uncertainty is not explicitly considered. Here, we introduced $x_k \in \mathcal{S}$ to denote predictions of the state within the MPC problem. Since a deterministic model does not capture the possibility of deviations from the prediction, the planned action trajectory is a trajectory of fixed actions (u_0, \dots, u_{N-1}) , $u_k \in \mathcal{A}$, as opposed to a policy function π . The resulting deterministic optimal control problem is given by

$$\min_{u_0, \dots, u_{N-1}} \gamma^N \bar{V}^{\star}(x_N) + \sum_{k=0}^{N-1} \gamma^k l(x_k, u_k), \quad (21)$$

$$\text{where } x_0 = s, x_{k+1} = f^{\text{MPC}}(x_k, u_k),$$

where \bar{V}^{\star} is an approximation of the optimal terminal value function.

In the MDP framework (2), the stage cost $l(x_k, u_k)$ may assign some regions of the state space an infinite cost in order to prohibit them. Similarly, due to actuator limitations, the action space \mathcal{A} is often a compact subset of \mathbb{R}^{n_u} . In numerical optimization, this is typically handled by explicitly considering constraints $h^{\text{MPC}}(x_k, u_k)$ and the terminal safe set $h_N^{\text{MPC}}(x_k, u_k)$ as part of the problem formulation. This results in a constrained nonlinear program (NLP), associated with the MPC value function $V^{\text{MPC}}(s)$ and the terminal value function $\bar{V}^{\text{MPC}}(s)$ within the NLP formulation, and can be stated as

$$V^{\text{MPC}}(s) = \min_z \bar{V}^{\text{MPC}}(x_N) + \sum_{k=0}^{N-1} l^{\text{MPC}}(x_k, u_k) \quad (22a)$$

$$\text{s.t. } x_0 = s, \quad (22b)$$

$$x_{k+1} = f^{\text{MPC}}(x_k, u_k), 0 \leq k < N, \quad (22c)$$

$$0 \leq h^{\text{MPC}}(x_k, u_k), 1 \leq k < N, \quad (22d)$$

$$0 \leq h_N^{\text{MPC}}(x_N), \quad (22e)$$

using the vector of decision variables $z = (x_0, \dots, x_N, u_0, \dots, u_{N-1}) \in \mathbb{R}^{n_z}$. In the above formulation, we introduced the state trajectory (x_0, \dots, x_N) as additional decision variables, which are constrained to start at the given value of the current state (22b), and to follow the system dynamics (22c). This is in contrast to formulation (21), where the state trajectory is considered as an explicit function of the action trajectory. Both approaches are equivalent in terms of the solutions they admit. However, the iterations of numerical optimization algorithms may differ depending on the formulation. The formulation in (21) is referred to as a single shooting or sequential formulation, whereas (22) is a

multiple shooting or simultaneous formulation. For nonlinear unstable systems, the latter is typically preferable.

When deploying MPC (22), the NLP is solved online at every discrete time instant, based on a specified value of the current state s . This yields an optimal trajectory of actions, $(u_0^*, \dots, u_{N-1}^*)$, of which only the first one, u_0^* , is applied to the environment. The resulting new state is then used as the initial state for the next optimization problem, and a new input trajectory is computed. Hence MPC (22) defines a policy.

Similarly to the definition of the value function in (22), we can define the corresponding Q-function by additionally fixing the initial action vector u_0 to the given value,

$$Q^{\text{MPC}}(s, a) = \min_z \sum_{k=0}^{N-1} l^{\text{MPC}}(x_k, u_k) + \bar{V}^{\text{MPC}}(x_N) \quad (23a)$$

$$\text{s.t.} \quad x_0 = s, \quad (23b)$$

$$u_0 = a, \quad (23c)$$

$$x_{k+1} = f^{\text{MPC}}(x_k, u_k), \quad 0 \leq k < N, \quad (23d)$$

$$0 \leq h^{\text{MPC}}(x_k, u_k), \quad 1 \leq k < N, \quad (23e)$$

$$0 \leq h_N^{\text{MPC}}(x_N). \quad (23f)$$

Based on this Q-function and assuming a unique minimizer, the MPC policy is

$$\mu^{\text{MPC}}(s) = \arg \min_a Q^{\text{MPC}}(s, a), \quad (24)$$

where μ instead of π is used to denote that the MPC policy is deterministic.

We now take a closer look at the components of the MPC problem (22), followed by various other relevant considerations.

1) Dynamics model: The central component is the dynamics constraint (22c), in which $f^{\text{MPC}}(x, u)$ is a model of the environment (1). This model can be derived from first principles or identified from data. In the case of nominal MPC, it is deterministic and does not consider the stochasticity of (1).

2) Objective: The stage cost $l^{\text{MPC}}(x, u)$ in the MPC objective (22a) commonly corresponds to the stage cost of the MDP (2). Note that the stage cost may be implicitly time-varying by including an augmented clock state. Thus, it may implicitly include a discounting factor γ . Crucially, the MPC optimization problem needs to be numerically tractable, and the ultimate goal is to provide a policy that achieves a sufficient closed-loop performance when applied to the real environment. Therefore, the stage cost may also be approximated by a function with favorable numerical properties. Often, the stage cost imposes a convex penalty on the deviation of the action and state trajectories from a reference point or trajectory. This is referred to as regulatory MPC or tracking MPC. The focus in regulatory MPC problems is mostly the stabilization of systems. Historically, the ability of MPC to incorporate inequality constraints and to handle multiple inputs simultaneously justified the additional computational complexity. In contrast, economic MPC uses a cost that directly expresses a quantity of interest to optimize, e.g., time, energy use, financial cost, or yield of a production process. Therefore,

economic MPC is closely related to solving MDPs beyond stabilization.

The terminal cost $\bar{V}^{\text{MPC}}(x)$ should ideally capture the cost-to-go at the end of the MPC horizon, cf. (20). The choice of horizon length and terminal cost is often crucial for the performance of the resulting MPC policy. Indeed, a close-to-optimal cost-to-go approximation via the terminal cost allows for shorter prediction horizons. The approximation quality becomes less important as the horizon length grows, and $\bar{V}^{\text{MPC}}(x_N) \equiv 0$ is a widely used choice [43]. However, the NLP (22) becomes computationally more expensive for longer horizons. Thus, the horizon length is typically limited by the available computational resources. In practice, the terminal cost function is often chosen heuristically or based on stability considerations. Still, there is also research on how to explicitly select it as an approximation of the infinite-horizon cost (with respect to the MPC cost), e.g., by simulating forward a pre-selected simple feedback law [44], [45], [46], [47]. When stabilizing a system at a steady state using a locally smooth convex stage cost, a straightforward choice is the infinite horizon linear quadratic regulator (LQR) [48] cost computed for the system resulting from the linearization of the model (22c) at that steady state.

3) Constraints: The final components are the stage and terminal constraints. Stage constraints (22d) can be used to avoid prohibited regions of the state space and to take into account actuator constraints. Terminal constraints (22e) can be used to ensure stability and recursive feasibility of the resulting MPC policy. For the terminal constraints, considerations similar to the terminal cost apply. In principle, they should capture the system's future behavior over the infinite horizon and ensure that the MPC plan will not be too short-sighted regarding the stability and recursive feasibility of the resulting MPC policy.

Constraints can lead to situations in which (22) is infeasible for the current initial state value s , i.e., there exists no value for the decision variables such that all constraints are satisfied. Consequently, the solver would not be able to return a solution, and the evaluation of the MPC policy would fail. Thus, for practical MPC implementations, it can often be helpful not to enforce the constraints strictly but to penalize their violation. For exact penalties with sufficiently high penalty weight, constraint satisfaction is guaranteed if the original problem is feasible [49], [50]. Otherwise, a solution that minimizes the constraint violation is returned.

4) Uncertainty-aware MPC: In contrast to the deterministic model in (22c), stochastic and robust MPC formulations [51], [52], [53] explicitly take into account the uncertainty of the model prediction. The former models the uncertainty as a probability distribution, whereas the latter predicts bounded sets of all possible realizations (respective tractable outer approximations). Both can be separated into scenario [54], [55] and tube [56], [57] approaches. Scenario approaches consider discrete distributions, which may be obtained by sampling from a continuous distribution. This can take the form of sampling several disturbance trajectories separately [58] and planning the corresponding state trajectories in parallel or of constructing

a tree of scenarios that are branched at every time step [59]. Tube approaches predict parameterized approximations of the state distribution respective uncertainty set trajectories. Typical parametrizations are, e.g., normal distributions, ellipsoids [60], [61], [62], [63] or polytopes [64], [65], [66], [67], [68]. This allows for a finite-dimensional representation of the uncertainty, such that a tractable NLP is obtained.

Both scenario and tube approaches can also be classified into open-loop and closed-loop formulations. Open-loop formulations plan only one fixed trajectory of actions. This can quickly lead to unrealistically conservative uncertainty predictions because they do not encode that the noise will be counteracted in the real environment by feedback [69]. Closed-loop predictions consider future feedback, leading to more realistic predictions. Using scenario trees, this can be achieved by planning a distinct action for every tree node. This implicitly corresponds to planning over policies with respect to the discretized disturbance space because the actions depend on past disturbances. Here, nonanticipativity with respect to the causality of the policy should be carefully considered. Closed-loop tube approaches usually consider explicitly parameterized simple feedback laws, e.g., linear feedback, which reacts to state deviations from the tube center. These feedback laws can be precomputed [57], [70] or optimized [71], [72], [68]. While the optimization of state feedback gains is highly nonconvex even for linear systems, the optimization over affine disturbance feedback leads to equivalent convex, but also higher-dimensional, optimization problems [69], [73]. In the context of polyhedral sets, it is, under some assumptions, sufficient to consider only their vertices, which results in tree-structured formulations [74], [75].

5) *Optimization problem classification:* Depending on the mathematical form of the functions in (22), the optimization problem can be classified differently. This is relevant, as it informs both the choice of solution algorithm and the theory regarding the resulting policy. In linear MPC (LMPC), the model is linear, the constraint functions are affine, and the cost functions are convex quadratic, resulting in a quadratic program (QP). LMPC problems can be solved reliably and efficiently. This is often used in contrast to nonlinear model predictive control (NMPC), where typically the model is nonlinear, and the resulting optimization problem is an NLP (22). When the action space is discrete, this corresponds to an additional restriction of the action variables to the space of integers, resulting in a mixed-integer NLP (MINLP) or mixed-integer QP (MIQP) [76]. If the dynamics contain nonsmooth or discontinuous events, such as contact physics, this leads to mathematical programs with complementarity constraints (MPCCs) [77].

6) *Suboptimality and control theory:* As discussed, the MPC problem (22) leverages several forms of approximations of the MDP (4). This opens the question of how the resulting MPC policy behaves with respect to the MDP or when applied to a real system. Since it solves the MDP only approximately, it can be considered a form of suboptimal control. An overview of several sources of suboptimality can

be found in [78]. The suboptimality from the finite horizon approximation is analyzed in [79], [80], [81]. The consequences of approximating the expected value via sampling are addressed in the stochastic programming literature in the context of the sample average approximation [54]. In [82], the authors analyze the suboptimality resulting from affine feedback parametrization in a robust problem formulation. The suboptimality of nominal MPC in a stochastic environment is analyzed in [83], [84], and regret bounds in [85].

The control theory literature often asks a different, though closely related, question, see, e.g., [2], [42]. It investigates under which conditions the MPC policy exhibits desirable behavior. This includes system theoretical properties of the resulting closed-loop system, such as stability [86], [87], [88], and properties like recursive feasibility [89], [90], [91], i.e., the controller should not maneuver itself into a state in which the MPC problem (22) becomes infeasible. MPC should also be able to work under model-plant mismatch and reject disturbances. This is referred to as inherent robustness, and the theory covers nominal MPC, which uses no uncertainty model [92], [93], [94], [95], [96], [97], but can also be extended to stochastic MPC [95], even if it is designed with respect to a wrong disturbance model [98]. Additionally, fast-paced applications may only allow for a suboptimal solution to (22) in the assigned computational budget. Stability [99], [100] and inherent robustness results [101], [102] also exist for this case. Further, while stability results are usually derived for continuous action spaces, they can also be generalized to discrete actuators [51].

B. Numerical Methods for MPC Problems

We distinguish two common and fundamentally different approaches to – possibly approximately – solving finite-horizon optimal control problem (OCP) formulations: Sampling-based methods and methods leveraging derivative-based numerical optimization. In the following, we briefly introduce and discuss both approaches, focusing on algorithms that address the nominal OCP formulation.

1) *Sampling-Based Methods:* As a first class of methods, we consider sampling-based approaches that aim at finding an approximate solution to the stochastic open-loop or nominal OCP by sampling control trajectories and evaluating the associated cost via forward simulation.

The simplest sampling-based method, also known as random shooting and used, e.g., in [103], [58], considers a finite number of independently sampled action sequences and the corresponding state trajectories, which are obtained via forward simulation. The algorithm then chooses the open-loop action trajectory associated with the lowest cost as an approximate solution.

A second, more sophisticated, sampling-based approach is the cross-entropy method (CEM) [104], where the probability distribution generating the open-loop action trajectory samples is iteratively refined based on previously sampled actions yielding low costs. In particular, CEM samples a finite number of action sequences and evaluates their associated costs via

forward simulation. The action sequences yielding the lowest cost trajectories are used to adapt the probability distribution from which new action samples are generated. Typically, a Gaussian distribution or a Gaussian mixture model is used, in which the mean and covariance are adapted at each step. The method has been successfully implemented for MPC in [105], [106].

As a third sampling approach, we consider model predictive path integral control (MPPI), which provides a framework [107], [108], [109], [110] for solving MPC problems via trajectory sampling. In particular, the approach directly addresses the stochastic formulation without resorting to the nominal problem (22). As the method is derived based on a continuous-time model, we refer the interested reader to [110] for an in-depth derivation. Crucially for this survey, the method poses some strong restrictions on the structure of the model and cost: First, the stochasticity of the dynamics needs to enter via the actions, i.e., $f_{\text{MPPI}}(s_k, a_k + w_k)$, where w_k is a random variable in the dimension of the actions. Secondly, the cost J^{MPC} is assumed to be separable in states and actions, as well as quadratic in the actions with a weighting matrix that is inversely proportional to the noise variance. Furthermore, the main theoretical results and optimality (in the limit of infinite samples) only hold if the dynamics are affine in controls and noise [109]. Similar to CEM, MPPI adaptively updates the probability distribution from which action trajectories are sampled. MPPI uses a weighted average to update the mean of this distribution where the weights are based on the associated costs. Since the underlying algorithm of MPPI requires limited implementation efforts, many papers use custom implementations [111]. However, recently an efficient implementation as part of TorchRL [112] and a CUDA-based parallel computation framework was published [113].

Sampling-based approaches naturally allow for stochastic models [114], [106] and can thus directly tackle the stochastic open-loop OCP. Furthermore, sampling-based methods can be applied to problems with highly nonlinear, or even non-smooth, costs or dynamics as long as a simulator is available. While sampling-based optimization methods are typically straightforward to implement and benefit from parallelization, they scale poorly with the dimension of both the planning horizon and the dimension of the action space, i.e., they severely suffer from the curse of dimensionality. Furthermore, state constraints can be addressed only via penalty reformulations or the rejection of infeasible samples. For highly constrained systems, the rejection method further increases the sampling complexity.

2) *Derivative-Based Numerical Optimization:* As a second class of methods, we discuss derivative-based numerical optimization. On this account, approaches derived from a continuous-time OCP formulation, such as, e.g., indirect methods, collocation, and pseudospectral methods, as well as numerical simulation methods, fall into this class, but are outside of the scope of this survey. We refer the interested reader to [115], [116] for a survey on numerical methods starting from a continuous-time formulation and to [2, Chap. 8] for a textbook overview of direct methods.

Assuming that all problem functions in (22) are sufficiently smooth, OCP (22), which is typically referred to as multiple shooting formulation [117], can directly be addressed with standard numerical methods for constrained nonlinear optimization. While multiple shooting formulation (22) keeps both states and actions as optimization variables, one may alternatively eliminate the states from (22) via the equality constraints yielding the single shooting formulation as introduced at the beginning of this survey and given in (21). The single-shooting formulation typically results in dense subproblems with few optimization variables and can thus be efficiently solved by general-purpose nonlinear solvers. In the following, we focus on the multiple shooting problem, given in (22), and tailored numerical methods addressing the particular problem structure arising from this formulation.

First, we distinguish between numerical methods that use first-order derivative information versus approaches that use second-order derivative information. First-order methods are less computationally complex but may require many more iterations. Second-order methods require less but computationally more complex iterations. Due to their low complexity, first-order methods have been considered for solving OCPs in particular in the context of embedded applications [118], [119], [120], [121], [53]. Furthermore, a tailored method for the scenario-based OCP formulations based on alternating direction method of multipliers (ADMM) has been developed in [122].

Second-order methods include nonlinear interior point (IP) methods and sequential quadratic programming (SQP), two widely used classes of methods for numerical optimization. We will briefly discuss both of them in the following. For a more detailed overview of second-order numerical methods for optimal control, we refer to [123].

Nonlinear interior point methods tackle the non-smooth Karush-Kuhn-Tucker (KKT) conditions associated with the constrained nonlinear optimization problem by formulating an approximate but smooth root-finding problem parametrized by a homotopy parameter, which is iteratively lowered towards zero – in the limit recovering the nonsmooth optimality conditions. The intermediate root-finding problems are solved via Newton-type iterations. Nonlinear interior point methods tailored to the OCP structure are discussed in [124], [125], [126], [127].

Within the SQP framework, a sequence of quadratic approximations of the nonlinear OCP is solved. The quadratic subproblems arising in SQP might, in turn, be solved using an interior point method or an active-set solver. SQP methods can typically be warm-started, rendering them particularly attractive for MPC applications where the solutions to subsequent problem instances are expected to be similar. For further implementation details tailored to SQP methods for MPC, such as full and partial condensing, we refer to the survey in [128]. SQP-type solvers tailored to OCP are implemented in [129], [130]. A widely used approximate approach closely tied to SQP is the real time iteration (RTI) [131], [132], [133]. Within the RTI framework, a single iteration of an SQP method is performed, i.e., a single QP approximation to the nonlinear OCP is solved, in order to obtain an approximate solution drastically reducing the computation time per iteration.

Both IP and SQP methods require the computation of (an

approximation of) the Hessian of the Lagrangian associated with the OCP. While the exact Hessian yields locally quadratic convergence to the solution [49], its computation is typically costly. This motivates the use of Hessian approximations, which are cheaper to compute. A common choice for OCP is the Gauss-Newton Hessian [134], [135], which is applicable to nonlinear least squares objectives, and typically very cheap to compute (or even for free, in the case of quadratic objectives). As an additional advantage, the resulting subproblems are convex by construction, unlike the exact Hessian, which may yield nonconvex subproblems. However, due to the neglected curvature, in general only a linear convergence rate is achieved. The core principle of Gauss-Newton can also be extended to problem classes beyond nonlinear least squares, cf. [136] for an overview. Quasi-Newton methods define alternative choices of Hessian approximation. Most prominently, these include the BFGS Hessian, which, with each iteration, converges towards the exact Hessian, yielding a superlinear convergence rate [49].

Independently of the particular choice of Hessian approximation, the subproblems encountered in both IP and SQP methods applied to the multiple shooting formulation (22) exhibit a particular sparsity pattern, which is typically exploited by tailored solvers.

Another second-order method, which can not directly be interpreted within the Newton-type framework, is differential dynamic programming (DDP), originally proposed in [137]. DDP is also based on solving QP subproblems via a Riccati recursion, followed by a nonlinear forward sweep of the system, which employs the linear feedback law returned by the Riccati recursion. The DDP variant using a Gauss-Newton Hessian approximation, which is more commonly referred to as iterative linear quadratic regulator (iLQR), especially within the robotics community [138], has been introduced in [139], [140]. In their standard form, DDP and iLQR cannot directly handle additional constraints, but extensions to OCPs with input bounds have been proposed e.g. in [138], [141]. Sequential linear quadratic programming (SLQ) [142] is often mentioned in the context of DDP, although it can more precisely be classified as an SQP method with a Gauss-Newton Hessian approximation applied to the single shooting OCP (21) for which each of the QP subproblems is solved in a sparsity exploiting manner, i.e., by a Riccati recursion, cf. also [143].

Crucially, all discussed methods will generally converge to a local optimum and thus require a sufficiently good initialization. The local rate of convergence is determined by the accuracy of the Jacobian and Hessian approximation [49].

Only in the particular case of a convex NLP, such as for LMPCs, convergence to a global optimum can be guaranteed. Under these assumptions, the solution map can be shown to be continuous and piecewise affine. This fact is leveraged in explicit MPC where the optimal feedback law is precomputed offline in order to minimize online computation [144]. Explicit MPC is usually limited to small state dimensions, few inequality constraints, or short horizons. Explicit NMPC was investigated e.g. in [145].

In addition to algorithms and software focusing on nominal OCP formulations, numerical methods tailored to tree-structured problems arising in open-loop as well as closed-

loop stochastic formulations have been developed [146], [147], [148], [128], [149]. Numerical methods addressing the tube-based open-loop and closed-loop stochastic OCP formulation are presented in [150], [151], [152], [153] and [154], [72], [155], respectively.

In contrast to the exponential scaling of sampling-based approaches, e.g., shown for linear unstable systems in [156], solving the nonlinear OCP via numerical optimization alleviates the curse of dimensionality, as the computational complexity of the nominal problem typically scales linearly with the horizon and polynomially with the state and control dimension. On the other hand, the sub-problems need to be optimization-friendly and require the availability of derivative information. In comparison, sampling-based methods only require (fast) forward simulation.

V. COMPARISON OF MPC AND RL

Starting from the general problem of solving MDPs, we have shown how RL and MPC are different techniques that aim at deriving optimal policies, cf., Sect. III and Sect. IV, respectively. MPC, for instance, emerged from the problem of solving multivariable constrained control problems, initially with the goal of setpoint stabilization [157]. In contrast, RL methods aim at maximizing closed-loop performance without necessarily relying on a model of the real environment. Their discrepancy is unsurprising since both approaches were developed in parallel communities with different focuses. This is further substantiated in the nearly orthogonal properties of MPC and RL, which were reviewed and emphasized in a case study for a specific linear system in [8]. The more recent adaptions towards economic MPC, e.g. [158] fit the paradigm of solving MDPs and, hence, bring the goals of both communities closer together.

Considering the practicalities of MPC and RL of solving MDPs, further particularities appear, which we compare in the following. The comparison is concluded in Tab. II that shows an overview of relevant properties, similar to [8] and Tab. III, from work that explicitly compares MPC and RL in practical applications. We compare requirements on the state space representation and the properties of the mathematical model, such as smoothness or continuity required by MPC. We refer to [3], [78] for further theoretical comparisons.

In the following, the main practical and conceptual differences between MPC and RL are stated.

1) State Space: Real environments can only be approximately described by a state, which is usually not always directly measurable, leading to the concept of partially observable MDPs (POMDPs). So far, we omitted a discussion about POMDPs and only mentioned some points required for a high-level discussion on the state space. Considering an environment where the state s is unknown and only observations $O_k \in \mathbb{R}^{n_o}$ are made at step k , the most general concept of a state space would involve the collection of all observations O_0, O_1, \dots and applied actions A_0, A_1, \dots . Even this very general concept may not allow to fully describe the real environment due to partial observability.

TABLE II: Comparison of practical properties between MPC and RL. The evaluation is simplified and conceptual. Exemptions may exist.

Property	MPC	RL
state-space	model specific ✗	(quite) arbitrary ✓
model requirements	differentiability/ online simulation ✗	offline simulation ✓
uncertainty	guarantees with known uncertainty ✓	probabilistic guarantees ✓
stability	strong theory ✓	minor theory ✗
constraint handling	inherent ✓	hard to achieve ✗
online computation time	high ✓	low ✗
offline computation time	low ✗	high ✓
adaptability	inherent ✓	needs retraining ✗
generalization	inherent ✓	poor when out-of-distribution ✗

However, often these observations, or even the M most recent observations $\mathcal{O}_M = (O_{k-M}, A_{k-M}, \dots, A_{k-1}, O_k)$ at step k are sufficient to estimate the relevant states of an environment [159].

In MPC and RL algorithms, the state space is interpreted conceptually differently. In derivative-based MPC, the state space is usually constructed by relating it to an optimization-friendly model, usually having a physical interpretation following differential equations. Yet, alternative approaches also consider a sequence of measurements as state [160], cf. Sect. IV-B. States used within derivative-based MPC do not necessarily correspond to the measured sensor outputs and, thus, are often estimated by a state observer that converts a series of observations \mathcal{O}_M into a state estimate.

RL methods and sampling-based MPC methods may use states based on a corresponding physics-based model and the related estimators [161], [162], [163] or a state observer. However, the state may also be kept as a recent history of raw sensor data, which could be based on a variety of different input modalities, such as images or text. Often, end-to-end learning is used, where the raw sensor input, such as images, is fed directly to NNs. The NN outputs subsequently the controls, e.g., [164]. It is common to make the learned dynamics model or policy dependent on a window of the most recent history of observations and actions \mathcal{O}_M , cf., [21], which is particularly useful for POMDPs. While an NN architecture does not explicitly model states, the stacking of layers, the related transformations, the exploitation of equi/invariances, and the condensing of information can be interpreted as modeling of hidden states, cf. [165]. Subsequent layers can be interpreted as the policy based on these hidden states. Remarkably, this structure is not enforced explicitly. Also, recent approaches of the more classical observer/controller architecture propose to tune MPC and the state estimator together [166].

2) *Model and Application:* Highly related to the state space are the characteristics of a potentially used model. As explained in Sect. IV, MPC requires a model to simulate the dynamics. Whereas sampling-based MPC only requires the forward simulation of a model, derivative-based MPC involves the computation of gradients through the model. Therefore, derivative-based MPC requires an optimization-friendly model

with stark limitations to its structure since the model is part of the optimization problem. The MPC optimization problem becomes particularly challenging if the model is non-smooth [77], stochastic, or contains integer variables [161].

A physically motivated prediction model, which is often used in MPC, has the advantage of better understanding and explaining the environment behavior, often referred to as explainability. The possibility of predicting interpretable model states allows for the straightforward definition of constraints. For example, a certain velocity must not be exceeded in a vehicle control problem. This constraint can be readily formulated by a model that predicts the system's velocity accurately. Besides physically motivated models, more general models such as NNs or nonparametric models such as Gaussian processes [167] can be used. These models still have the advantage of predicting the environment. However, the explainability of the states can be lost.

Not requiring the explicit modeling of the environment is a major claim of RL. However, that statement needs some additional framing. In fact, many successful model-free RL applications use models, at least for training the policy [168]. One fundamental difference between models used for RL or MPC is that RL models are often used for offline simulation but not during deployment of the final policy. This allows an abundance of complex computations involved in the simulation, which could comprise logical statements or complex high-fidelity models. Even though most RL methods still require simulation models, real-world RL is an active research field [169] making significant progress [170].

The low inference time of NNs makes it appealing to increase the sampling frequency of RL algorithms for faster feedback loops. However, a too-small discretization time step in RL can result in the function approximation error exceeding the value difference of actions, rendering deep RL methods that rely on function approximation useless [171]. Standard offline system identification techniques used to identify the MPC model also involve problems with too high sampling frequencies, e.g., a decreased signal-to-noise ratio and an ill-conditioned model [172]. However, in derivative-based MPC, the choice of the sampling frequency is often limited by the solution time of the underlying optimization problem.

3) *Intrinsic stochasticity:* As introduced in (1), the environment is typically assumed to be intrinsically stochastic. Thus, even if the environment is perfectly known, it is not possible to precisely predict future state trajectories. Both robust and stochastic MPC explicitly take into account this uncertainty, which is in this context typically referred to as noise or disturbance. Robust MPC is, in a sense, agnostic to the question of whether the uncertainty is intrinsic or due to a systematic modeling error: it considers all trajectories possible under the given assumptions. For stochastic MPC, on the other hand, it is important to be aware of whether the prediction errors correlate over time since this affects the predicted state distribution. Intrinsically stochastic noise is often assumed to be independent, i.e., noncorrelating.

As stochasticity is inherently part of the MDP framework, RL algorithms naturally consider stochastic environments. The

policies can be trained directly on the real-world environment to account for the real-world uncertainty, cf., Sect. III. A form of stochasticity particularly challenging for RL algorithms are rare events, i.e., large but rare deviations from the average obtained cost. Learning value estimates with rare events is particularly difficult [173].

4) Model mismatch: In practice, the environment for which a policy is designed or trained will often differ from the one it is deployed on. Thus, it needs to be ensured that the policy will still perform well during deployment. Standard RL algorithms can be biased towards the specific model used during training. Different strategies like domain randomization, robust RL [174], [175], and meta RL tackle model uncertainty. In all of these approaches, the agent is trained not only on a single model but also on a potentially adaptive distribution of models, which can improve the generalization to new models [176]. The underlying assumption is that the real-world environment lies in the distribution of the training models [176]. Optimization-based meta RL [176] learns weights that can quickly adapt to new models. In contrast, in-context meta RL uses history-dependent policies to infer the current dynamics of the environment [177].

Like in the case of intrinsic stochasticity, both stochastic and robust MPC can explicitly take model mismatch into account. For stochastic MPC, it is important to consider that predictive errors due to model uncertainty are typically strongly correlated across time. Note that the model used in a stochastic MPC formulation is typically a simplification or approximation of the environment, leading to a mismatch of the stochastic MPC model with respect to the environment. The field of distributionally robust MPC aims to robustify against mismatches in the distribution model [178]. Even without considering the model mismatch explicitly, MPC can perform remarkably well. This is the property of inherent robustness of MPC, as explained in more detail in Sect. IV.

5) Stability: Stability theory is usually not a major concern of RL algorithms since the main objective is closed-loop performance and practical stability instead. However, as denoted in Sect. IV, asymptotic stability and constraint satisfaction are, or were at least historically, the main focus of MPC algorithms and applications. A widely used tool for analyzing the stability of control problems involves the construction of Lyapunov functions. If a Lyapunov function exists for a controlled deterministic MDP, it is said to be asymptotically stable, i.e., trajectories converge. If the MDP is stochastic, the concept of input-to-state-stability can be applied, which requires the trajectories of the controlled system to converge to a region around the origin whose magnitude depends on the maximum norm of the noise [2].

Input-to-state stability and asymptotic stability may not be applied to general MDPs since they require certain properties of the cost function, such as a feasible origin. Therefore, the authors in [179] propose a stability concept named D-stability that applies to MDPs and generalizes the dissipativity theory of economic MPC.

6) Constraints: The ability to account for constraint satisfaction and stability are some of the main reasons MPC is outstanding compared to other control techniques. In recent years, the RL community also has increasingly focused on providing safety guarantees [9]. A widely adopted framework in RL involves modeling constraints using constrained MDPs [180]. Following [9], these include: Hard constraints, the constraints are always fulfilled, chance constraints, the constraints are fulfilled with high probability, and constraint violations transformed as accumulated costs. The latter can be either formulated such that the accumulated costs can not exceed a safety budget or as a penalty integrated into the task objective [9]. The choice of formulation dictates the strategies employed to address these constraints. Common approaches include deriving safe action sets [181], optimizing a Lagrange formulation with dual gradient descent [182], using trust-region optimization [183] or control-barrier functions [184]. Additionally, using MPC [185] to provide safety guarantees related to a nominal model was proposed. Further work suggests augmenting the RL state with Lagrange multipliers [186] or with the currently used safety budget [187]. Still, a common problem is that safe RL policies become either too conservative or, otherwise, may violate constraints [188]. Thus, combining MPC and RL is highly desirable for constraint satisfaction.

7) Online Computation Time: A major concern in embedded applications is the maximum online computation time of an algorithm on embedded hardware. The maximum inference time of many NN architectures, which are often explicit functions, can usually be tightly bounded. However, the computation time of optimization solvers for NMPC problems is often unbounded. In fact, it cannot be guaranteed in general that a meaningful solution, i.e., at least a feasible solution, is returned by the optimization algorithm outside of particular optimization problem classes such as convex QPs [189], [190]. For NMPC, the primal and, possibly, dual variable initialization of the optimization algorithm is essential for fast online computations. If the variables are sufficiently close to the optimal solution, the local convergence rate is fast. In fact, it can be quadratic or superlinear, depending on different numerical algorithms, cf. Sect. IV.

8) Offline Computation, Engineering, and Maintenance: In contrast to their fast inference time, RL algorithms usually require a tremendous amount of training samples, and therefore, training time, even for small-sized MDPs. The training may be performed without human intervention. Yet in practice, fine-tuning on the RL hyperparameters may be required [191]. Besides the initial system identification, standard MPC does not require further training time. System identification differs from RL training by the target of predicting relevant outputs and possibly states of a system. In contrast, the RL target and the potential internal model focus on closed-loop performance. In the RL approach, the possible indirect internal model of the environment is learned only for the particular task described by the MDP. The classical system identification may also use models based on physical models, which makes it possible to

adapt the controller to changing environments or requirements. RL may require a whole new training data set for changes in the cost, model changes, or changes in the distribution of the states-space, e.g., an adjusted operating range in the environment.

9) Generalization: To some extent, RL policies can generalize out of their training distribution but with hardly any predictable behavior and guarantees on the closed-loop performance [192]. Thus, practitioners often need to ensure that the support of the training distribution covers the state and transition distribution encountered during deployment [193]. An alternative is to further train the RL policy during deployment [194]. If the model used within MPC can approximate the real environment well on the whole state space, MPC generalizes well, and safety and performance guarantees can be found. Arguably, the model may generalize well based on available knowledge, starting from first principles and physical or mathematical insight. The knowledge about the model results in a more interpretable generalization and is among the main motivations for MPC in general, learning-based MPC or model-based RL. It may not be possible to evaluate whether a physical model approximates the real environment well on the full state space of the environment. However, the physical explanation may also provide insights into its limitations.

10) Performance in Practice: In practice, the expected performance difference between MPC and RL depends on the environment's specific characteristics, computational resource availability, and the model or data quality. Both approaches have their strengths and weaknesses, depending on the particular requirements and constraints of the control problem. Several works compare both approaches on specific applications; see Tab. III. The problems differ vastly, from drone racing [164] to multi-energy environments that involve integer variables [161]. The MPC formulations vary depending on the applications. For high sampling times, fast solvers such as acados [130] are used. Instead, for combinatorial problems, computationally highly demanding mixed integer solvers, such as Gurobi [195], are required. Most of the authors use the same state space for MPC and RL, despite the RL's ability to cope with arbitrary information inputs, which is exploited only in [164], [196]. For a known model, most authors report superiority of MPC w.r.t. the closed-loop performance, with less than 4% cost reduction in [197], [163] and 16% in [198] when compared to RL. In [162], the authors report that MPC outperforms RL on a flexible robot manipulation task with a rather high-dimensional state space. RL was claimed to be superior to MPC for environments with poor models, e.g., in [199], [161], yet robust control techniques are not implemented. The authors in [164] claimed superiority of a particular choice of RL algorithms against MPC in real-world experiments (RWE) of racing drones.

VI. COMBINATION APPROACHES

As pointed out in the previous sections, RL is a collection of algorithms to learn an optimal policy for MDPs by

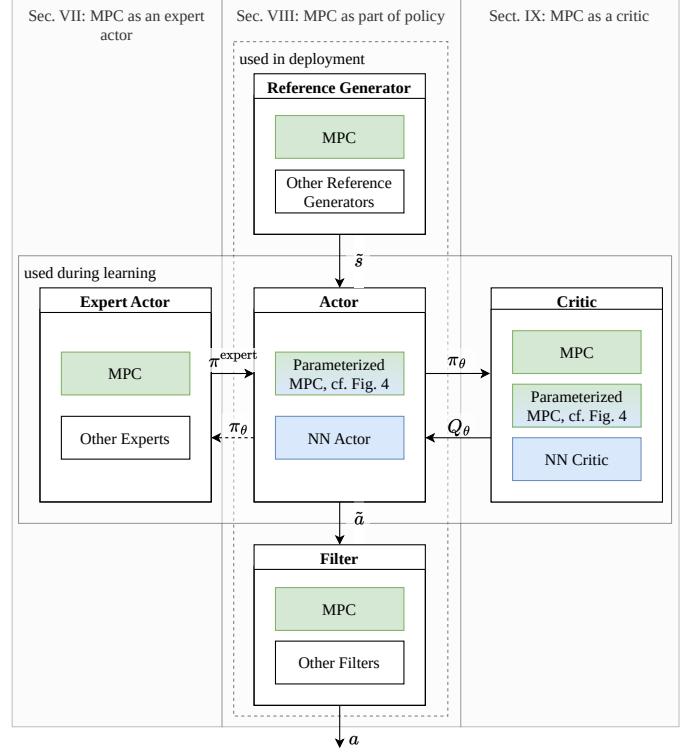


Fig. 2: Modular view on combinations of MPC and RL. The combinations are aligned with the sections of this survey. The horizontal separation corresponds to using MPC as part of the expert actor, the deployed policy, and the RL critic. This overview highlights the possibility of using several instances of MPC with different roles in various parts of an RL algorithm and a deployed policy. MPC is used with fixed expert parameters within the expert actor, as a reference generator, as a postprocessing filter, or, possibly, in the critic. Learned MPC, i.e., an MPC structure involving learned parameters, can be used within the learned actor or possibly within the learned critic.

interacting with the environments. MPC, in contrast, refers to a mathematical program that implicitly approximates the MDP. The two approaches exhibit different advantages. Therefore, a combination of both is appealing.

We propose a categorization of combination approaches that distinguishes in which algorithmic part of RL the MPC framework is used. The related general RL building blocks are the actor, the critic, and possibly an expert actor, c.f., Fig. 2. Accordingly, we propose the categories

- 1) MPC as an expert actor, cf. Sect. VII,
- 2) MPC within the deployed policy, cf. Sect. VIII,
- 3) MPC as part of the critic, cf. Sect. IX.

In the following, we shortly outline the different categories.

MPC as an expert actor. An MPC with fixed parameters and the desired behavior can be used as an expert to learn policies. The expert behavior can either be used by an IL algorithm that tries to purely mimic the behavior or to guide the exploration process during RL training [217]. MPC used as an expert is elaborated in Sect. VII.

MPC within the deployed policy. Another common variant of how MPC and RL are combined is using MPC within the deployed policy. A parameterized MPC can be used as part of the learned actor, as, for instance, proposed by [179]. Closely related are concepts that use MPC as a posterior filter

TABLE III: Literature that compares MPC with RL on practical applications. The table lists whether real-world experiments (RWE) were performed and which RL and MPC algorithms were used for the experiments.

Application-oriented comparison between MPC and RL						
Ref.	Authors	Year	Application	RWE	MPC Formulation (Algorithm, Solver)	RL Algorithm
[197]	Ernst et al.	2009	power system	No	NMPC (IP, N/A)	fitted Q iteration [200]
[161]	Ceusters et al.	2021	multi-energy system	No	MILP-MPC (cplex [201])	TD3, PPO
[199]	Hasankhani et al.	2021	ocean current turbine	Yes	N/A	DQN
[202]	Lin et al.	2021	cruise control	No	NMPC (IP, IPOPT [203])	DDPG
[163]	Brandi et al.	2022	thermal energy system	Yes	MILP (N/A) GP-MPC [167], Bilevel DeePC [205], (custom)	SAC
[204]	Di Natale et al.	2022	building temperature control	Yes	Bilevel DeePC [205], (custom)	TD3
[206]	Dobriborsci et al.	2022	mobile robot	Yes	NMPC (SLSQP [207])	DQN
[208]	Byravan et al.	2022	MuJoCo locomotion	No	Sequential Monte Carlo [209], CEM	MPO [210]
[198]	Wang et al.	2023	building temperature control	No	NMPC (IP, IPOPT [203])	SAC, DDPG, DDQN
[211]	Song et al.	2023	drone racing	Yes	NMPC (SQP, acados[130])	PPO
[212]	Shi et al.	2023	vehicle parking	No	NMPC (N/A)	PPO
[213]	Imran et al.	2023	traffic control	No	MIQP-MPC (Gurobi [195])	DQN
[214]	Reiter et al.	2023	autonomous racing	No	NMPC (SQP, acados[130])	SAC
[215]	Mercego et al.	2023	greenhouse climate control	No	NMPC (IP, IPOPT [203])	DDPG
[162]	Mamedov et al.	2024	flexible robot manipulation	No	NMPC (SQP, acados[130])	PPO, SAC
[216]	Hoffmann et al.	2024	vehicle lane change	No	N/A LMPC (N/A), NMPC (IP, IPOPT [203])	PPO, Q-learning
[196]	Oh	2024	chemical and biological processes	No	N/A LMPC (N/A), NMPC (IP, IPOPT [203])	SAC, DDPG, TD3

or reference provider as an element of the deployed policy but are not used during learning. For instance, the authors in [185] use MPC as a safety filter and [218] use MPC as a reference provider, cf. Fig. 2. All concepts that use MPC within the deployed control policy are elaborated in Sect. VIII.

MPC as part of the critic. An MPC, possibly parameterized with learnable parameters, can further be used to evaluate the value function at a given state, i.e., the critic uses an MPC variant. For instance, the in [219] uses an MPC with fixed parameters to compute the value at a given state. This structure is reviewed in Sect. IX.

Sections VII to IX describe combination approaches of MPC and RL conceptually, leaving out a detailed theoretical discussion. Nonetheless, the additional theory Sect. X highlights some important findings and relevant literature when combining the two approaches.

To the best of the authors' knowledge, in the following review, the most important works that combine MPC and RL can be classified according to the proposed categories in Fig. 2. The works are further compared related to their application, whether RWEs on embedded hardware was performed, which MPC type and solver the authors used, and which RL algorithm the authors applied. While the MPC algorithm classification is more distinct, the classification of the RL algorithm can be ambiguous and intertwined with the overall presented algorithm. Besides widespread RL algorithms such as SAC

and PPO, several forms of DP, such as policy iteration (PI) or VI, forms of IL such as behavior cloning (BC), and even supervised learning for learning value functions, such as maximum likelihood estimation (MLE), are denoted as RL algorithm in the tables. Applications include unmanned ground vehicle (UGV), temperature control, energy systems, chemical processes, traffic management, autonomous racing with vehicles or drones, and automated driving (AD).

A. Architectures of Parameterized MPC

Before diving into the particular approaches of how MPC is used as part of RL or IL algorithms, a short classification of parameterized MPC architectures that use NNs is given. This distinction is particularly important for designing combination approaches as it centrally governs the desired properties and choices of algorithms.

We consider an MPC optimization layer that can be parameterized by a parameter $\phi \in \mathbb{R}^{n_\phi}$ and may depend on the decision variables z of the MPC optimization problem and the current state s through a highly nonlinear function approximator (FA) $\phi = \varphi_\theta(s, z)$, e.g., an artificial neural network (NN) with the (very) high dimensional parameter vector $\theta \in \mathbb{R}^{n_\theta}$. Since NNs are the most common FAs, we consecutively mainly write NN but also comprise other forms for FAs. When fixing the parameters ϕ , the MPC optimization problem is assumed to contain only optimization-friendly objective, model, and constraint functions.

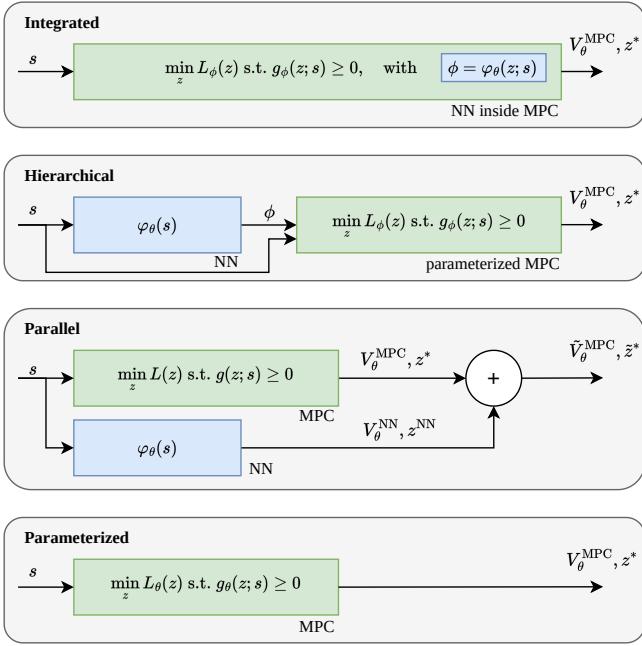


Fig. 3: Parameterized MPC architectures: Proposed architectures of actors (or potentially critics) used in RL utilizing MPCs and NNs/simple parameters. In the integrated architecture, the NN is part of the optimization layer and depends on the decision variables. Suppose an NN can be evaluated separately from the optimization routine but provides the parameters to an MPC optimization layer. In that case, the architecture is referred to as hierarchical. If the MPC problem is solved in parallel to the NN, we refer to a parallel architecture. If the learned parameters do not depend on the current state s , the architecture is referred to as parameterized.

A general form of the parameterized MPC, based on (22), can be written as

$$\min_z \quad L_\phi(z) \quad (25a)$$

$$\text{s.t.} \quad x_0 = s, \quad (25b)$$

$$x_{k+1} = f_\phi^{\text{MPC}}(x_k, u_k), \quad 0 \leq k < N, \quad (25c)$$

$$0 \leq h_\phi^{\text{MPC}}(x_k, u_k), \quad 0 \leq k < N, \quad (25d)$$

$$0 \leq \tilde{h}_\phi^{\text{MPC}}(x_N), \quad (25e)$$

and the parameterized objective is

$$L_\phi(z) := \bar{V}_\phi^{\text{MPC}}(x_N) + \sum_{k=0}^{N-1} l_\phi^{\text{MPC}}(x_k, u_k).$$

For easing the notation, all constraints of (25) are summarized by $g_\phi(z; s) \geq 0$. Thus, the optimization problem (25) can be concisely written as

$$\min_z L_\phi(z) \quad \text{s.t.} \quad g_\phi(z; s) \geq 0. \quad (26)$$

Note that we use the notation $g_\phi(z; s)$ to indicate that z are decision variables of the optimization problem and s are parameters. The following architectures are proposed within this context, starting with the most general one, i.e., the integrated architecture, cf. Fig 3.

1) *Integrated Architecture:* In the integrated setting, the parameters depend on both the state s and the optimization variables z via the function $\varphi_\theta(s, z)$, as, for instance, within

the algorithm proposed by [220]. Therefore, the NN is part of the optimization problem and can not be evaluated separately in the inference path, leading to a highly nonlinear and, possibly, nonconvex optimization problem, which may be computationally challenging. Particularly, when using derivative-based solvers, this involves differentiating the highly nonlinear parameterized function $\varphi_\theta(z; s)$ w.r.t. the decision variables z by $\nabla_z \varphi_\theta(z; s)$ as part of solving the MPC optimization problem. Note that the optimization layer may provide any of the optimal decision variables z^* or the MPC objective function V_θ^{MPC} of the MPC optimization problem.

2) *Hierarchical Architecture:* In the hierarchical architecture, the neural network provides an input to the MPC. Therefore, the parameter ϕ depends on the current states s via the highly nonlinear function $\phi = \varphi_\theta(s)$. For example, a reference trajectory parameterized by ϕ could be provided in case this architecture is used as a policy, such as in [221], [214], or ϕ could parameterize constraints in the OCP, similar to [222]. Notably, the function $\varphi_\theta(s)$ can be evaluated before solving the optimization problem. In this case, the potential nonlinearity of $\varphi_\theta(s)$ does not influence the numerical optimization problem structure, i.e., the optimization problem does not get harder to solve, despite the changing parameters.

3) *Parallel Architecture:* In the parallel architecture, the MPC problem is usually not parameterized. However, an NN is evaluated in parallel to the MPC optimization problem, and its output is used to correct the optimal solution for decision variables or the value function of the MPC. This architecture holds the advantage of not requiring differentiating the optimization problem and the disadvantage of potentially unsafe actions due to the perturbation of the MPC actions. An example of this architecture used to approximate the value function can be found in [223].

4) *Parameterized Architecture:* In the parameterized MPC architecture, the parameters ϕ are constant and independent of decision variables z or the state s . Conceptually, a parameterized optimization-friendly MPC problem is formulated that does not require the evaluation of highly nonlinear functions to obtain the parameter ϕ . For instance, such a parameterization is proposed in [224], [225] and [226].

Above, we introduced a number of architectures for parameterizing MPC with potentially highly nonlinear functions such as NNs. These architectures may be used in the following MPC and RL combinations, whereby some parameterized MPC architectures are more or less favored in particular MPC and RL combinations.

VII. MPC AS AN EXPERT ACTOR

For some problems, it is possible to design an expert MPC that achieves good closed-loop performance. A thorough collection of relevant literature can be found in Tab. IV and a sketch in Fig. 4. There are multiple ways to exploit such an expert motivating the structure of the following section:

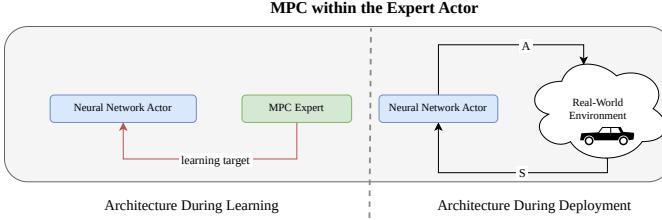


Fig. 4: Combinations: MPC as an expert actor. The plot is split into the learning and deployment phases. Blue boxes indicate Neural Networks (NNs), and green boxes are used for MPCs.

- A. **Imitation learning from MPC**, Sect. VII-A. Here, the goal is to replace the expert MPC with a NN to achieve faster computation time using imitation learning.
- B. **Guided Policy Search using MPC**, Sect. VII-B. In this paradigm, the expert MPC policy improves the exploration process during the RL training guiding the learned policy to low-cost regions.

Besides the explored application, the table specifies whether the algorithm is related to imitation learning (IL) or guided policy search (GPS) and which particular MPC, RL, and IL algorithms are used as a basis.

A. Imitation Learning from MPC

In a problem setting where MPC achieves a sufficient closed-loop performance burdened only by its online computation time, a trained NN may replace the MPC. The typically fast inference of the NN may drastically decrease the online computation time by omitting the time-consuming online optimization. In fact, in many real-world applications [162], [237], [227], the computation time of the MPC solver is a significant limitation and potentially even unbounded, particularly for more complex structures, e.g., involving nonlinear systems, stochasticity or discrete decisions, c.f. Sect. IV-B. Two research directions try to alleviate the problem of large online computation times.

First, for linear systems, linear constraints, and convex quadratic costs, the optimal feedback control law is piecewise linear within a polytope of the state space [245]. Within explicit MPC [246], [145], these polytopes and their control laws are computed offline and switched during deployment by determining the active region. For large state spaces or a large number of constraints, explicit MPC becomes quickly intractable. Thus, approximate explicit MPC only approximately stores the control law of the MPC policy [247].

A learning-based approach to approximate explicit MPC is using IL to learn the MPC expert. Methods can learn the whole trajectory predictions [248] or learn the first applied action [249]. Early works [250], [70] and several follow-up works [249], [251], [252], [253], [254], [235] are aiming to imitate a complex MPC with a neural network, by training the NN with supervised learning methods. Further aspects of the learned NN policy were analyzed regarding safety [255], [162], stability [256] and robustness [249]. Another important aspect is the verification of the learned NN, where the object of interest is the worst-case approximation error to the MPC expert. This can be done in a probabilistic fashion using Hoeffding's inequality, such as in [249], [257], where

confidence bounds are derived for an estimate of how often an approximation error threshold might be exceeded. Alternatively, verification techniques like in [256] solve an MIQP to bound the worst-case approximation error. For a general overview of NN verification, see [258].

Often, a surrogate loss function as in standard BC minimizes the squared distances between the predicted action of the NN and the action of the MPC expert. However, it disregards the costs and structure of the underlying OCP while learning the policy. Thus, a natural replacement is to use instead the Q-function of the MPC, cf. (23) as a learning objective [259], [260], [219]. Since the MPC provides a Q-value function in addition to the expert policy, cf. Fig. 2, we categorize such methods in a class named MPC as a critic, as further discussed in Sect. IX.

B. Guided Policy Search using MPC

In the previously mentioned IL methods, the MPC expert does not consider the progress of the learned policy during training. In the guided policy search (GPS) regime, the expert gradually guides the learned policy to better trajectories [217], [228], [230], [231]. This is ensured by coupling the trajectory optimization and policy learning by requiring that the MPC expert does not deviate too far from the current predictions of the learned policy [230]. The final learned policies can generalize even when the MPC expert fails to find a solution [231]. Further, the authors in [230] have shown superior performance over policies learned from a fixed dataset of expert trajectories.

Another possibility to guide the exploration process for RL was proposed in [244] and is related to the options framework [261]. A learned high-level exploration policy decides between using an MPC expert policy or the currently learned low-level RL policy. The usage of the MPC expert is regularized over time to enforce that the learned low-level RL policy works as a stand-alone policy during deployment. A benefit over GPS is that the high-level policy can avoid using the MPC expert for state regions, where the MPC expert guidance is suboptimal.

VIII. MPC WITHIN THE DEPLOYED POLICY

This section discusses methods that use MPC during deployment in the real environment. A primary distinction is drawn on whether parameterized MPC is trained by RL algorithms or if MPC is used for pre/postprocessing after the RL training. When learning a parameterized MPC, we furthermore distinguish between approaches that aim to align the MPC formulation with the MDP structure, e.g., to learn an internal MPC model that aims to approximate the real environment as closely as possible or methods that primarily focus on closed-loop optimality. Accordingly, we structure this section as follows

- A. **Aligned-learning**, Sect. VIII-A. In the paradigm of aligned learning, the MPC structure (21) approximates the real-world MDP structure (2), i.e., it uses a transition model, costs, constraints and a terminal value function that individually approximate the MDP and the optimal value function V^* , respectively.

TABLE IV: Literature that uses MPC as an expert actor for RL. The table lists the applications and whether real-world experiments (RWE) were performed. Additionally, the IL/RL and MPC algorithms are stated. note that RL algorithms can be used to perform IL, e.g., [227]. The table further differentiates whether the expert adapts to the approximation error of the learned policy via guided policy search (GPS)

MPC as an Expert Actor							
Ref.	Authors	Year	Application	RWE	MPC Formulation (Algorithm, Solver)	IL/RL Algorithm	GPS
[217]	Levine et al.	2013	MuJoCo locomotion	no	NMPC (custom iLQR)	Off-policy PG	yes
[228]	Levine et al.	2013	MuJoCo locomotion	no	NMPC (custom iLQR)	Var. Policy Search [229]	yes
[230]	Mordatch et al.	2014	MuJoCo locomotion	no	NMPC (custom iLQR)	BC/ADMM	yes
[231]	Levine et al.	2016	PR2 robot arm robot manipulation	yes	NMPC (custom iLQR)	BC/Bregman ADMM [232]	yes
[233]	Sun et al.	2018	autonomous driving	no	NMPC (IP, IPOPT [203])	DAgger [193]	no
[58]	Nagabandi et al.	2018	MuJoCo locomotion	no	NMPC, random sampling (custom)	DAgger [193], TRPO	yes
[234]	Wang et al.	2019	MuJoCo locomotion	no	CEM	BC, IL	no
[235]	Pinneri et al.	2021	MuJoCo locomotion	no	CEM	DAgger [193]	no
[236]	Sacks et al.	2022	robot arm	no	MPPI (custom)	DAgger [193]	no
[237]	Dawood et al.	2023	Kuboki Turtlebot 2	yes	NMPC (SQP, acados [130])	SAC, TD3	no
[227]	Kang et al.	2023	Unitree Go1 and Aliengo (quadruped)	yes	NMPC (N/A)	PPO	no
[238]	Ahn et al.	2023	random linear system	no	LMPC (N/A)	DAgger [193]	no
[239]	Le Lidec et al.	2023	robot arm	no	NMPC (FDDP [240])	BC/ADMM	yes
[162]	Mamedov et al.	2024	flexible robot arm	no	NMPC (SQP, acados [130])	BC, DAgger [193], AIRL [241], GAIL [242]	no
[243]	Hoffmann et al.	2024	autonomous driving	no	LMPC (SQP, acados [130])	custom IL	no
[244]	Schulz et al.	2024	cart pole, spacecraft	no	NMPC (SQP, acados [130])	TD3	yes

- B. **Closed-loop learning**, Sect. VIII-B. In the paradigm of closed-loop learning, an MPC is used as an optimization layer within the actor policy and trained within an RL algorithm for closed-loop optimality, which, in general, does not require the individual parts of the MPC to be aligned with the MDP. For instance, the MPC model should not necessarily fit the real system expected or most likely transition to provide closed-loop optimality [262].
- C. **MPC for pre/postprocessing**, Sect. VIII-C. Using MPC for pre/postprocessing does not involve MPC during learning but as part of the deployed policy. This section is further split into reference generation and filtering.

Different structures that include MPC in the deployed policy are proposed; see Fig. 2 and Fig. 5.

A. Aligned Learning

The classical design procedure of MPC focuses in its first step on finding an usually deterministic mathematical model $f_\theta^{\text{MPC}}(x, u)$ assumed to be parameterized by θ that describes the environment sufficiently well [172]. Secondly, a horizon length N and a terminal value function $\bar{V}_\theta^{\text{MPC}}(s)$ as in (21) with parameters θ are obtained to approximate the optimization problem over a finite horizon.

The advantage of aligning the MPC with the MDP can be seen in the well-interpretable formulation, adaptability to new problems, generalization, potential guarantees for stability and recursive feasibility, and the division of the overall design into subtasks. However, the online computation time may high if

complex MPC formulations are used, e.g., stochastic MPC formulations that account for uncertainty.

Obtaining the terminal value function to approximate the infinite horizon value function as in (21) is computationally challenging and, therefore, usually approximated and often obtained through learning algorithms closely related to the RL framework. For simple MDPs, e.g., deterministic MDPs with linear models and a quadratic cost, the terminal value function can be computed exactly and does not require learning or approximation. The costs $l^{\text{MPC}}(x, u)$ are usually given by the application, i.e., the MDP, but may be altered to improve the numerical properties related to the optimization problem regarding smoothness and convexity. We denote parameters of both the model $f_\theta^{\text{MPC}}(x, u)$ and the terminal value function $\bar{V}_\theta^{\text{MPC}}(s)$ by θ .

Literature that can be categorized as MDP aligned typically uses the integrated or parameterized architecture according to Fig. 3. For instance, the paramters θ could parameterize a NN that is used as the internal MPC model [58] or the terminal value function [220], corresponding to the integrated architecture. Tab. V summarizes literature that can be related to the MDP aligned learning paradigm. In the comparison of Tab. V, a distinction is made, whether the model or the terminal value function was learned. Additionally, the table indicates if the corresponding object was learned during the deployment of the continually improving MPC policy, referred to as on-policy learning (ON-P), or learned by using a separate off-policy sampling strategy (OFF-P). An example of an off-policy sampling strategy would involve system identification

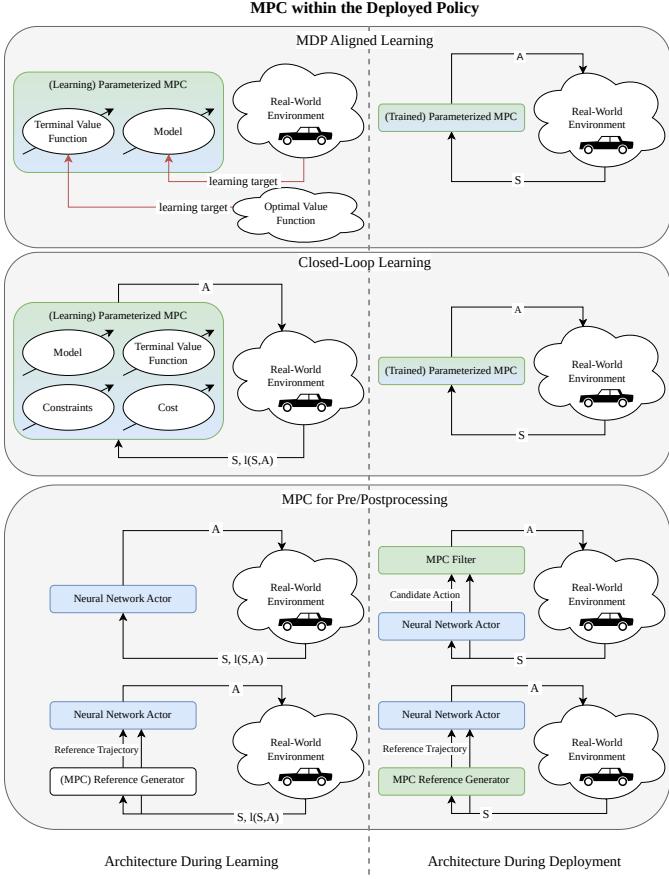


Fig. 5: Combinations: MPC within the deployed policy. The plot is split into the learning and deployment phase. Blue boxes indicate Neural Networks (NNs), green boxes are used for MPCs, and blue/green boxes refer to parameterized MPCs that involve parameters/NNs that are learned during the learning phase, see Sect. VI-A.

to identify a model before deploying the MPC.

In the following, the two objectives of learning the model or the terminal value function are explained in further detail.

1) Learning the Model: The first target of aligned learning is to approximate the stochastic model of the real environment P with a model f_θ^{MPC} that is used within MPC. The model is usually a parameterized mathematical model designed via first principles or a more generic model using a function approximator like a NN or Gaussian process (GP). Parameters θ of candidate models are fit to observed transition data $\mathcal{D}^{\text{id}} = \{(S_0, A_0, S_1), \dots, (S_{M-1}, A_{M-1}, S_M)\}$ of state transitions and related actions. The evaluation of the model fit requires a validation or loss function $\mathcal{L}^{\text{id}}(\cdot)$, which could be, for instance, the least-square loss function. The parameters can be found by minimizing

$$\min_{\theta} \mathbb{E}_{(S, A, S^+) \sim \mathcal{D}^{\text{id}}} \left[\mathcal{L}^{\text{id}}(S^+ - f_\theta^{\text{MPC}}(S, A)) \right]. \quad (27)$$

2) Learning the Terminal Value Function: In order to approximate the MDP on a finite MPC horizon as in (21), a terminal value function $\bar{V}_\theta^{\text{MPC}}(s)$ with parameters θ needs to be established. From a system theoretical perspective, the terminal value function is important in terms of stability or recursive feasibility, c.f., Sect. IV. However, when focusing on

closed-loop cost, the terminal value function needs to tractably approximate the true value function V^* of the MDP. In the end, different aspects can be considered when learning an appropriate terminal value function.

For instance, the terminal value function $\bar{V}_\theta^{\text{MPC}}$ can be learned via a temporal difference update, cf. [220], where a learned terminal value function is incorporated into an MPPI planner. In the following, a simplified version is provided to learn a parameterized terminal value function $\bar{V}_\theta^{\text{MPC}}$, potentially a NN, that can be defined via

Terminal value function learning ($\bar{V}_\theta^{\text{MPC}} \approx ?^1$)

$$\begin{aligned} \theta &\leftarrow \theta + \frac{\alpha}{B} \sum_{i=1}^B \delta_i \nabla_\theta \bar{V}_\theta^{\text{MPC}}(S_i), \\ \delta_i &:= l(S_i, A_i) + \bar{V}_\theta^{\text{MPC}}(S_i^+) - \bar{V}_\theta^{\text{MPC}}(S_i), \\ \text{with } S_i &\sim \mathcal{D}^{\pi_\theta^{\text{MPC}}}, A_i = \mu_\theta^{\text{MPC}}(S_i), S_i^+ \sim P(\cdot | S_i, A_i), \end{aligned}$$

where $\mathcal{D}^{\pi_\theta^{\text{MPC}}}$ is a distribution of states generated by controlling the system via the parametrized MPC exploration policy π_θ^{MPC} and α is the learning rate. Another approach to learning the terminal value function is shown in [263], where the authors use either model-free SAC or PPO to obtain a terminal value function for an MPC planner.

B. Closed-Loop Learning

In the following section, the paradigm of viewing MPC as a parameterized optimization layer as part of an actor policy is explained in more detail. In the closed-loop learning paradigm, the MPC model is not required to minimize a prediction loss \mathcal{L}^{id} or to provide a terminal value function that approximates the optimal value function as in the MDP aligned setting. Instead, the model, costs, or constraints may be changed to solely improve the closed-loop performance of the MPC policy μ^{MPC} according to Definition 4. Fig. 5 shows a sketch of the closed-loop optimal learning paradigm.

An advantage of this paradigm is the ability to achieve optimal closed-loop performance in the real environment without the necessity of computationally demanding aligned formulation of Sect. VIII-A. However, by modifying the various parts of the MPC purely to increase the closed-loop performance, explainability, safety, generalization, and adaptability get lost, particularly if the model or constraints are allowed to change.

Practically, this paradigm differs depending on whether the controls provided by the MPC or the parameterization of the MPC obtained through a learnable NN are assumed as RL actions. This two practical implementations are explained in the following.

1) Differentiable MPC: MPC can be used as an optimization layer within the RL policy to provide a good initial performance by leveraging knowledge about the task [179]. Ideally, the MPC

¹The question mark indicates that, to the best of the authors' knowledge, the update scheme does not converge to the optimal value function V^* in general, and its convergence target remains unclear.

TABLE V: Literature that uses MDP-aligned learning. Either the value function, the model, or the cost function of an MPC actor are learned. The table lists the applications and whether real-world experiments (RWE) were performed and which MPC formulation was used. Moreover, the learning or approximation algorithm used to obtain the model or the terminal value function approximation is stated. The model or the terminal value function of the MPC is either updated by using the current MPC policy to collect samples, i.e., on-policy (ON-P), or another policy, referred to as off-policy (OFF-P) sampling. The terminal value function can also be computed by offline DP or online planning (e.g., RRT) based on a given model. This computation does not utilize sampling strategies, thus is neither related to on-policy or off-policy learning and indicated by N/A.

MPC as an Actor: MDP Aligned Supervised Learning									
Ref.	Authors	Year	Learning of	Architecture/ specific FA	Application	RWE	MPC Formulation (Algorithm, Solver)	Learning/Approx. Algorithm	Sampling
[264]	Zhong et al.	2013	term. val. fun.	integrated/ several FAs	pendulum and acrobot	no	NMPC (iLQR, custom)	DP	OFF-P
[265]	Aswani et al.	2013	model	parameterized	HVAC, quadrotor	yes	NMPC (SQP, SNOPT [266])	supervised	ON-P
[58]	Nagabandi et al.	2018	model	integrated/ NN	MuJoCo locomotion	no	random sampling (custom)	DAgger-like	ON-P
[220]	Lowery et al.	2019	term. val. fun.	integrated/ NN	3D humanoid, five-fingered hand	no	MPPI (custom)	supervised	ON-P
[234]	Wang et al.	2019	model	integrated/ NN	MuJoCo locomotion	no	CEM [267] (N/A)	IL	ON-P
[268]	Deits et al.	2019	term. val. fun.	integrated/ NN	walls pendulum 2D humanoid	no	MIQP-MPC (Gurobi [195])	supervised	OFF-P
[269]	Yang et al.	2019	model	integrated/ NN	legged robot [270]	yes	CEM [267] (custom)	supervised	ON-P
[271]	Lambert et al.	2019	model	integrated/ NN	Crazyflie quadrotor	yes	random sampling (custom)	supervised	OFF-P
[272]	Karmananachari et al.	2020	term. val. fun.	integrated/ NN	UGV	yes	NMPC (SQP, acado [129])	TD	ON-P
[273]	Beckenbach et al.	2020	term. val. fun., stage cost	parameterized	chemical reaction	no	N/A	TD	N/A
[274]	Hoeller et al.	2020	term. val. fun.	integrated/ NN	ballbot [274]	yes	NMPC (SLQ/iLQR, custom)	TD	ON-P
[275]	Hatch et al.	2021	term. val. fun.	custom/ roll-out	four-wheeled skid-steered robot	no	MPPI (custom)	DP-related (RRT#)	N/A
[276]	Morgan et al.	2021	model	integrated/ NN	MuJoCo, robotic hand	yes	MPPI (custom)	SAC	ON-P
[206]	Dobriborsci et al.	2022	term. val. fun.	parameterized	mobile robot	yes	NMPC (SLSQP [207])	DQN	OFF-P
[277]	Beckenbach et al.	2022	term. val. fun.	parameterized	chemical reaction	no	N/A	ADP	N/A
[278]	Moreno-Mora et al.	2023	term. val. fun.	parameterized	spacecraft	no	NMPC (fminunc [279])	VI	OFF-P
[280]	Lin et al.	2024	term. val. fun.	integrated/ polynomial	mobile robot	no	LQR (N/A)	PI	ON-P
[263]	Reiter et al.	2024	term. val. fun.	integrated/ NN	autonomous driving	no	NMPC (SQP, acados [130])	PPO, SAC	OFF-P
[281]	Qu et al.	2024	term. val. fun. term. val. fun., stage cost,	integrated/ NN	unmanned aerial vehicle	no	MPPI (custom)	SAC	OFF-P
[282]	Cai et al.	2023	model, constraints	parameterized	home energy management	no	NMPC (N/A)	TD3, AC	ON-P

performance would only be slightly suboptimal and provide safety guarantees for a known model. The hope would be that by only a few RL iterations, the parameters can be modified towards nearly optimal closed-loop performance, particularly related to stochasticity.

However, the MPC optimization problem as part of the learned actor creates potential challenges for both the forward path, where the policy evaluation includes solving the optimization problem, and the training of parameters, which, in some algorithms, requires appropriately passing gradients through an optimization algorithm. Solving optimization problems is numerically challenging. Thus, the related iterative algorithms may slow down learning when evaluating a policy. Moreover, obtaining a global optimizer outside the class of convex optimization problems is intractable in general. Although local optima are often sufficient but require warm-starting, the initial states used to warm-start an optimization solver introduce additional states, which are required to be considered in the learning algorithm.

When using the MPC in an RL actor, gradients need to be computed through the optimization solver as part of the backpropagation. These gradients can be computed using the implicit function theorem, cf. App. A and related literature [283], [284], [285] for further details. However, existing optimization software is required to support such features. Related features were recently developed in various tools, see Sect. XI.

When differentiating through the optimization layer, the

integrated, hierarchical, parallel, and parameterized architectures according to Fig. 3 may be used, but to the best of the authors' knowledge, only the hierarchical, e.g. [286], [285], and the parameterized architectures, e.g., [283], [224], [225], were proposed so far.

The following algorithm based on the ideas of [224] provides a basic version of closed-loop optimal learning, where the parameters are updated by differentiating the MPC. The temporal difference update (13) is adapted to the parameterized MPC setting to

Q-learning with MPC Q-function ($Q_\theta^{\text{MPC}} \approx Q^*$)

$$\begin{aligned} \theta &\leftarrow \theta + \alpha \delta \nabla_\theta Q_\theta^{\text{MPC}}(S, A), \\ \delta &:= l(S, A) + \gamma V_\theta^{\text{MPC}}(S^+) - Q_\theta^{\text{MPC}}(S, A), \\ \text{with } (S, A) &\sim \mathcal{D}^{\pi_\theta^{\text{MPC}}}, S^+ \sim P(\cdot | S, A). \end{aligned}$$

The state-action distribution \mathcal{D} is generated by controlling the system via the stochastic parameterized MPC exploration policy π_θ^{MPC} . Note that the update scheme only considers a single sample, which is often reasonable in the case that the Q_θ^{MPC} is just a parameterized MPC scheme as described in Fig. 3, without a NN.

Another example of closed-loop optimal learning based on a DDPG actor-critic formulation similar to (19) was introduced in [224] and is given by

DDPG with MPC actor ($\mu_\theta^{\text{MPC}} \approx \pi^*$)

$$\begin{aligned} w &\xleftarrow{Q} w + \frac{\alpha_w}{B} \sum_{i=1}^B \delta_i \nabla_w Q_w(S_i, A_i), \\ \theta &\xleftarrow{\mu} \theta + \frac{\alpha_\theta}{B} \sum_{i=1}^B \nabla_\theta \mu_\theta^{\text{MPC}}(S_i) \nabla_a Q_w(S_i, a)|_{a=\mu_\theta^{\text{MPC}}(S_i)}, \\ \delta_i &:= l(S_i, A_i) + \gamma Q_{\bar{w}}(S_i^+, A_i^+) - Q_w(S_i, A_i), \\ \text{with } (S_i, A_i, S^+) &\sim \mathcal{D}^{\text{buffer}}, \quad A_i^+ = \mu_\theta^{\text{MPC}}(S_i). \end{aligned}$$

The parameters are updated for the deterministic policy μ_θ^{MPC} and a NN critic Q_w with parameters w . The stochastic policy π_θ^{MPC} is used for exploration in order to fill the replay buffer $\mathcal{D}^{\text{buffer}}$.

2) *MPC as part of the environment*: Passing gradients through the MPC in the actor in the closed-loop optimal learning paradigm can also be omitted by, conceptually, considering the MPC as part of the environment. In such a setting, a parameterized MPC that controls an environment can be seen as an augmented new environment whose actions are the MPC parameters. Accordingly, the RL critic evaluates the values related to these parameters instead of the actions provided by the MPC, which are, in fact, hidden from the RL framework. Considering the parameterization of the MPC as the environment input, instead of the actions obtained from the MPC, avoids computing sensitivities but comes at the cost of potentially vastly increasing the dimensions of the action space and obtaining gradient information through sampling. Such a setting usually involves the hierarchical [221], [214] or the parameterized architecture [287] of Fig. 3.

Whether MPC is considered as part of the environment in literature, involving its differentiation, is indicated by the differentiating MPC column in Tab. VI.

3) *Exploration*: Tab. VI lists closed-loop optimal algorithms that differ in the architecture and whether they include differentiation through the optimization layer. Moreover, Tab. VI considers how exploration was performed in related literature, cf. Sect. III-B. Exploration is often required to improve the currently learned policy in RL. Four choices of exploration are used, particularly in the case of hierarchical and parameterized architectures:

- 1) Adding noise to the action proposed by the optimization layer. The downside of adding noise posterior to the optimizer is the potentially unsafe exploration, depending on the chosen exploration noise.
- 2) Modifying the cost in the optimization layer (25a) with an additive term $d^\top u_0$ where d is a possibly randomly selected vector. Given that only the cost is modified, the actions remain feasible with respect to the model while the additive term introduces a gradient over the initial control input [288] for exploration.
- 3) Add a perturbation to the parameter ϕ for the hierarchical architecture and θ for the parameterized architecture as shown in Fig. 3, guaranteeing safe actions [289].

MPC as reference generator

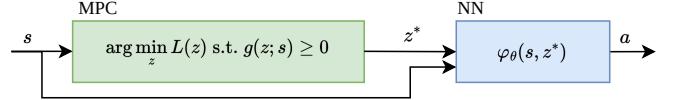


Fig. 6: MPC can be used as a reference generator for an RL policy. The distribution of the input of the policy may depend on the distribution of optimizers obtained from MPC while interacting with the environment.

- 4) Instead of noise on the parameters or controls, optimistic initialization is an RL technique where Q-values (or estimates) are initialized with higher-than-expected values, encouraging the agent to explore and gradually reduce these optimistic estimates to reflect the true values. For a discussion in the context of deep RL see [290].

Closed-loop optimal learning algorithms require the MPC to be part of the RL algorithm. Yet, the following two variants in Sect. VIII-C use a hierarchical MPC and RL setting, where MPC is added posterior to the training in the deployed controller.

C. MPC for Pre- and Postprocessing

In the following Section, two concepts are highlighted that use MPC within the deployed policy, yet, not during the training procedure. Since the trained policy is not aware of the filter, the expected performance can not be expected to be closed-loop optimal, as in the previous section. Depending on the purpose of the MPC, a distinction is made between an MPC used as a reference generator for preprocessing, cf. Fig. 6 and Sect. VIII-C1, and MPC used for postprocessing, cf. Fig. 7 and Sect. VIII-C2. Relevant literature is compared in Tab. VII.

1) *Preprocessing: MPC as a Reference Generator*: MPC may be used as a reference generator for an RL policy such as in [218]. This combination approach of MPC and RL is particularly useful if the output of MPC is a planned trajectory rather than a single action. Since solving the MPC may be slow, the RL policy may be trained with a computationally cheaper reference generator, as proposed in [218]. Notably, this setting may often be used in publications without an explicit statement that describes the reference provider.

2) *MPC for Postprocessing*: Adding MPC to a trained policy may fulfill one of the two purposes: (i) providing safety w.r.t. an assumed model and constraints, known as safety filter [301] or (ii) providing a coarse solution by a warm-start of an optimization solver.

RL policies may struggle with guaranteeing safety [9] and plan for smooth trajectories, particularly in a high-dimensional state space [302].

As opposed to the previous Sect. VIII-B of closed-loop optimal learning, the approaches that use MPC as a filter do not consider MPC during learning. Thus, the expected behavior may not be closed-loop optimal since the policy is unaware of MPC during training.

TABLE VI: Literature that uses MPC as part of the actor in closed-loop learning. The table lists the applications, whether real-world experiments (RWE) were performed, and the particular RL and MPC algorithms. The computation of gradients during the back-propagation pass of the RL algorithm may require the computation of the sensitivities of the MPC solution, indicated in the column “differentiating MPC”. Alternatively, the sensitivities could be sampled, by, e.g., considering the MPC as part of the environment. Similarly, the table compares where the exploration noise of the RL algorithm is added. The noise could be added to the parameters of the MPC optimization problem, or posterior, to the optimal controls provided by the MPC or by using an optimistic initialization of the value functions.

MPC as an actor: closed-loop optimal									
Ref.	Authors	Year	Exploration	Diff. MPC	Architecture	Application	RWE	MPC Formulation (Algorithm, Solver)	RL Algorithm
[283]	Amos et al.	2018	actions	yes	parameterized	pendulum, cartpole	no	LMPC (N/A)	IL
[291]	Greatwood et al.	2019	optimistic	no	hierarchical	drone navigation	yes	LMPC (custom)	TD
[292]	Tram et al.	2019	N/A	no	hierarchical	vehicle intersection	no	LMPC (N/A)	Q-learning
[224]	Gros et al.	2020	actions	yes	parameterized	evaporation process	no	NMPC (SQP, acados [130])	Q-learning
[221]	Brito et al.	2021	parameters	no	hierarchical	multi-agent unicycle	no	NMPC (N/A, ForcesPro[293])	PPO
[284]	Zanon et al.	2021	actions	yes	hierarchical	evaporation process	yes	LMPC (N/A)	Q-learning
[225]	Moradimaryamnegari et al.	2022	N/A	yes	parameterized	water tank	yes	NMPC (IP, IPOPT [203])	SARSA
[294]	Brito et al.	2022	parameters	no	hierarchical	highway traffic	no	NMPC (N/A, ForcesPro[293])	SAC
[295]	Zhang et al.	2022	parameters	no	hierarchical	quadruped robot	no	LMPC (N/A)	
[296]	Pfrommer et al.	2022	actions	no	hierarchical	2D linearized quadrotor	no	LMPC (MOSEK [297])	PG
[214]	Reiter et al.	2023	parameters	no	hierarchical	autonomous racing	no	NMPC (SQP, acados [130])	SAC
[226]	Liu et al.	2023	N/A	yes	parameterized	multi-agent games	yes	generalized Nash equilibrium solver (custom)	custom
[285]	Romero et al.	2024	actions	yes	hierarchical	drone racing	yes	NMPC (iLQR, custom)	PPO
[298]	Tao et al.	2024	N/A	yes	parameterized	quadrotor	no	NMPC (SQP, acados [130])	N/A
[299]	Zarrouki et al.	2024	parameters	no	hierarchical	race car	yes	NMPC, (SQP, acados [130])	PPO
[287]	Zarrouki et al.	2024	parameters	no	parameterized	race car	yes	NMPC (SQP, acados [130])	PPO
[300]	Wen et al.	2024	parameters	no	hierarchical	mobile robot	no	LMPC (N/A)	PPO

MPC for postprocessing

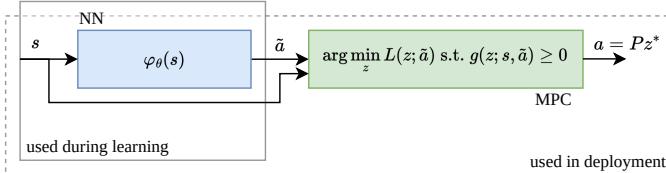


Fig. 7: MPC can be used to smooth or filter reference trajectories. For example, MPC may be used to provide safety guarantees that are hard to achieve by RL policies.

Tab. VII shows relevant work that uses the MPC for postprocessing and distinguishes the filtering of a single proposed action, e.g., used in the safety-filter framework [301], or a whole trajectory of actions, e.g., [303] or filtering of a state trajectory as in [304]. The filtering of action or state trajectories can be interpreted as providing an initial guess of decision variables of a nonconvex optimization problem to a solver, which then aims to find a good local minimum. In fact, the MPC needs to be approximately aligned with the MDP, and the role of the policy rather assists the MPC optimization solver by finding global/low-cost optimizers.

One ought to observe, though, that if an MPC formulation is available to ensure the safety of the action taken in the real environment, typically in the form of a robust MPC scheme, then it is debatable whether training a policy (typically based on a NN) to be filtered by the robust MPC scheme or training the robust MPC scheme directly [284] is more effective.

So far, MPC has been considered as part of the actor or as an expert actor. In the following, we also highlight works that consider MPC as part of the critic, i.e., the MPC is used solely during training to provide a value function estimate.

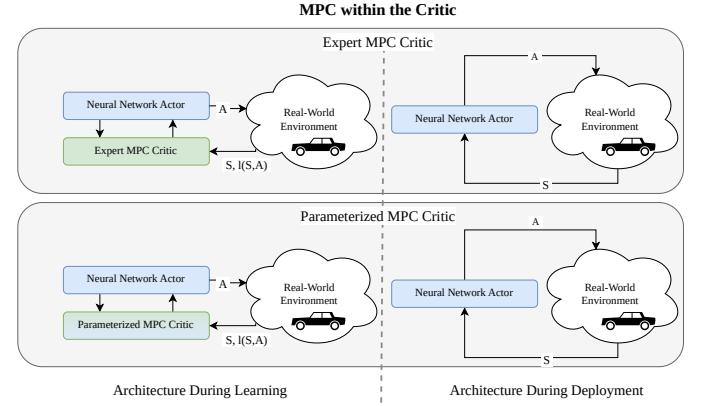


Fig. 8: Combinations: MPC as a critic. The plot is split into the learning and deployment phases. Blue boxes indicate Neural Networks (NNs), green boxes are used for MPCs, and blue/green boxes refer to parameterized MPCs that involve parameters/NNs that are learned during the learning phase, see Sect. VI-A.

IX. MPC AS A CRITIC

As outlined in Sect. IV, parameterized variants of the OCPs (22) and (23) allow for a structured function approximation of value function, action-value function, and the policy. While earlier discussions focused on using MPC as an actor, we next discuss the role of MPC used as a critic, cf., Fig. 2. We distinguish between three variants of MPC as a critic.

- A. **MPC as an expert critic**, Sect. IX-A. The MPC is parameterized based on expert knowledge of the problem at hand, entering through the cost, model, or constraints, which we refer to as expert critic.
- B. **MPC as a learnable critic**, Sect. IX-B. The MPC involves learnable parameters to improve the accuracy of the critic.
- C. **MPC as a learnable actor-critic**, Sect. IX-C. The critic parameterization (possibly) differs from the one of the

TABLE VII: Literature that uses MPC during the deployed algorithm together with the RL policy but not within the learning phase. Algorithms either use MPC as a reference provider for an RL policy or for postprocessing. Postprocessing is used to provide safety w.r.t. a known model and constraints or to take an action or state trajectory as an initial guess for the optimization solver.

MPC for Postprocessing							
Ref.	Authors	Year	Filtering of	Application	RWE	MPC Formulation (Algorithm, Solver)	RL Algorithm
[305]	Li et al.	2020	action	point-mass, kin. vehicle	no	NMPC, robust (N/A)	DDPG
[301]	Tearle et al.	2021	action	miniature race cars	yes	NMPC (SQP, acados [130])	IL, DAgger [193]
[302]	Shen et al.	2023	traj. roll-out	multi-vehicle motion planning	no	NMPC (IP, IPOPT [203])	Q-learning
[306]	Didier et al.	2023	action	autonomous driving	no	NMPC (IP, IPOPT [203])	N/A
[304]	Grandesso et al.	2023	traj. roll-out	3-DOF planar manipulator	no	NMPC (IP, IPOPT [203])	custom AC
[162]	Mamedov et al.	2024	action	flexible robot arm	no	NMPC (SQP, acados [130])	BC, DAgger, AIRL GAIL, PPO, SAC
[263]	Reiter et al.	2024	traj. roll-out	vehicle motion planning	no	NMPC (SQP, acados [130])	PPO, SAC
[307]	Alboni et al.	2024	traj. roll-out	point-mass with obstacles	no	NMPC (IP, IPOPT [203])	custom AC
[303]	Ceder et al.	2024	action roll-out	mobile robot	no	NMPC (proximal gradient, PANOC [308])	DDPG
[281]	Qu et al.	2024	traj. roll-out	unmanned aerial vehicle	no	MPPI (custom)	SAC
MPC as Reference Generator							
[218]	Jenelten et al.	2024		quadruped robot	yes	NMPC (TAMOLS [309])	PPO
[310]	Bang et al.	2024		humanoid robot	no	LMPC (N/A)	PPO

actors.

In Tab. VIII, different publications are presented, depending on whether MPC parameters are tuned and which critic type described in the following three sections was used.

A. MPC as an Expert Critic

An MPC scheme, as previously discussed, can readily deliver an action-value function Q^{MPC} , which can be used to derive policy gradient equations to train an actor. An important assumption here is that the value function Q^{MPC} approximates Q^* as closely as possible. In fact, the MPC may usually just provide a value function of a desirable good suboptimal policy. Approximating Q^* by the fixed MPC Q-function was recently proposed in [219]. Very related is the line of work in [259], [219] that, inspired by the Hamiltonian-Jacobi-Bellman equations, uses a first-order approximation of a Q-value function to criticize the actions generated by a learned policy.

Consider the evaluation of the critic, involving solving the MPC optimization problem to obtain the corresponding action-value function Q^{MPC} . Then, for a deterministic, parameterized policy μ_θ , the learning objective related to (14) is

$$J_{\text{MPC}}^\mu(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{S \sim \rho^{\mu_\theta}} \left[Q^{\text{MPC}}(S, \mu_\theta(S)) \right], \quad (28)$$

with the discounted visitation frequency ρ^{μ_θ} as defined in Sect. III-A3. Note that the policy parameter θ is updated to minimize the objective (28) instead of the behavior cloning objective in Sect. VII or the MDP objective in (14).

An update scheme using the Q-function from MPC to criticize the learned policy μ_θ as in [219] is defined by

DDPG with an MPC expert critic ($\mu_\theta \approx \mu^{\text{MPC}}$)

$$\theta \leftarrow \theta + \frac{\alpha_\theta}{B} \sum_{i=1}^B \nabla_\theta \mu_\theta(S_i) \nabla_a Q^{\text{MPC}}(S_i, a)|_{a=\mu_\theta(S_i)},$$

with $S_i \sim \mathcal{D}$,

which is related to the DPG in (16) but with a fixed expert critic. Here, \mathcal{D} could be either a fixed dataset of states or generated iteratively by a mixture of μ_θ and μ^{MPC} like in [193].

Similar to Sect. VII, using an MPC as an expert critic leads to imitating the MPC expert policy. The primary advantage of using the MPC as an expert critic when compared to standard IL that relies on the mismatch between the expert and learned controls is its ability to guide a function approximator with limited expressive power by emphasizing which actions are more important to fit by using the Q-function. It further can introduce the constraint satisfactions to the objective using slacked constraints [219].

It is important to emphasize that the MPC expert critic remains fixed and, therefore, does not approximate the action-value function Q^{μ_θ} of the learned policy during training. Indeed, policy gradient methods formally require the critic to be of the policy, i.e., to deliver the policy action-value function Q^{μ_θ} . In contrast, an MPC as an expert critic instead delivers the action-value function Q^{MPC} based on expert knowledge of the environment, with the aim of approximating the optimal action-value function Q^* as accurately as possible. If the actor has enough flexibility to clone the MPC policy perfectly, then using MPC as an expert critic would ultimately result in the actor matching the MPC performance but not improving it further.

Consequently, as discussed in the following sections, further

improvements can be achieved by allowing the MPC critic to adapt to the currently learned policy μ_θ .

B. MPC as a Learnable Critic

In the context of policy gradient methods, a parameterized MPC of any architecture shown in Fig. 3 can be used to approximate the policy action-value function Q^π , and updated using data to capture it as correctly as possible. For instance, the parallel architecture of Fig. 3 is used in [223]. Using the MPC sensitivities, the MPC parameters can be learned using value-based methods from (13). As discussed before, a parametric MPC can fully capture Q^π given that it has a rich parameterization [282], [311], [312], [313]. This approach can be thought of as introducing expert knowledge into the policy gradient pipeline, using a critic combining harmoniously model-based knowledge and learning to capture Q^π as accurately and quickly as possible. The main reasoning behind this combination is that MPC is typically capable of providing a correct structure for the action-value function Q^π prior to any training, giving a good starting point for the policy-gradient method. Then, the classic learning of Q^π allows the MPC to become a better critic of the policy and to remain a good critic as the policy is modified by the policy-gradient method. It is worth mentioning here that MPC as a learnable critic can be readily combined with a more classic NN, e.g., as a summation of their contribution to the action-value approximation [314]. In that context, the MPC scheme can deliver a broadly correct approximation, while the NN can provide fine corrections. Unlike the approach of Sec. IX-A, MPC as a learnable critic aligns well with the actor-critic framework.

C. MPC as a Learnable Actor-Critic

Using MPC as an actor and as a learnable critic naturally offers the opportunity to combine them into an actor-critic setup using MPC for both. In this setting, MPCs can be used both as an actor, delivering a parameterized policy π_θ^{MPC} to be trained, and as a critic, delivering value functions that are by construction close to the value function of the MPC policy, i.e., $V_\theta^{\text{MPC}} \approx V^{\pi_\theta^{\text{MPC}}}$, $Q_\theta^{\text{MPC}} \approx Q^{\pi_\theta^{\text{MPC}}}$, as opposed to random initializations of NNs.

During the learning, the MPC scheme operating as an actor typically needs to differ from the MPC scheme operating as a critic because π is not a minimizer of Q^π when the policy is not optimal. In practice, two different MPC schemes can be used and trained in parallel: one as a critic, maintaining a good approximation of $Q^{\pi_\theta^{\text{MPC}}}$ associated with π_θ^{MPC} given by (23), and one to support π_θ^{MPC} itself, given by (22). It can be, however, computationally expensive to use two MPC schemes in parallel, as the optimal solution of both must be produced at every training step of the policy, rather than the solution of only the MPC supporting the policy π_θ^{MPC} .

Along that line, [315] proposed a critic formulation based on the value function obtained from an MPC-based actor. Their formulation employed V_θ^{MPC} along with the compatible function approximation suggested in [27] to build a local approximation of the critic by using a first-order Taylor expansion, simplifying the actor-critic formulation with a single MPC scheme. This

approach is effective if the MPC policy is sufficiently close to the optimal policy.

X. THEORETICAL CONSIDERATIONS FOR COMBINING MPC AND RL

MDPs establish a fundamental bridge between RL and MPC, as RL provides tools to solve MDPs while MPC formulations can provide approximations, as discussed in the context of aligned learning in Sect. VIII-A and closed-loop optimal learning in Sect. VIII-B. This section explores the theoretical foundations of combining MPC and RL within the framework of MPC-based MDP approximations. A central theoretical contribution by [224] provides a theoretical link between economic nonlinear model predictive control (ENMPC) and RL by connecting the MDP value functions and policy discussed in Sect. II with those generated by derivative-based MPC formulations detailed in Sect. IV-B2. This connection provides theoretical justification for approaches that incorporate MPC as either an actor or critic, as described in Sect. VIII and Sect. IX, respectively. We begin by presenting this key theoretical result before outlining its subsequent developments.

Following the definition in Sect. III-A3, let $p_{f^{\text{MPC}}}(s_0 \rightarrow s^+, k, \pi_\theta)$ denote the probability of reaching state s^+ at step k when starting from the initial state s_0 and following the policy π , under model dynamics f^{MPC} . A key assumption in [224], [262] is that the optimal value function remains bounded under the optimal policy and model of the dynamics:

Assumption 10.1 ([262, Assumption 1]): The following set is non-empty for a given $\bar{N} \in \mathbb{N}$.

$$\mathcal{S} = \left\{ s \in \mathcal{X} \mid \left| \mathbb{E}_{S \sim p_{f^{\text{MPC}}}(s_0 \rightarrow s^+, k, \pi^*)} [V^*(S)] \right| < \infty, \forall k \leq \bar{N} \right\} \quad (29)$$

The authors of [224] demonstrate that discounted MPC can capture the optimal value functions and policy of a discounted MDP through modifications of the stage and terminal costs, even when the model f^{MPC} differs from the true state-transition function. This result was later extended in [262] to undiscounted MPC by the following Theorem, establishing a central link to classical MPC stability theory:

Theorem 10.1 ([262, Theorem 1]): Suppose that Assumption 10.1 holds for $\bar{N} \geq N$. Then, there exists a terminal cost $\bar{V}_\theta^{\text{MPC}}$ and a stage cost $\bar{l}_\theta^{\text{MPC}}$ such that the following identities hold, for all $\gamma, N \in \mathbb{N}$ and $s \in \mathcal{S}$:

- 1) $\pi_\theta^{\text{MPC}}(s) = \pi^*(s)$,
- 2) $V_\theta^{\text{MPC}}(s) = V^*(s)$,
- 3) $Q_\theta^{\text{MPC}}(s, a) = Q^*(s, a)$, for the inputs $a \in \mathcal{U}$ such that $\mathbb{E}_{S^+ = f^{\text{MPC}}(s, A), A \sim \pi_\theta^{\text{MPC}}(\cdot | s)} [V^*(S^+) | s, A] < \infty$.

Proof 10.1: See [262].

Given the conditions in Theorem 10.1, we classify MPC formulations that satisfy condition 1) as closed-loop optimal, where the learning process that aims at satisfying condition 1) using MPC within the actor as closed-loop optimal learning, see VIII-B. When both conditions 1) and 2) are satisfied, the MPC is MDP complete since it captures both the optimal policy and optimal action-value function of an MDP. At the top of this theoretical hierarchy lies the MDP aligned property, which demands that the MPC model exactly matches the state-transition kernel of the MDP. However, this property is rarely

TABLE VIII: Literature that uses MPC as a critic for RL. The MPC parameters are either learned within the RL algorithm or fixed. Further, a distinction is made, whether the critic is a fixed expert, see Sect. IX-A, a more flexible learned critic as described in Sect. IX-B, or is part of both the actor and the critic, Sect. IX-C.

MPC as a Critic								
Ref.	Authors	Year	Learned MPC Parameters	Critic Type/ Par. MPC Arch.	Application	RWE	MPC Formulation (Algorithm, Solver)	RL Algorithm
[260]	Reske et al.	2021	no	expert / N/A	quadruped robot	yes	NMPC (DDP, N/A)	IL
[223]	Bhardwaj et al.	2021	no	learned / parallel	several robotic in-hand manipulation	no	MPPI	TD
[219]	Ghezzi et al.	2023	no	expert / N/A	cart pole	no	NMPC (SQP, acados [130])	IL
[315]	Anand et al.	2023	yes	Actor-Critic / parameterized	cart pole	no	NMPC (IP, IPOPT [203])	DPG
[304]	Grandesso et al.	2023	no	expert / N/A	3-DoF planar manipulator	no	NMPC (IP, IPOPT [203])	custom AC
[259]	Carius et al.	2023	no	expert / N/A	quadruped robot	yes	NMPC (DDP, N/A)	IL
[307]	Alboni et al.	2024	no	expert / N/A	Dubin's car parking	no	NMPC (DDP, IP, IPOPT [203])	custom AC

satisfied in practice, as MPC formulations typically employ deterministic models, while MDP state transitions are inherently stochastic.

A. Extensions of MPC-MDP Equivalence

Theorem 10.1 was originally developed in the context of ENMPC. Subsequent research has extended the result to several MPC variants, including real-time iteration MPC [286], mixed-integer MPC [316], scenario-based MPC [317], or any model-based policy [318].

Instead of using OCP formulations that rely on one-step prediction rollouts of the environment, [319] explores the connection between MDPs and QP formulations that arise for tracking problems with linear dynamics or as subproblems in OCP solvers. By parameterizing the QP directly rather than the prediction model, this approach offers additional flexibility that can enhance the performance of the MPC policy. Building on this idea, [320] incorporates formulations from subspace predictive control (SPC) [321], showing that the past input/output sequences can serve as a surrogate state from which future predictions are built. While this approach effectively handles systems without state estimators, its application is restricted to problems permitting autoregressive linear maps from inputs and outputs to future trajectories.

For systems requiring state estimation, an estimation layer such as moving horizon estimation (MHE) can be integrated to provide state feedback to the MPC. Research by [322], [312] demonstrated that jointly optimizing the parameters of both the MHE and MPC leads to significantly better performance compared to optimizing MPC parameters alone. The previously discussed challenges of differentiation through optimization layers and computational complexity in the forward pass extend naturally to the MHE layer.

The use of MPC and safety filters to ensure the safety of learned policies has emerged as a prominent research direction. One common approach projects control actions from trained policies onto a safe set by minimizing the distance to feasible actions that satisfy model dynamics and safety constraints. The authors of [185], [323] exemplify this strategy by introducing a predictive safety filter that post-processes actions from trained RL policies. However, this projection step may lead to sub-optimal actions when the agent is not aware of the filter, as discussed in [324]. An approach to learning safe and stable

policies by construction via robust MPC within the policy is discussed in [284] and further developed in [325], [326].

B. Approximating Discounted MDP with Undiscounted MPC

Research exploring the relationship between MPCs and MDPs has focused on identifying conditions under which undiscounted MPCs can approximate discounted MDPs. This connection draws from dissipativity theory [179], a key concept in ENMPC stability analysis. Using dissipativity arguments, [327] investigates undiscounted Q-Learning of tracking MPC schemes that are locally equivalent to dissipative ENMPC, focusing on learning storage functions that maintain ENMPC dissipativity. The authors of [328] prove that under weak stability conditions, the optimal policy of undiscounted and discounted MDPs coincide - enabling stable undiscounted MPCs to yield the optimal policy of stable discounted MDPs. Recent advances by [262] extend these results to unknown dynamics, demonstrating that undiscounted MPCs can capture optimal policies of discounted MDPs, as discussed around Theorem 10.1. Complementary work in [288] introduces methods to align undiscounted MPC cost function minimizers with those of discounted MDPs. These contributions share a common thread: by enforcing parameter updates that yield dissipative ENMPC formulations, stable ENMPC policies can be designed constructively rather than by using dissipativity solely as a verification criterion.

C. State-transition Model for Closed-Loop Optimality

Traditional MPC applications select prediction models by minimizing the identification loss (27), along with considerations regarding considerations such as convexity or smoothness. However, models achieving good mean-error predictions or maximum-likelihood models may not necessarily guarantee closed-loop optimality. Recent research by [332], [333] establishes formal requirements for prediction models in (23) to achieve closed-loop optimality and capture optimal value functions. They demonstrate that for a model f^{MPC} to deliver both the optimal policy π^* and optimal action-value function Q^* , it must satisfy:

$$\mathbb{E}_{S^+ \sim P(\cdot|s, a)} [V^*(S^+) | s, a] - V^*(f^{\text{MPC}}(s, a)) = V_0, \quad (30)$$

TABLE IX: Literature that discusses theoretical aspects of the connection between MPC and RL.

Ref.	Authors	Year	Contribution
MPC as a Model of the MDP			
[8]	Görge	2017	Compares explicit MPC for linear input and state-constrained systems to function approximation by NN in the context of RL
[329]	Zanon et al.	2019	Economic nonlinear model predictive control (ENMPC) as function approximation
[224]	Gros et al.	2020	Fundamental principles for ENMPC to approximate MDP and its connection to RL
[286]	Zanon et al.	2020	Extends [224] to Real-Time Iteration NMPC
[316]	Gros et al.	2020	Extends [224] to Mixed-Integer MPC
[317]	Kordabad et al.	2021	Extends [224] to scenario-based MPC
[322]	Esfahani et al.	2021	Use combined parameterized MHE and parameterized MPC as a function approximation in RL
[319]	Sawant et al.	2022	Extends [224] to generic parameterized QP layers instead of MPC
[320]	Sawant et al.	2023	Extends to subspace predictive control (SPC) using input-output data as surrogate state
[311]	Seel et al.	2022	Extends the cost to convex artificial neural network (NN) formulations to facilitate a more complex function approximation
[315]	Anand et al.	2023	Proposes deterministic policy gradient to train MPC via a parameter-perturbed variant as the critic.
[312]	Esfahani et al.	2023	Use combined parameterized MHE and parameterized MPC (extension of [322])
Post-processing and Safety			
[324]	Gros et al.	2020	Discusses the impact of safe policy projections, cf. [185], on learning and proposes corrections in the context of Q-learning
[284]	Zanon et al.	2021	Addresses safety in RL via robust MPC and safe parameter updates
[185]	Wabersich et al.	2021	Address safety via MPC-based predictive safety filters that modify unsafe actions from RL policies
[323]	Wabersich et al.	2021	Address safety via MPC-based predictive safety filters that modify unsafe actions from RL policies
[325]	Gros et al.	2022	Studies safety and stability of MPC schemes subject to parameter updates by an RL policy
[326]	Kordabad et al.	2022	Use Wasserstein Distributionally Robust MPC as a tool to generate safe RL policies
Approximation Theory			
[327]	Kordabad et al.	2021	Proposes Q-learning to modify the storage function in ENMPC such that it meets dissipativity criteria.
[330]	Kordabad et al.	2021	Address the problem that MPC-based policies with hard constraints can lead to biased gradient estimates in actor-critic methods.
[179]	Gros et al.	2022	Extends dissipativity theory of ENMPC to a wider class of probabilistic dynamics to connect to undiscounted MDPs.
[328]	Zanon et al.	2022	Connects discounted and undiscounted MDPs with known dynamics to generate stability-constrained optimal policies through MPC.
[288]	Kordabad et al.	2023	Proposes a cost modification for discounted ENMPC such that stage-cost minimizer coincides with the undiscounted case.
[331]	Seel et al.	2023	Proposes a combination of policy gradient and Q-learning to exploit gradient information and capture cost function properties not affecting the policy.
[262]	Kordabad et al.	2024	Extends [328] to mismatching dynamics and extends [224] to capture a discounted MDP via undiscounted MPC.

where V_0 is a constant. While (30) does not directly yield a system identification procedure, it provides insights into models minimizing (27). Such models prove optimal for deterministic MDPs and LQR problems. Local optimality can be established for tracking problems, set-point stabilization, and dissipative economic problems by bounding the conditional covariance of the state transition dynamics P given in (1) and the curvature of the (locally smooth) optimal value function V^* . However, models minimizing (27) may not yield optimal policies for problems with non-smooth cost functions or dynamics, non-dissipative MDPs, or strong disturbances.

The authors of [334] approach the problem of model design by formulating an equivalent expression to (30) in terms of the Bellman equation, introducing the optimal model design (OMD) method for simultaneous learning of Q-functions and models. This connects to broader developments in RL, such as the learned hidden state dynamic models in [335], which focus on capturing only the state information essential for optimal actions and rewards. While [332], [318], [333] focus on the theoretical properties of models in MPC contexts, [334] provides algorithmic tools for their construction.

XI. SOFTWARE TOOLS AND IMPLEMENTATION ASPECTS

This section highlights challenges in practical implementations and points to available software solutions for combining MPC and RL.

A. Integrating Software from Machine Learning and Numerical Optimization

The architectural combinations of NNs and MPCs described in Sect. VI and illustrated in Fig. 3 significantly influence the choice of optimization software, particularly for derivative-based MPC solvers. When objective or constraint functions are highly nonlinear, the resulting optimization problems become nonconvex and computationally challenging. Furthermore, the selected software must be compatible with machine learning libraries to enable seamless integration.

To address these challenges, recent open-source tools have emerged. The package `14casadi` [336] enables the integration of NNs $\varphi_\theta(s, z)$ within the automatic differentiation tool `CasADi` [337]. Similarly, `14acados` [338] bridges learned models from `PyTorch` to `acados`, focusing on residual models based on GPs in an integrated approach $\phi = \varphi_\theta(z; s)$. While these tools address derivative-based optimization, sampling-based MPC solvers offer an alternative approach, as they are less sensitive to nonlinear functions and only require forward simulation.

B. Implementation Aspects

Besides the difficulty of solving an optimization problem that involves NNs with derivative-based solvers, several further issues arise when combining MPC and RL, and should be considered in the implementation.

- 1) *Differentiation:* The MPC solver differentiation requires the computation of sensitivities - a process that is theoretically straightforward but demands existing solvers to

provide an efficient implementation of the parametric sensitivity computations. This capability is currently ongoing work in `acados` [130] and will be detailed in a future publication.

- 2) *Solver States*: MPC solvers often introduce additional solver states that must be incorporated into an augmented MDP. Nonlinear optimization processes start from initial guesses and may converge to different local minima, making these initial guesses part of the state space. This becomes particularly relevant in the RTI scheme [131] (cf. Sect. IV), where the solver tracks optima across iterations rather than achieving full convergence. These additional solver states increase the MDP's complexity.
- 3) *Computational Efficiency*: Some RL algorithms, such as SAC, require random sampling from replay buffers during learning. Computing gradients for each sample necessitates multiple solutions to optimization problems, resulting in significant computational overhead.
- 4) *Hardware Utilization*: While machine learning frameworks benefit from parallelizable architectures, MPC solvers typically target CPU-based embedded applications. Efforts to leverage GPUs or parallel CPU implementations represent an emerging research direction, with recent developments including GPU-based sampling MPC [113] and multi-core CPU implementations for derivative-based MPC in `acados` [130].

Complementary to this survey, early-stage software development aims at addressing these aspects. The code is publicly available as Learning for Predictive Control (`leap-c`)², a tool for derivative-based closed-loop learning utilizing MPC policies with the `acados` solver as a numerical optimization backend. Similar software projects with a lower degree of integration with fast OCP solvers exist³.

XII. CONCLUSION AND DISCUSSION

Model predictive control (MPC) and reinforcement learning (RL) each require extensive background knowledge, as detailed in Sect. III and Sect. IV, and possess complementary features as highlighted in Sect. V. The latter raises the question of how to effectively combine these approaches to leverage their respective strengths while mitigating their limitations. In Sect. VI, various possible combinations within the framework of solving Markov decision processes (MDPs) through actor-critic RL are demonstrated.

Within this framework, MPC can serve as an algorithmic part in multiple roles. One role is as a supervisory expert actor in imitation learning (IL) or guided policy search (GPS) frameworks as discussed in VII. This allows to include domain knowledge and other features to develop computationally efficient policies for online deployment.

Another role for MPC is as a part of the policy - the most prevalent approach in the literature as shown in Sect. VIII. The categorization becomes more intricate in this case. We distinguish between two main directions: MDP aligned approaches that aim at modifying the MPC to reflect the

MDP structure in terms of cost and state transitions, and closed-loop optimal approaches that focus on the closed-loop optimality of MPC. In both cases, the MPC enters the learning pipeline through an optimization layer that may be expensive to evaluate in the forward inference path or to differentiate in the backpropagation path in order to provide the respective gradients, cf. App. A. A challenge concerning the implementation is that the computation of gradients is often not provided by optimization solvers, despite recent implementations in, e.g., `acados` [130] and other software discussed in Sect. XI. It is possible to avoid the gradient computation of the MPC solution by integrating it into the environment (cf. Sect. VIII-B) such that its parameterization becomes part of the action space. The critic then evaluates the parameterization of the MPC as opposed to the actions provided by the MPC output. Considering MPC as a part of the environment and evaluating the critic on the MPC parameterization can be viewed as an example of the second practical architecture that we emphasize, i.e., architectures that use the MPC solver within the forward path but do not require its differentiation. Other architectures that avoid differentiation the category of using an expert MPC (Sect. VII), using a fixed MPC critic (Sect. IX) and on-policy learning of the model and the terminal value function in the MDP aligned structure (Sect. VIII-A) can be seen as variants that do not require differentiation but use MPC as part of the forward path. Still, the repeated solutions of optimization problems make learning iterations significantly more expensive, and implementation aspects and tailored software as outlined in Sect. XI should be considered.

The last role of MPC in the RL framework is that of a critic, as discussed in IX, which is not as prominent in the literature but may have the advantage of building explainable value functions estimates and significantly improving the sample efficiency and avoidance of local minima of RL training. Again, repeated MPC evaluations increase the computational cost compared to other artificial neural network (NN) layers.

Beyond these categories, several variants exist in the literature where MPC is not used during RL training but where MPC is solely used in the final deployed policy. These include off-policy learning of the model and some algorithms that learn a terminal value function in the MDP aligned structure (Sect. VIII-A) as well as pre- and post-processing components (Sect. VIII-C). In the latter, MPC can function as a safety filter to enhance trained policies with its inherent constraint satisfaction capabilities [167] and as a reference provider (VIII-C) - a role that may extend to the learning phase as well.

Yet another important design decision is the combination architecture of NNs and MPCs in Sect. VI for which we identified the integrated, hierarchical, parallel or parameterized approaches outlined in Fig. 3. NNs are usually highly nonlinear functions that deserve a distinction from moderately nonlinear functions used as part of derivative-based MPC. In Sect. VI, we contrast between using NNs outside the MPC, i.e., it does not depend on decision variables of the optimization problem, using NNs inside the MPC optimization problem or avoiding NNs and using parameters of simpler linear or quadratic functions.

²<https://github.com/leap-c/leap-c>

³<https://github.com/FilippoAiraldi/mpc-reinforcement-learning>

Many variants of classifying MPC and RL methods exist in the literature, with notable frameworks presented in [3], [8] and [17]. We used an actor-critic perspective and categories within that setting, that allows us to give a broad classification of very diverse algorithms.

As indicated in this survey, the expectations for the superiority of MPC and RL combinations are high, but challenges remain. Theoretical work provided already a solid foundation, cf. Sect. X. Nonetheless, practical work that shows large improvements over various domains is still required. Most importantly, the practical work requires numerical solvers to align with machine learning frameworks and tools that allow fast MPC solvers to be embedded in learning frameworks, see Section XI. With the increased development of such tools, the outlook on the MPC and RL combination appears prosperous. In future work, we hope the community will investigate whether the control paradigm shifts from MDP aligned learning to closed-loop optimal learning and whether the sampling complexity and other challenges of RL can be improved by MPC optimization modules.

REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*, second edition ed., ser. Adaptive computation and machine learning series. Cambridge, Massachusetts: The MIT Press, 2018.
- [2] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Santa Barbara, California: Nob Hill Publishing, 2017.
- [3] D. Bertsekas, *Reinforcement Learning and Optimal Control*, first edition ed. Belmont, Massachusetts: Athena Scientific, Jul. 2019.
- [4] G. B. Dantzig, *The Simplex Method*. Santa Monica, CA: RAND Corporation, 1956.
- [5] M. Schwenzer, M. Ay, T. Bergs, and D. Abel, “Review on model predictive control: an engineering perspective,” *The International Journal of Advanced Manufacturing Technology*, vol. 117, pp. 1327 – 1349, 2021.
- [6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nat.*, vol. 518, no. 7540, pp. 529–533, 2015.
- [7] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, L. Gilpin, P. Khandelwal, V. Kompella, H. Lin, P. MacAlpine, D. Oller, T. Seno, C. Sherstan, M. D. Thomure, H. Aghabozorgi, L. Barrett, R. Douglas, D. Whitehead, P. Dürr, P. Stone, M. Spranger, and H. Kitano, “Outracing champion Gran Turismo drivers with deep reinforcement learning,” *Nature*, vol. 602, pp. 223 – 228, 2022.
- [8] D. Görges, “Relations between Model Predictive Control and Reinforcement Learning,” *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4920–4928, Jul. 2017.
- [9] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, “Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.
- [10] B. Recht, “A Tour of Reinforcement Learning: The View from Continuous Control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. Volume 2, 2019, pp. 253–279, May 2019, publisher: Annual Reviews.
- [11] Y. Bengio, A. Lodi, and A. Prouvost, “Machine learning for combinatorial optimization: A methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, Apr. 2021.
- [12] D. Bertsekas, *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.
- [13] ———, “Newton’s method for reinforcement learning and model predictive control,” *Results in Control and Optimization*, vol. 7, p. 100121, Jun. 2022.
- [14] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, “Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,” Dec. 2017, arXiv:1712.01815 [cs].
- [15] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, publisher: Nature Publishing Group.
- [16] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, “Learning-based model predictive control: Toward safe learning in control,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, no. 1, pp. 269–296, 2020.
- [17] A. Mesbah, K. P. Wabersich, A. P. Schoellig, M. N. Zeilinger, S. Lucia, T. A. Badgwell, and J. A. Paulson, “Fusion of Machine Learning and MPC under Uncertainty: What Advances Are on the Horizon?” in *American Control Conference (ACC)*, Jun. 2022, pp. 342–357, iSSN: 2378-5861.
- [18] A. Norouzi, H. Heidarifar, H. Borhan, M. Shahbakhti, and C. R. Koch, “Integrating Machine Learning and Model Predictive Control for automotive applications: A review and future directions,” *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105878, Apr. 2023.
- [19] H. Zhang, S. Seal, D. Wu, F. Bouffard, and B. Boulet, “Building Energy Management With Reinforcement Learning and Model Predictive Control: A Survey,” *IEEE Access*, vol. 10, pp. 27853–27862, 2022.
- [20] T. Wang, X. Bao, I. Clavera, J. Hoang, Y. Wen, E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba, “Benchmarking Model-Based Reinforcement Learning,” Jul. 2019, arXiv:1907.02057 [cs].
- [21] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*, ser. Wiley series in probability and statistics. Hoboken, NJ: Wiley-Interscience, 2005, oCLC: 254152847.
- [22] R. Bellman, “Dynamic programming,” *American Association for the Advancement of Science*, vol. 153, no. 3731, pp. 34–37, 1966.
- [23] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*, ser. Optimization and neural computation series. Belmont, Mass: Athena Scientific, 1996.
- [24] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.
- [25] C. J. C. H. Watkins and P. Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3, pp. 279–292, May 1992.
- [26] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992.
- [27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, “Deterministic policy gradient algorithms,” in *Proceedings of the 31th international conference on machine learning, ICML 2014, beijing, china, 21-26 june 2014*, ser. JMLR workshop and conference proceedings, vol. 32. JMLR.org, 2014, pp. 387–395.
- [28] S. Fujimoto, H. van Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *Proceedings of the 35th international conference on machine learning, ICML 2018*, ser. Proceedings of machine learning research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 1582–1591.
- [29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proceedings of the 35th international conference on machine learning, ICML*, ser. Proceedings of machine learning research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 1856–1865.
- [30] C. R. Tilbury, F. Christianos, and S. V. Albrecht, “Revisiting the Gumbel-Softmax in MADDPG,” 2023, arXiv:2302.11793 [cs].
- [31] Y. LeCun, L. Bottou, G. B. Orr, and K. R. Müller, “Efficient BackProp,” in *Neural Networks: Tricks of the Trade*, G. B. Orr and K.-R. Müller, Eds. Berlin, Heidelberg: Springer, 1998, pp. 9–50.
- [32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” in *4th International Conference on Learning Representations, ICLR*, Y. Bengio and Y. LeCun, Eds., 2016.
- [33] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.
- [34] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust region policy optimization,” *CoRR*, vol. abs/1502.05477, 2015.
- [35] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, “High-Dimensional Continuous Control Using Generalized Advantage Esti-

- mation.” in *4th International Conference on Learning Representations, ICLR*, 2016.
- [36] J. Grudzien, C. A. S. De Witt, and J. Foerster, “Mirror learning: A unifying framework of policy optimisation,” in *International Conference on Machine Learning*. PMLR, 2022, pp. 7825–7844.
- [37] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, “Soft Actor-Critic Algorithms and Applications,” Jan. 2019, arXiv:1812.05905 [cs].
- [38] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” in *Proceedings of the 23rd AAAI conference on artificial intelligence*. AAAI Press, 2008, pp. 1433–1438.
- [39] P. J. Ball and S. J. Roberts, “OffCon3: What is state of the art anyway?” Mar. 2021.
- [40] B. Eysenbach and S. Levine, “Maximum Entropy RL (Provably) Solves Some Robust RL Problems,” in *10th International Conference on Learning Representations, ICLR*, 2022.
- [41] J. Müller and S. Cayci, “Essentially Sharp Estimates on the Entropy Regularization Error in Discounted Markov Decision Processes,” Jun. 2024.
- [42] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control. Theory and Algorithms*, 2nd ed. Springer, 2017.
- [43] L. Grüne, “Economic receding horizon control without terminal constraints,” *Automatica*, vol. 49, no. 3, pp. 725–734, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0005109812006024>
- [44] G. D. Nicolao, L. Magni, and R. Scattolini, “Stabilizing Receding-Horizon control of nonlinear time varying systems,” *IEEE Transactions on Automatic Control*, vol. AC-43, no. 7, pp. 1030–1036, 1998.
- [45] ———, “Stabilizing nonlinear receding horizon control via a nonquadratic terminal state penalty,” in *Symposium on Control, Optimization and Supervision, CESA’96 IMACS Multiconference*, Lille, 1996, pp. 185–187.
- [46] M. Diehl, L. Magni, and G. D. Nicolao, “Online NMPC of a looping kite using approximate infinite horizon closed loop costing,” in *Proceedings of the IFAC Conference on Control Systems Design*. Bratislava, Slovak Republic: IFAC, Sep. 2003.
- [47] ———, “Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing,” *Annual Reviews in Control*, vol. 28, no. 1, pp. 37–45, 2004.
- [48] B. D. O. Anderson and J. B. Moore, *Optimal Control - Linear Quadratic Methods*. Dover, 1990.
- [49] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed., ser. Springer Series in Operations Research and Financial Engineering. Springer, 2006.
- [50] E. C. Kerrigan, “Robust Constraint Satisfaction: Invariant Sets and Predictive Control,” PhD Thesis, University of Cambridge, UK, 2000.
- [51] J. B. Rawlings and M. J. Risbeck, “Model predictive control with discrete actuators: Theory and application,” *Automatica*, vol. 78, 2017.
- [52] A. Mesbah, “Stochastic Model Predictive Control: An Overview and Perspectives for Future Research,” *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 30–44, Dec. 2016, conference Name: IEEE Control Systems Magazine.
- [53] B. Kouvaritakis and M. Cannon, *Model Predictive Control: Classical, Robust and Stochastic*, 1st ed. Cham Heidelberg New York Dordrecht London: Springer, Dec. 2015.
- [54] A. Shapiro, D. Dentcheva, and A. Ruszczynski, *Lectures on Stochastic Programming: Modelling and Theory*. SIAM, 2009.
- [55] G. C. Calafiori and M. C. Campi, “The Scenario Approach to Robust Control Design,” *IEEE Trans. Automat. Control*, 2006.
- [56] W. Langson, S. R. I. Chryssochoos, and D. Q. Mayne, “Robust model predictive control using tubes,” *Automatica*, vol. 40, no. 1, pp. 125–133, 2004.
- [57] D. Mayne, E. Kerrigan, E. J. v. Wyk, and P. Falugi, “Tube-based robust nonlinear model predictive control,” *International Journal of Robust and Nonlinear Control*, vol. 21, pp. 1341–1353, 2011.
- [58] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7559–7566.
- [59] D. Kouzoupis, E. Klüntberg, G. Frison, S. Gros, and M. Diehl, “A dual Newton strategy for tree-sparse quadratic programs and its implementation in the open-source software treeQP,” *International Journal of Robust and Nonlinear Control*, 2019.
- [60] A. B. Kurzhanski and P. Valyi, *Ellipsoidal Calculus for Estimation and Control*. Birkhäuser Boston, 1997.
- [61] B. Houska, “Robust Optimization of Dynamic Systems,” PhD Thesis, KU Leuven, 2011.
- [62] J. Gillis and M. Diehl, “A Positive Definiteness Preserving Discretization Method for nonlinear Lyapunov Differential Equations,” in *CDC*, 2013.
- [63] M. E. Villanueva, R. Quirynen, M. Diehl, B. Chachuat, and B. Houska, “Robust MPC via min-max differential inequalities,” *Automatica*, vol. 77, pp. 311–321, Mar. 2017.
- [64] S. Rakovic, B. Kouvaritakis, M. Cannon, C. Panos, and R. Findeisen, “Parameterized Tube Model Predictive Control,” *Automatic Control, IEEE Transactions on*, vol. 57, no. 11, pp. 2746–2761, Nov. 2012.
- [65] S. V. Raković, “Robust Model Predictive Control,” in *Encyclopedia of Systems and Control*, J. Baillieul and T. Samad, Eds. London: Springer London, 2019, pp. 1–11.
- [66] J. Fleming, B. Kouvaritakis, and M. Cannon, “Robust Tube MPC for Linear Systems With Multiplicative Uncertainty,” *IEEE Trans. Automat. Control*, vol. 60, no. 4, 2015.
- [67] S. V. Raković, L. Dai, and Y. Xia, “Homothetic Tube Model Predictive Control for Nonlinear Systems,” *IEEE Trans. Automat. Control*, vol. 68, no. 8, 2022.
- [68] M. E. Villanueva, M. A. Müller, and B. Houska, “Configuration-Constrained Tube MPC,” *Automatica*, vol. 163, 2024.
- [69] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski, “Adjustable robust solutions of uncertain linear programs,” *Mathematical Programming*, vol. 99, no. 2, pp. 351–376, 2004.
- [70] T. Parisini and R. Zoppoli, “A receding-horizon regulator for nonlinear systems and a neural approximation,” *Automatica*, vol. 31, no. 10, pp. 1443–1451, Oct. 1995.
- [71] Z. K. Nagy and R. D. Braatz, “Open-loop and closed-loop robust optimal control of batch processes using distributional and worst-case analysis,” *Journal of Process Control*, vol. 14, pp. 411–422, 2004.
- [72] F. Messerer and M. Diehl, “An Efficient Algorithm for Tube-based Robust Nonlinear Optimal Control with Optimal Linear Feedback,” in *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2021.
- [73] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, “Optimization over state feedback policies for robust control with constraints,” *Automatica*, vol. 42, pp. 523–533, 2006.
- [74] P. O. M. Scokaert and D. Q. Mayne, “Min-max feedback model predictive control for constrained linear systems,” *IEEE Transactions on Automatic Control*, vol. 43, pp. 1136–1142, 1998.
- [75] M. Diehl, “Formulation of Closed Loop Min-Max MPC as a Quadratically Constrained Quadratic Program,” *IEEE Transactions on Automatic Control*, vol. 52, no. 2, pp. 339–343, 2007.
- [76] R. Reiter, M. Kirchengast, D. Watzenig, and M. Diehl, “Mixed-integer optimization-based planning for autonomous racing with obstacles and rewards,” *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 99–106, Jan. 2021.
- [77] A. Nurkanović, “Numerical Methods for Optimal Control of Nonsmooth Dynamical Systems,” PhD Thesis, University of Freiburg, 2023.
- [78] D. P. Bertsekas, “Dynamic Programming and Suboptimal Control: A Survey from {ADP} to {MPC}*,” *European Journal of Control*, vol. 11, no. 4, pp. 310–334, Jan. 2005.
- [79] L. Grüne and A. Rantzer, “On the infinite horizon performance of receding horizon controllers,” *IEEE Trans. Automat. Control*, vol. 53, no. 9, 2008, publisher: University of Bayreuth.
- [80] A. Karapetyan, E. C. Baltá, A. Iannelli, and J. Lygeros, “On the Finite-Time Behavior of Suboptimal Linear Model Predictive Control,” *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2023.
- [81] Y. Li, A. Karapetyan, J. Lygeros, K. H. Johansson, and J. Mårtensson, “Performance Bounds of Model Predictive Control for Unconstrained and Constrained Linear Quadratic Problems and Beyond,” *Proceedings of the IFAC World Congress*, 2023.
- [82] M. J. Hadjiyannis, P. J. Goulart, and D. Kuhn, “An Efficient Method to Estimate the Suboptimality of Affine Controllers,” *IEEE Trans. Automat. Control*, vol. 56, no. 12, 2011.
- [83] W. H. Fleming, “Stochastic Control for Small Noise Intensities,” *SIAM J. Control*, vol. 9, no. 3, 1971.
- [84] F. Messerer, K. Baumgärtner, S. Lucia, and M. Diehl, “Fourth-order suboptimality of nominal model predictive control in the presence of uncertainty,” *arXiv*, 2024.
- [85] Y. Lin, Y. Hu, G. Qu, T. Li, and A. Wierman, “Bounded-Regret MPC via Perturbation Analysis: Prediction Error, Constraints, and Nonlinearity,” *NeurIPS*, 2022.
- [86] H. Chen and F. Allgöwer, “A Quasi-Infinite Horizon Nonlinear Model Predictive Control Scheme with Guaranteed Stability,” *Automatica*, vol. 34, no. 10, pp. 1205–1218, 1998.

- [87] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 26, no. 6, pp. 789–814, 2000.
- [88] D. Limon and others, "Input-to-State Stability: A Unifying Framework for Robust Model Predictive Control," in *Nonlinear Model Predictive Control. Lecture Notes in Control and Information Sciences*, F. A. L. Magni, D. M. Raimondo, Eds. Springer, Berlin, Heidelberg, 2009, vol. 384.
- [89] L. Grüne, "NMPC without terminal constraints," *IFAC Proceedings Volumes*, vol. 45, no. 17, pp. 1–13, 2012, publisher: Elsevier.
- [90] J. Löfberg, "Oops! I cannot do it again: Testing for recursive feasibility in MPC," *Automatica*, vol. 48, no. 3, 2012.
- [91] D. Mayne, "An apologia for stabilising terminal conditions in model predictive control," *Internat. J. Control.*, vol. 11, 2013.
- [92] P. Scokaert, J. Rawlings, and E. Meadows, "Discrete-time Stability with Perturbations: Application to Model Predictive Control," *Automatica*, vol. 33, no. 3, pp. 463–470, 1997.
- [93] G. D. Nicolao, L. Magni, and R. Scattolini, "On the Robustness of Receding-Horizon Control with Terminal Constraints," *IEEE Trans. Automat. Control*, vol. 41, no. 3, 1996.
- [94] L. Magni and R. Sepulchre, "Stability margins of nonlinear receding-horizon control via inverse optimality," *Systems & Control Letters*, vol. 32, pp. 241–245, 1997.
- [95] R. D. McAllister and J. B. Rawlings, "Inherent Stochastic Robustness of Model Predictive Control to Large and Infrequent Disturbances," *IEEE Trans. Automat. Control*, vol. 67, no. 10, 2022.
- [96] G. Grimm, M. J. Messina, S. E. Tuna, and A. R. Teel, "Examples when nonlinear model predictive control is nonrobust," *Automatica*, vol. 40, pp. 1729–1738, 2004.
- [97] S. Yu, M. Reble, H. Chen, and F. Allgöwer, "Inherent robustness properties of quasi-infinite horizon nonlinear model predictive control," *Automatica*, vol. 50, 2014.
- [98] R. D. McAllister and J. B. Rawlings, "On the Inherent Distributional Robustness of Stochastic and Nominal Model Predictive Control," *IEEE Trans. Automat. Control*, vol. 69, no. 2, 2024.
- [99] P. O. M. Scokaert, D. Q. Mayne, and J. Rawlings, "Suboptimal Model Predictive Control (Feasibility Implies Stability)," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 648–654, 1999.
- [100] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder, "Nominal Stability of the Real-Time Iteration Scheme for Nonlinear Model Predictive Control," *IEE Proc.-Control Theory Appl.*, vol. 152, no. 3, pp. 296–308, 2005, publisher: IEE.
- [101] G. Pannocchia, J. Rawlings, and S. Wright, "Conditions under which suboptimal nonlinear MPC is inherently robust," *System & Control Letters*, vol. 60, no. 9, pp. 747–755, 2011.
- [102] D. A. Allan, C. N. Bates, M. J. Risbeck, and J. B. Rawlings, "On the inherent robustness of optimal and suboptimal nonlinear MPC," *Systems & Control Letters*, vol. 106, pp. 68–78, 2017.
- [103] J. L. Piovesan and H. G. Tanner, "Randomized model predictive control for robot navigation," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 94–99.
- [104] R. Y. Rubinstein, "Optimization of computer simulation models with rare events," *European Journal of Operational Research*, vol. 99, no. 1, pp. 89–112, 1997, publisher: Elsevier.
- [105] M. Kobilarov, "Cross-entropy motion planning," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 855–871, 2012, publisher: SAGE Publications Sage UK: London, England.
- [106] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.
- [107] H. J. Kappen, "Linear Theory for Control of Nonlinear Stochastic Systems," *Physical Review Letters*, vol. 95, no. 20, p. 200201, Nov. 2005, publisher: American Physical Society.
- [108] E. A. Theodorou and E. Todorov, "Relative entropy and free energy dualities: Connections to Path Integral and KL control," in *IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec. 2012, pp. 1466–1473, iSSN: 0743-1546.
- [109] E. A. Theodorou, "Nonlinear Stochastic Control and Information Theoretic Dualities: Connections, Interdependencies and Thermodynamic Interpretations," *Entropy*, vol. 17, no. 5, pp. 3352–3375, May 2015, number 5 Publisher: Multidisciplinary Digital Publishing Institute.
- [110] G. Williams, A. Aldrich, and E. A. Theodorou, "Model Predictive Path Integral Control: From Theory to Parallel Computation," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 2, pp. 344–357, 2017.
- [111] B. Goldfain, P. Drews, C. You, M. Barulic, O. Velev, P. Tsotras, and J. M. Rehg, "AutoRally: An Open Platform for Aggressive Autonomous Driving," *IEEE Control Systems Magazine*, vol. 39, no. 1, pp. 26–55, Feb. 2019, conference Name: IEEE Control Systems Magazine.
- [112] A. Bou, M. Bettini, S. Dittert, V. Kumar, S. Sodhani, X. Yang, G. D. Fabritiis, and V. Moens, "TorchRL: A data-driven decision-making library for PyTorch," Nov. 2023, arXiv:2306.00577 [cs].
- [113] B. Vlahov, J. Gibson, M. Gandhi, and E. A. Theodorou, "MPPI-Generic: A CUDA Library for Stochastic Optimization," Sep. 2024.
- [114] N. Kantas, J. Maciejowski, and A. Leccolini-Visintini, "Sequential Monte Carlo for model predictive control," *Nonlinear model predictive control: Towards new challenging applications*, pp. 263–273, 2009, publisher: Springer.
- [115] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009, publisher: Univelt, Inc.
- [116] T. Binder, L. Blank, H. G. Bock, R. Bulirsch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J. P. Schlöder, and O. v. Stryk, "Introduction to Model Based Optimization of Chemical Processes on Moving Horizons," in *Online Optimization of Large Scale Systems: State of the Art*, M. Grötschel, S. O. Krumke, and J. Rambau, Eds. Springer, 2001, pp. 295–340. [Online]. Available: <http://www.kuleuven.be/optec/OLD/research/subgroups/fastMPC/publications/Binder2001.php>
- [117] H. G. Bock and K. J. Plitt, "A Multiple Shooting Algorithm for Direct Solution of Optimal Control Problems," in *Proceedings of the IFAC World Congress*. Pergamon Press, 1984, pp. 242–247.
- [118] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control," *IEEE Transactions on Automatic Control*, vol. 59, no. 1, pp. 18–33, 2013, publisher: IEEE.
- [119] J. L. Jerez, P. J. Goulart, S. Richter, G. A. Constantinides, E. C. Kerrigan, and M. Morari, "Embedded online optimization for model predictive control at megahertz rates," *IEEE Transactions on Automatic Control*, vol. 59, no. 12, pp. 3238–3251, 2014, publisher: IEEE.
- [120] B. Käpernick and K. Graichen, "The gradient based nonlinear model predictive control software GRAMPC," in *2014 European Control Conference (ECC)*. IEEE, 2014, pp. 1170–1175.
- [121] T. Englert, A. Völz, F. Mesmer, S. Rhein, and K. Graichen, "A software framework for embedded nonlinear model predictive control using a gradient-based augmented Lagrangian approach (GRAMPC)," *Optimization and Engineering*, vol. 20, no. 3, pp. 769–809, Sep. 2019.
- [122] J. Kang, A. U. Raghunathan, and S. Di Cairano, "Decomposition via ADMM for scenario-based model predictive control," in *American Control Conference (ACC)*. IEEE, 2015, pp. 1246–1251.
- [123] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient Numerical Methods for Nonlinear MPC and Moving Horizon Estimation," in *Nonlinear model predictive control*, ser. Lecture Notes in Control and Information Sciences, L. Magni, M. D. Raimondo, and F. Allgöwer, Eds. Springer, 2009, vol. 384, pp. 391–417.
- [124] C. V. Rao, S. J. Wright, and J. B. Rawlings, "Application of Interior-Point Methods to Model Predictive Control," *Journal of Optimization Theory and Applications*, vol. 99, pp. 723–757, 1998.
- [125] J. Frey, S. D. Cairano, and R. Quirynen, "Active-Set based Inexact Interior Point QP Solver for Model Predictive Control," in *Proceedings of the IFAC World Congress*, 2020.
- [126] G. Frison and M. Diehl, "HPIPM: a high-performance quadratic programming framework for model predictive control**This research was supported by the German Federal Ministry for Economic Affairs and Energy (BMWi) via eco4wind (0324125B) and DyConPV (0324166B), and by DFG via Research Unit FOR 2401." *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6563–6569, Jan. 2020.
- [127] L. Vanroye, A. Sathya, J. De Schutter, and W. Decré, "Fatrop: A fast constrained optimal control problem solver for robot trajectory optimization and control," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 10 036–10 043.
- [128] D. Kouzoupis, G. Frison, A. Zanelli, and M. Diehl, "Recent advances in quadratic programming algorithms for nonlinear model predictive control," *Vietnam Journal of Mathematics*, vol. 46, no. 4, pp. 863–882, 2018.
- [129] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO toolkit—An open-source framework for automatic control and dynamic optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/oca.939>.
- [130] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. v. Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados—a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, vol. 14, no. 1, pp. 147–183, Mar. 2022.

- [131] M. Diehl, *Real-Time Optimization for Large Scale Nonlinear Processes*, ser. Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik. Düsseldorf: VDI Verlag, 2002, vol. 920.
- [132] H. G. Bock, M. Diehl, E. A. Kostina, and J. P. Schlöder, “Constrained Optimal Feedback Control of Systems Governed by Large Differential Algebraic Equations,” in *Real-Time and Online PDE-Constrained Optimization*. SIAM, 2007, pp. 3–22.
- [133] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, “From Linear to Nonlinear MPC: bridging the gap via the Real-Time Iteration,” *International Journal of Control*, 2016.
- [134] C. F. Gauss, *Theoria motus corporum coelestium in sectionibus conicis solem ambientium*. Perthes et Besser, 1809, vol. 7.
- [135] H. G. Bock, “Recent Advances in Parameter Identification Techniques for ODE,” in *Numerical Treatment of Inverse Problems in Differential and Integral Equations*. Birkhäuser, 1983, pp. 95–121.
- [136] F. Messerer, K. Baumgärtner, and M. Diehl, “Survey of Sequential Convex Programming and Generalized Gauss-Newton Methods,” *ESAIM: Proceedings and Surveys*, vol. 71, pp. 64–88, 2021.
- [137] D. Mayne, “A Second-order Gradient Method for Determining Optimal Trajectories of Non-linear Discrete-time Systems,” *Int. J. Control*, vol. 3, no. 1, pp. 85–96, 1966.
- [138] Y. Tassa, N. Mansard, and E. Todorov, “Control-Limited Differential Dynamic Programming,” in *IEEE International Conference on Robotics and Automation*, 2014.
- [139] W. Li and E. Todorov, “Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems,” in *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics*, 2004.
- [140] E. Todorov and W. Li, “A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems,” in *Proceedings of the American Control Conference (ACC)*, 2005.
- [141] J. Martí-Sauvall, J. Solà, C. Mastalli, and A. Santamaría-Navarro, “Squash-box feasibility driven differential dynamic programming,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 7637–7644.
- [142] A. Sideris and J. Bodrow, “An efficient sequential linear quadratic algorithm for solving unconstrained nonlinear optimal control problems,” *IEEE Transactions on Automatic Control*, vol. 50, no. 12, pp. 2043–2047, 2005.
- [143] K. Baumgärtner, F. Messerer, and M. Diehl, “A Unified Local Convergence Analysis of Differential Dynamic Programming, Direct Single Shooting, and Direct Multiple Shooting,” in *Proceedings of the European Control Conference (ECC)*, 2023.
- [144] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit solution of model predictive control via multiparametric quadratic programming,” in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*, vol. 2. IEEE, 2000, pp. 872–876.
- [145] A. Grancharova and T. A. Johansen, *Explicit Nonlinear Model Predictive Control: Theory and Applications*, ser. Lecture Notes in Control and Information Sciences. Berlin, Heidelberg: Springer, 2012, vol. 429.
- [146] M. C. Steinbach, “Tree-Sparse Convex Programs,” *Mathematical Methods of Operations Research*, vol. 56, no. 3, pp. 347–376, 2002.
- [147] E. Klintberg, J. Dahl, J. Fredriksson, and S. Gros, “An improved dual Newton strategy for scenario-tree MPC,” in *CDC*, 2016, pp. 3675–3681.
- [148] G. Frison, D. Kouzoupis, M. Diehl, and J. B. Jørgensen, “A high-performance Riccati based solver for tree-structured quadratic programs,” in *IFAC*, vol. 50, 2017, pp. 14 399–14 405, issue: 1.
- [149] D. Kouzoupis, “Structure-exploiting numerical methods for tree-sparse optimal control problems,” PhD Thesis, University of Freiburg, 2019.
- [150] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel, “Scaling up Gaussian Belief Space Planning Through Covariance-Free Trajectory Optimization and Automatic Differentiation,” in *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, H. L. Akin, N. M. Amato, V. Isler, and A. F. van der Stappen, Eds. Cham: Springer International Publishing, 2015, pp. 515–533.
- [151] X. Feng, S. Di Cairano, and R. Quirynen, “Inexact adjoint-based SQP algorithm for real-time stochastic nonlinear MPC,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 6529–6535, 2020, publisher: Elsevier.
- [152] A. Zanelli, J. Frey, F. Messerer, and M. Diehl, “Zero-Order Robust Nonlinear Model Predictive Control with Ellipsoidal Uncertainty Sets,” *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2021.
- [153] J. Frey, Y. Gao, F. Messerer, A. Lahr, M. N. Zeilinger, and M. Diehl, “Efficient Zero-Order Robust Optimization for Real-Time Model Predictive Control with acados,” in *Proceedings of the European Control Conference (ECC)*, 2024.
- [154] P. J. Goulart, E. C. Kerrigan, and D. Ralph, “Efficient robust optimization for robust control with constraints,” *Mathematical Programming*, vol. 114, no. 1, pp. 115–147, 2008, publisher: Springer.
- [155] A. P. Leeman, J. Köhler, F. Messerer, A. Lahr, M. Diehl, and M. N. Zeilinger, “Fast System Level Synthesis: Robust Model Predictive Control using Riccati Recursions,” in *Proceedings of the IFAC Conference on Nonlinear Model Predictive Control (NMPC)*, 2024.
- [156] H.-J. Yoon, C. Tao, H. Kim, N. Hovakimyan, and P. Voulgaris, “Sampling Complexity of Path Integral Methods for Trajectory Optimization,” in *2022 American Control Conference (ACC)*, 2022, pp. 3482–3487.
- [157] M. Morari and J. H. Lee, “Model predictive control: past, present and future,” *Computers & Chemical Engineering*, vol. 23, no. 4, pp. 667–682, 1999.
- [158] D. Angeli, R. Amrit, and J. B. Rawlings, “On Average Performance and Stability of Economic Model Predictive Control,” *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1615–1626, 2012.
- [159] B. Bakker, “Reinforcement Learning with Long Short-Term Memory,” in *Advances in Neural Information Processing Systems*, vol. 14. MIT Press, 2001.
- [160] J. Coulson, J. Lygeros, and F. Dörfler, “Regularized and Distributionally Robust Data-Enabled Predictive Control,” in *IEEE 58th Conference on Decision and Control (CDC)*, Dec. 2019, pp. 2696–2701, iSSN: 2576-2370.
- [161] G. Ceusters, R. C. Rodríguez, A. B. García, R. Franke, G. Deconinck, L. Helsen, A. Nowé, M. Messagie, and L. R. Camargo, “Model-predictive control and reinforcement learning in multi-energy system case studies,” *Applied Energy*, vol. 303, p. 117634, Dec. 2021.
- [162] S. Mamedov, R. Reiter, S. M. B. Azad, R. Viljoen, J. Boedecker, M. Diehl, and J. Swevers, “Safe Imitation Learning of Nonlinear Model Predictive Control for Flexible Robots,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 3613–3619, iSSN: 2153-0866.
- [163] S. Brandi, M. Fiorentini, and A. Capozzoli, “Comparison of online and offline deep reinforcement learning with model predictive control for thermal energy management,” *Automation in Construction*, vol. 135, p. 104128, Mar. 2022.
- [164] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, “Champion-level drone racing using deep reinforcement learning,” *Nature*, vol. 620, no. 7976, pp. 982–987, Aug. 2023, number: 7976 Publisher: Nature Publishing Group.
- [165] G. Lambrechts, A. Bolland, and D. Ernst, “Recurrent networks, hidden states and beliefs in partially observable environments,” *Transactions on Machine Learning Research*, May 2022.
- [166] L. Simpson, A. Ghezzi, J. Asprion, and M. Diehl, “An Efficient Method for the Joint Estimation of System Parameters and Noise Covariances for Linear Time-Variant Systems,” in *62nd IEEE Conference on Decision and Control (CDC)*, 2023, pp. 4524–4529.
- [167] L. Hewing, J. Kabzan, and M. N. Zeilinger, “Cautious Model Predictive Control Using Gaussian Process Regression,” *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020, conference Name: IEEE Transactions on Control Systems Technology.
- [168] B. Singh, R. Kumar, and V. Singh, “Reinforcement learning in robotic applications: a comprehensive survey,” *Artificial Intelligence Review*, vol. 55, Feb. 2022.
- [169] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, “Challenges of real-world reinforcement learning: definitions, benchmarks and analysis,” *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, Sep. 2021.
- [170] L. Smith, I. Kostrikov, and S. Levine, “Demonstrating a walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning,” *Robotics: Science and Systems (RSS) Demo*, vol. 2, no. 3, p. 4, 2023.
- [171] C. Tallec, L. Blier, and Y. Ollivier, “Making deep q-learning methods robust to time discretization,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 6096–6104.
- [172] L. Ljung, *System Identification: Theory for the User*. Prentice Hall PTR, 1999, google-Books-ID: nHFoQgAACAAJ.
- [173] J. Frank, S. Mannor, and D. Precup, “Reinforcement learning in the presence of rare events,” in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 336–343.
- [174] J. Morimoto and K. Doya, “Robust Reinforcement Learning,” in *Advances in Neural Information Processing Systems*, vol. 13. MIT Press, 2000.
- [175] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, “Robust Reinforcement Learning: A Review of Foundations and Recent Advances,” *Machine Learning and Knowledge Extraction*, vol. 4, no. 1,

- pp. 276–315, Mar. 2022, number: 1 Publisher: Multidisciplinary Digital Publishing Institute.
- [176] C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.
- [177] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, and A. Askell, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [178] B. P. G. Van Parys, D. Kuhn, P. J. Goulart, and M. Morari, “Distributionally Robust Control of Constrained Stochastic Systems,” *IEEE Transactions on Automatic Control*, vol. 61, no. 2, pp. 430–442, 2016.
- [179] S. Gros and M. Zanon, “Economic MPC of Markov Decision Processes: Dissipativity in undiscounted infinite-horizon optimal control,” *Automatica*, vol. 146, p. 110602, Dec. 2022.
- [180] E. Altman, *Constrained Markov Decision Processes*, 1st, Ed. New York: Routledge, 1999.
- [181] G. Kalweit, M. Huegle, M. Werling, and J. Boedecker, “Deep Inverse Q-learning with Constraints,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 14291–14302.
- [182] A. Ray, J. Achiam, and D. Amodei, “Benchmarking Safe Exploration in Deep Reinforcement Learning,” 2019. [Online]. Available: <https://cdn.openai.com/safexp-short.pdf>
- [183] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained Policy Optimization,” in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 22–31, iSSN: 2640-3498.
- [184] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, 2019, pp. 3387–3395, issue: 01.
- [185] K. P. Wabersich and M. N. Zeilinger, “A predictive safety filter for learning-based control of constrained nonlinear dynamical systems,” *Automatica*, vol. 129, p. 109597, Jul. 2021.
- [186] M. Calvo-Fullana, S. Paternain, L. F. O. Chamon, and A. Ribeiro, “State Augmented Constrained Reinforcement Learning: Overcoming the Limitations of Learning With Rewards,” *IEEE Transactions on Automatic Control*, vol. 69, no. 7, pp. 4275–4290, Jul. 2024, conference Name: IEEE Transactions on Automatic Control.
- [187] A. Sootla, A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mguni, J. Wang, and H. Ammar, “Saute RL: Almost Surely Safe Reinforcement Learning Using State Augmentation,” in *Proceedings of the 39th International Conference on Machine Learning*. PMLR, Jun. 2022, pp. 20423–20443, iSSN: 2640-3498.
- [188] B. Zhang, Y. Zhang, L. Frison, T. Brox, and J. Bödecker, “Constrained Reinforcement Learning with Smoothed Log Barrier Function,” Mar. 2024, arXiv:2403.14508 [cs].
- [189] S. Richter, C. N. Jones, and M. Morari, “Computational complexity certification for real-time MPC with input constraints based on the fast gradient method,” *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1391–1403, 2011, publisher: IEEE.
- [190] D. Arnström, “Real-Time Certified MPC : Reliable Active-Set QP Solvers,” 2023, publisher: Linköping University Electronic Press.
- [191] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, F. Hutter, and R. Calandra, “On the importance of hyperparameter optimization for model-based reinforcement learning,” in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021, pp. 4015–4023.
- [192] L. Nasvytis, K. Sandbrink, J. Foerster, T. Franzmeyer, and C. Schroeder de Witt, “Rethinking Out-of-Distribution Detection for Reinforcement Learning: Advancing Methods for Evaluation and Detection,” in *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024, pp. 1445–1453.
- [193] S. Ross, G. Gordon, and D. Bagnell, “A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning,” in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, Apr. 2011, pp. 627–635.
- [194] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan, “Rapidly adaptable legged robots via evolutionary meta-learning,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 3769–3776.
- [195] Gurobi Optimization, LLC, “Gurobi Optimizer Reference Manual,” 2023. [Online]. Available: <https://www.gurobi.com>
- [196] T. H. Oh, “Quantitative comparison of reinforcement learning and data-driven model predictive control for chemical and biological processes,” *Computers & Chemical Engineering*, vol. 181, p. 108558, Feb. 2024.
- [197] D. Ernst, M. Glavic, F. Capitanescu, and L. Wehenkel, “Reinforcement Learning Versus Model Predictive Control: A Comparison on a Power System Problem,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, no. 2, pp. 517–529, Apr. 2009, conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics).
- [198] D. Wang, W. Zheng, Z. Wang, Y. Wang, X. Pang, and W. Wang, “Comparison of reinforcement learning and model predictive control for building energy system optimization,” *Applied Thermal Engineering*, vol. 228, p. 120430, Jun. 2023.
- [199] A. Hasankhani, Y. Tang, J. VanZwieten, and C. Sultan, “Comparison of Deep Reinforcement Learning and Model Predictive Control for Real-Time Depth Optimization of a Lifting Surface Controlled Ocean Current Turbine,” in *IEEE Conference on Control Technology and Applications (CCTA)*, Aug. 2021, pp. 301–308, iSSN: 2768-0770.
- [200] D. Ernst, P. Geurts, and L. Wehenkel, “Iteratively Extending Time Horizon Reinforcement Learning,” in *Machine Learning: ECML 2003*, ser. Lecture Notes in Computer Science, N. Lavrač, D. Gamberger, H. Blockeel, and L. Todorovski, Eds. Berlin, Heidelberg: Springer, 2003, pp. 96–107.
- [201] I. I. Cplex, “V12. 1: User’s Manual for CPLEX,” *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.
- [202] Y. Lin, J. McPhee, and N. L. Azad, “Comparison of Deep Reinforcement Learning and Model Predictive Control for Adaptive Cruise Control,” *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 221–231, Jun. 2021, conference Name: IEEE Transactions on Intelligent Vehicles.
- [203] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006.
- [204] L. Di Natale, Y. Lian, E. T. Maddalena, J. Shi, and C. N. Jones, “Lessons Learned from Data-Driven Building Control Experiments: Contrasting Gaussian Process-based MPC, Bilevel DeePC, and Deep Reinforcement Learning,” in *IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 1111–1117, iSSN: 2576-2370.
- [205] J. Coulson, J. Lygeros, and F. Dörfler, “Data-Enabled Predictive Control: In the Shallows of the DeepPC,” in *18th European Control Conference (ECC)*, Jun. 2019, pp. 307–312.
- [206] D. Dobriborsci, P. Osinenko, and W. Aumer, “An experimental study of two predictive reinforcement learning methods and comparison with model-predictive control,” *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 1545–1550, Jan. 2022.
- [207] “SciPy v1.15.0 Manual.” [Online]. Available: <https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html>
- [208] A. Byravan, L. Hasenclever, P. Trochim, M. Mirza, A. D. Ialongo, J. T. Tassa, Y. andSpringenberg, A. Abdolmaleki, N. Heess, J. Merel, and M. Riedmiller, “Evaluating Model-Based Planning and Planner Amortization for Continuous Control,” in *10th International Conference on Learning Representations, ICLR*, Apr. 2022.
- [209] A. Piche, V. Thomas, C. Ibrahim, Y. Bengio, and C. Pal, “Probabilistic Planning with Sequential Monte Carlo methods,” Sep. 2018.
- [210] A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller, “Maximum a Posteriori Policy Optimisation,” in *International Conference on Learning Representations*, 2018.
- [211] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza, “Reaching the limit in autonomous racing: Optimal control versus reinforcement learning,” *Science Robotics*, vol. 8, no. 82, p. eadg1462, Sep. 2023, publisher: American Association for the Advancement of Science.
- [212] J. Shi, K. Li, C. Piao, J. Gao, and L. Chen, “Model-Based Predictive Control and Reinforcement Learning for Planning Vehicle-Parking Trajectories for Vertical Parking Spaces,” *Sensors*, vol. 23, no. 16, p. 7124, Jan. 2023, number: 16 Publisher: Multidisciplinary Digital Publishing Institute.
- [213] M. Imran, R. Izzo, A. Tortorelli, and F. Liberati, “Comparison of Traffic Control with Model Predictive Control and Deep Reinforcement Learning,” in *9th International Conference on Control, Decision and Information Technologies (CoDIT)*, Jul. 2023, pp. 989–994, iSSN: 2576-3555.
- [214] R. Reiter, J. Hoffmann, J. Boedecker, and M. Diehl, “A Hierarchical Approach for Strategic Motion Planning in Autonomous Racing,” in *European Control Conference (ECC)*, Jun. 2023, pp. 1–8.
- [215] B. Morcego, W. Yin, S. Boersma, E. van Henten, V. Puig, and C. Sun, “Reinforcement Learning versus Model Predictive Control on greenhouse

- climate control," *Computers and Electronics in Agriculture*, vol. 215, p. 108372, Dec. 2023.
- [216] P. Hoffmann, K. Gorelik, and V. Ivanov, "Comparison of Reinforcement Learning and Model Predictive Control for Automated Generation of Optimal Control for Dynamic Systems within a Design Space Exploration Framework," *International Journal of Automotive Engineering*, vol. 15, no. 1, pp. 19–26, 2024.
- [217] S. Levine and V. Koltun, "Guided Policy Search," in *Proceedings of the 30th International Conference on Machine Learning*. PMLR, May 2013, pp. 1–9, iSSN: 1938-7228.
- [218] F. Jenelten, J. He, F. Farshidian, and M. Hutter, "DTC: Deep Tracking Control," *Science Robotics*, vol. 9, no. 86, p. eadh5401, Jan. 2024, publisher: American Association for the Advancement of Science.
- [219] A. Ghezzi, J. Hoffman, J. Frey, J. Boedecker, and M. Diehl, "Imitation Learning from Nonlinear MPC via the Exact Q-Loss and its Gauss-Newton Approximation," in *62nd IEEE Conference on Decision and Control (CDC)*, Dec. 2023, pp. 4766–4771, iSSN: 2576-2370.
- [220] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan Online, Learn Offline: Efficient Learning and Exploration via Model-Based Control," in *7th International Conference on Learning Representations*, 2019.
- [221] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go Next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4616–4623, Jul. 2021.
- [222] M. Jacquet and K. Alexis, "N-MPC for Deep Neural Network-Based Collision Avoidance exploiting Depth Images," Feb. 2024, arXiv:2402.13038 [cs].
- [223] M. Bhardwaj, S. Choudhury, and B. Boots, "Blending MPC & Value Function Approximation for Efficient Reinforcement Learning," in *9th International Conference on Learning Representations, ICLR*, 2021.
- [224] S. Gros and M. Zanon, "Data-Driven Economic NMPC Using Reinforcement Learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 2, pp. 636–648, Feb. 2020.
- [225] H. Moradimayamnegari, M. Frego, and A. Peer, "Model Predictive Control-Based Reinforcement Learning Using Expected Sarsa," *IEEE Access*, vol. 10, pp. 81 177–81 191, 2022, conference Name: IEEE Access.
- [226] X. Liu, L. Peters, and J. Alonso-Mora, "Learning to Play Trajectory Games Against Opponents With Unknown Objectives," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4139–4146, Jul. 2023, conference Name: IEEE Robotics and Automation Letters.
- [227] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros, "RL + Model-Based Control: Using On-Demand Optimal Control to Learn Versatile Legged Locomotion," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6619–6626, Oct. 2023.
- [228] S. Levine and V. Koltun, "Variational Policy Search via Trajectory Optimization," in *Advances in Neural Information Processing Systems*, vol. 26. Curran Associates, Inc., 2013.
- [229] G. Neumann, "Variational inference for policy search in changing situations," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*. Bellevue, Washington, USA: Omnipress, 2011, pp. 817–824.
- [230] I. Mordatch and E. Todorov, "Combining the benefits of function approximation and trajectory optimization," vol. 10, Jul. 2014.
- [231] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016, ISBN: 1533-7928.
- [232] H. Wang and A. Banerjee, "Bregman Alternating Direction Method of Multipliers," in *Advances in Neural Information Processing Systems*, vol. 27. Curran Associates, Inc., 2014.
- [233] L. Sun, C. Peng, W. Zhan, and M. Tomizuka, "A Fast Integrated Planning and Control Framework for Autonomous Driving via Imitation Learning," American Society of Mechanical Engineers Digital Collection, Nov. 2018.
- [234] T. Wang and J. Ba, "Exploring Model-based Planning with Policy Networks," Sep. 2019.
- [235] C. Pinneri, S. Sawant, S. Blaes, and G. Martius, "Extracting Strong Policies for Robotics Tasks from Zero-Order Trajectory Optimizers," in *International Conference on Learning Representations*, 2021.
- [236] J. Sacks and B. Boots, "Learning to Optimize in Model Predictive Control," in *International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 10 549–10 556.
- [237] M. Dawood, N. Dengler, J. de Hevel, and M. Bennewitz, "Handling Sparse Rewards in Reinforcement Learning Using Model Predictive Control," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 879–885.
- [238] K. Ahn, Z. Mhammedi, H. Mania, Z.-W. Hong, and A. Jadbabae, "Model Predictive Control via On-Policy Imitation Learning," in *Proceedings of The 5th Annual Learning for Dynamics and Control Conference*. PMLR, Jun. 2023, pp. 1493–1505, iSSN: 2640-3498.
- [239] Q. L. Lidec, W. Jallet, I. Laptev, C. Schmid, and J. Carpenter, "Enforcing the consensus between Trajectory Optimization and Policy Learning for precise robot control," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 946–952.
- [240] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpenter, S. Vijayakumar, and N. Mansard, "Crocoddyl: An Efficient and Versatile Framework for Multi-Contact Optimal Control," in *ICRA 2020 IEEE International Conference on Robotics and Automation*, Paris / Virtual, France, May 2020.
- [241] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," Aug. 2018.
- [242] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," in *Advances in Neural Information Processing Systems*, vol. 29. Curran Associates, Inc., 2016.
- [243] J. Hoffmann, D. F. Clausen, J. Brosseit, J. Bernhard, K. Esterle, M. Werling, M. Karg, and J. J. Bödecker, "PlanNetX: Learning an efficient neural network planner from MPC for longitudinal control," in *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*. PMLR, Jun. 2024, pp. 1214–1227, iSSN: 2640-3498.
- [244] F. Schulz, J. Hoffmann, Y. Zhang, and J. Boedecker, "Learning When to Trust the Expert for Guided Exploration in RL," in *ICML 2024 Workshop: Foundations of Reinforcement Learning and Control – Connections and Perspectives*, 2024.
- [245] A. Bemporad, F. Borrelli, and M. Morari, "The explicit solution of constrained LP-Based receding horizon control," in *Proceedings of the IEEE conference on decision and control (CDC)*, Sydney, Australia, 1999.
- [246] —, "Model predictive control based on linear programming - the explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, Dec. 2002, conference Name: IEEE Transactions on Automatic Control.
- [247] T. Johansen and A. Granchiarova, "Approximate explicit constrained linear model predictive control via orthogonal search tree," *IEEE Trans. Automatic Control*, vol. 48, pp. 810–815, 2003.
- [248] Y. Vaupel, N. C. Hamacher, A. Caspari, A. Mhamdi, I. G. Kevrekidis, and A. Mitsos, "Accelerating nonlinear model predictive control through machine learning," *Journal of Process Control*, vol. 92, pp. 261–270, Aug. 2020.
- [249] M. Hertneck, J. Köhler, S. Trimpe, and F. Allgöwer, "Learning an Approximate Model Predictive Controller With Guarantees," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 543–548, Jul. 2018, conference Name: IEEE Control Systems Letters.
- [250] B. M. Åkesson and H. T. Toivonen, "A neural network model predictive controller," *Journal of Process Control*, vol. 16, no. 9, pp. 937–946, Oct. 2006.
- [251] Y. Cao and R. B. Gopaluni, "Deep Neural Network Approximation of Nonlinear Model Predictive Control," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 11 319–11 324, Jan. 2020.
- [252] "A Neural Network Architecture to Learn Explicit MPC Controllers from Data," *Ifac Papersonline*, 2020.
- [253] S. Chen, K. Saulnier, N. Atanasov, D. D. Lee, V. Kumar, G. J. Pappas, and M. Morari, "Approximating Explicit Model Predictive Control Using Constrained Neural Networks," in *2018 Annual American Control Conference (ACC)*, Jun. 2018, pp. 1520–1527.
- [254] B. Karg and S. Lucia, "Efficient Representation and Approximation of Model Predictive Control Laws via Deep Learning," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3866–3878, Sep. 2020.
- [255] R. K. Cosner, Y. Yue, and A. D. Ames, "End-to-End Imitation Learning with Safety Guarantees using Control Barrier Functions," in *2022 IEEE 61st Conference on Decision and Control (CDC)*. Cancun, Mexico: IEEE, Dec. 2022, pp. 5316–5322.
- [256] R. Schwan, C. N. Jones, and D. Kuhn, "Stability Verification of Neural Network Controllers Using Mixed-Integer Programming," *IEEE Transactions on Automatic Control*, pp. 1–16, 2023, conference Name: IEEE Transactions on Automatic Control.
- [257] J. Drgoňa, A. Tuor, and D. Vrabie, "Learning Constrained Parametric Differentiable Predictive Control Policies With Guarantees," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–12, 2024.
- [258] M. Everett, "Neural Network Verification in Control," in *2021 60th IEEE Conference on Decision and Control (CDC)*. Austin, TX, USA: IEEE, Dec. 2021, pp. 6326–6340.

- [259] J. Carius, F. Farshidian, and M. Hutter, "MPC-Net: A First Principles Guided Policy Search," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2897–2904, Apr. 2020.
- [260] A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter, "Imitation Learning from MPC for Quadrupedal Multi-Gait Control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 5014–5020.
- [261] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999, publisher: Elsevier.
- [262] A. B. Kordabab, M. Zanon, and S. Gros, "Equivalence of Optimality Criteria for Markov Decision Process and Model Predictive Control," *IEEE Transactions on Automatic Control*, vol. 69, no. 2, pp. 1149–1156, Feb. 2024, conference Name: IEEE Transactions on Automatic Control.
- [263] R. Reiter, A. Ghezzi, K. Baumgärtner, J. Hoffmann, R. D. McAllister, and M. Diehl, "AC4MPC: Actor-Critic Reinforcement Learning for Nonlinear Model Predictive Control," Jun. 2024.
- [264] M. Zhong, M. Johnson, Y. Tassa, T. Erez, and E. Todorov, "Value function approximation and model predictive control," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, Apr. 2013, pp. 100–107, iSSN: 2325-1867.
- [265] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.
- [266] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 979–1006, 2002.
- [267] R. Y. Rubinstein and D. P. Kroese, *The Cross-Entropy Method*, ser. Information Science and Statistics, M. Jordan, J. Kleinberg, B. Schölkopf, F. P. Kelly, and I. Witten, Eds. New York, NY: Springer, 2004.
- [268] R. Deits, T. Koolen, and R. Tedrake, "LVIS: Learning from Value Function Intervals for Contact-Aware Robot Controllers," in *International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE Press, 2019, pp. 7762–7768.
- [269] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani, "Data Efficient Reinforcement Learning for Legged Robots," in *Proceedings of the Conference on Robot Learning*. PMLR, May 2020, pp. 1–10, iSSN: 2640-3498.
- [270] G. Kenneally, A. De, and D. E. Koditschek, "Design Principles for a Family of Direct-Drive Legged Robots," *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 900–907, Jul. 2016.
- [271] N. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. J. Pister, "Low-Level Control of a Quadrotor With Deep Model-Based Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 4, pp. 4224–4230, 2019.
- [272] N. Karnchanachari, M. I. Valls, D. Hoeller, and M. Hutter, "Practical Reinforcement Learning For MPC: Learning from sparse objectives in under an hour on a real robot," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. PMLR, Jul. 2020, pp. 211–224, iSSN: 2640-3498.
- [273] L. Beckenbach, P. Osinenko, and S. Streif, "A Q-learning predictive control scheme with guaranteed stability," *European Journal of Control*, vol. 56, pp. 167–178, Nov. 2020.
- [274] D. Hoeller, F. Farshidian, and M. Hutter, "Deep Value Model Predictive Control," in *Proceedings of the Conference on Robot Learning*. PMLR, May 2020, pp. 990–1004, iSSN: 2640-3498.
- [275] N. Hatch and B. Boots, "The Value of Planning for Infinite-Horizon Model Predictive Control," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, May 2021, pp. 7372–7378, iSSN: 2577-087X.
- [276] A. S. Morgan, D. Nandha, G. Chalvatzaki, C. D'Eramo, A. M. Dollar, and J. Peters, "Model Predictive Actor-Critic: Accelerating Robot Skill Acquisition with Deep Reinforcement Learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE Press, 2021, pp. 6672–6678.
- [277] L. Beckenbach and S. Streif, "Approximate infinite-horizon predictive control," in *IEEE 61st Conference on Decision and Control (CDC)*, Dec. 2022, pp. 3711–3717, iSSN: 2576-2370.
- [278] 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, Jan. 2023.
- [279] T. M. Inc, "MATLAB version: 9.13.0 (R2022b)," Natick, Massachusetts, United States, 2022. [Online]. Available: <https://www.mathworks.com>
- [280] M. Lin, Z. Sun, Y. Xia, and J. Zhang, "Reinforcement Learning-Based Model Predictive Control for Discrete-Time Systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 3, pp. 3312–3324, Mar. 2024, conference Name: IEEE Transactions on Neural Networks and Learning Systems.
- [281] Y. Qu, H. Chu, S. Gao, J. Guan, H. Yan, L. Xiao, S. E. Li, and J. Duan, "RL-Driven MPPI: Accelerating Online Control Laws Calculation With Offline Policy," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 2, pp. 3605–3616, Feb. 2024.
- [282] W. Cai, S. Sawant, D. Reinhardt, S. Rastegarpour, and S. Gros, "A Learning-Based Model Predictive Control Strategy for Home Energy Management Systems," *IEEE Access*, vol. 11, pp. 145 264–145 280, 2023.
- [283] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for End-to-end Planning and Control," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.
- [284] M. Zanon and S. Gros, "Safe Reinforcement Learning Using Robust MPC," *IEEE Transactions on Automatic Control*, vol. 66, no. 8, pp. 3638–3652, Aug. 2021, conference Name: IEEE Transactions on Automatic Control.
- [285] A. Romero, Y. Song, and D. Scaramuzza, "Actor-Critic Model Predictive Control," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. Yokohama, Japan: IEEE, May 2024, pp. 14 777–14 784.
- [286] M. Zanon, V. Kungurtsev, and S. Gros, "Reinforcement Learning Based on Real-Time Iteration NMPC," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5213–5218, Jan. 2020.
- [287] B. Zarrouki, C. Wang, and J. Betz, "Adaptive Stochastic Nonlinear Model Predictive Control with Look-ahead Deep Reinforcement Learning for Autonomous Vehicle Motion Control," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 12 726–12 733, iSSN: 2153-0866.
- [288] A. B. Kordabab and S. Gros, "Bias Correction of Discounted Optimal Steady-State using Cost Modification," in *2023 European Control Conference (ECC)*, Jun. 2023, pp. 1–6.
- [289] S. Gros and M. Zanon, "Towards Safe Reinforcement Learning Using NMPC and Policy Gradients: Part I - Stochastic case," Jun. 2019, arXiv:1906.04057 [cs].
- [290] T. Rashid, B. Peng, W. Böhmer, and S. Whiteson, "Optimistic Exploration even with a Pessimistic Initialisation," in *8th International Conference on Learning Representations, ICLR*, 2020.
- [291] C. Greatwood and A. G. Richards, "Reinforcement learning and model predictive control for robust embedded quadrotor guidance and control," *Autonomous Robots*, vol. 43, no. 7, pp. 1681–1693, Oct. 2019.
- [292] T. Tram, I. Batkovic, M. Ali, and J. Sjöberg, "Learning When to Drive in Intersections by Combining Reinforcement Learning and Model Predictive Control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 3263–3268.
- [293] A. Domahidi and J. Jerez, "FORCES Professional," 2014, published: Embotech AG, url=<https://embotech.com/FORCES-Pro>.
- [294] B. Brito, A. Agarwal, and J. Alonso-Mora, "Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 18 808–18 821, Oct. 2022, conference Name: IEEE Transactions on Intelligent Transportation Systems.
- [295] Z. Zhang, H. An, Q. Wei, and H. Ma, "Learning-Based Model Predictive Control for Quadruped Locomotion on Slippery Ground," in *4th International Conference on Control and Robotics (ICCR)*, Dec. 2022, pp. 47–52.
- [296] S. Pfommer, T. Gautam, A. Zhou, and S. Sojoudi, "Safe Reinforcement Learning with Chance-constrained Model Predictive Control," in *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*. PMLR, May 2022, pp. 291–303, iSSN: 2640-3498.
- [297] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.*, 2024. [Online]. Available: <http://docs.mosek.com/latest/toolbox/index.html>
- [298] R. Tao, S. Cheng, X. Wang, S. Wang, and N. Hovakimyan, "DiffTune-MPC: Closed-Loop Learning for Model Predictive Control," *IEEE Robotics and Automation Letters*, 2024, publisher: IEEE.
- [299] B. Zarrouki, M. Spanakakis, and J. Betz, "A Safe Reinforcement Learning driven Weights-varying Model Predictive Control for Autonomous Vehicle Motion Control: 35th IEEE Intelligent Vehicles Symposium, IV 2024," *35th IEEE Intelligent Vehicles Symposium, IV 2024*, pp. 1401–1408, 2024.
- [300] Z. Wen, M. Dong, and X. Chen, "Collision-Free Robot Navigation in Crowded Environments using Learning-based Convex Model Predictive Control," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 5452–5459, iSSN: 2153-0866.

- [301] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, “A Predictive Safety Filter for Learning-Based Racing Control,” *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, Oct. 2021.
- [302] X. Shen and F. Borrelli, “Reinforcement Learning and Distributed Model Predictive Control for Conflict Resolution in Highly Constrained Spaces,” in *IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2023, pp. 1–6, iSSN: 2642-7214.
- [303] K. Ceder, Z. Zhang, A. Burman, I. Kuangaliyev, K. Mattsson, G. Nyman, A. Petersén, L. Wisell, and K. Åkesson, “Bird’s-Eye-View Trajectory Planning of Multiple Robots using Continuous Deep Reinforcement Learning and Model Predictive Control,” in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 8002–8008, iSSN: 2153-0866.
- [304] 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 Robotics and Automation Letters*, vol. 8, no. 6, pp. 3318–3325, Jun. 2023.
- [305] S. Li and O. Bastani, “Robust Model Predictive Shielding for Safe Reinforcement Learning with Stochastic Dynamics,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 7166–7172, iSSN: 2577-087X.
- [306] A. Didier, R. C. Jacobs, J. Sieber, K. P. Wabersich, and M. N. Zeilinger, “Approximate Predictive Control Barrier Functions using Neural Networks: A Computationally Cheap and Permissive Safety Filter,” in *European Control Conference (ECC)*, Jun. 2023, pp. 1–7.
- [307] 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 *Proceedings of the 6th Annual Learning for Dynamics & Control Conference*. PMLR, Jun. 2024, pp. 1452–1463, iSSN: 2640-3498.
- [308] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, “A simple and efficient algorithm for nonlinear model predictive control,” in *56th IEEE Conference on Decision and Control (CDC)*, 2017, pp. 1939–1944.
- [309] F. Jenelten, R. Grandia, F. Farshidian, and M. Hutter, “TAMOLS: Terrain-Aware Motion Optimization for Legged Systems,” *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3395–3413, Dec. 2022, arXiv:2206.14049 [cs].
- [310] S. H. Bang, C. A. Jové, and L. Sentis, “RL-augmented MPC Framework for Agile and Robust Bipedal Footstep Locomotion Planning and Control,” Jul. 2024, arXiv:2407.17683 [cs].
- [311] K. Seel, A. B. Kordabad, S. Gros, and J. T. Gravdahl, “Convex Neural Network-Based Cost Modifications for Learning Model Predictive Control,” *IEEE Open Journal of Control Systems*, vol. 1, pp. 366–379, 2022, conference Name: IEEE Open Journal of Control Systems.
- [312] H. N. Esfahani, A. Bahari Kordabad, W. Cai, and S. Gros, “Learning-based state estimation and control using MHE and MPC schemes with imperfect models,” *European Journal of Control*, vol. 73, p. 100880, Sep. 2023.
- [313] S. Adhau, D. Reinhardt, S. Skogstad, and S. Gros, “Fast Reinforcement Learning Based MPC based on NLP Sensitivities,” *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 11 841–11 846, Jan. 2023.
- [314] H. Bharadhwaj, K. Xie, and F. Shkurti, “Model-predictive control via cross-entropy and gradient-based optimization,” in *Learning for Dynamics and Control*. PMLR, 2020, pp. 277–286.
- [315] A. S. Anand, D. Reinhardt, S. Sawant, J. T. Gravdahl, and S. Gros, “A Painless Deterministic Policy Gradient Method for Learning-based MPC,” in *European Control Conference (ECC)*, Jun. 2023, pp. 1–7.
- [316] S. Gros and M. Zanon, “Reinforcement Learning for mixed-integer problems based on MPC,” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 5219–5224, Jan. 2020.
- [317] A. B. Kordabad, H. N. Esfahani, A. M. Lekkas, and S. Gros, “Reinforcement Learning based on Scenario-tree MPC for ASVs,” in *2021 American Control Conference (ACC)*, May 2021, pp. 1985–1990, iSSN: 2378-5861.
- [318] A. S. Anand, S. Sawant, D. Reinhardt, and S. Gros, “All AI Models are Wrong, but Some are Optimal,” Jan. 2025, arXiv:2501.06086 [cs].
- [319] S. V. Sawant and S. N. Gros, “Bridging the gap between QP-based and MPC-based Reinforcement Learning,” *IFAC-PapersOnLine*, 2022.
- [320] S. Sawant, D. Reinhardt, A. B. Kordabad, and S. Gros, “Model-Free Data-Driven Predictive Control Using Reinforcement Learning,” in *62nd IEEE Conference on Decision and Control (CDC)*, Dec. 2023, pp. 4046–4052.
- [321] W. Favoreel, B. D. Moor, and M. Gevers, “SPC: Subspace Predictive Control,” *IFAC Proceedings Volumes*, vol. 32, no. 2, pp. 4004–4009, Jul. 1999.
- [322] H. N. Esfahani, A. B. Kordabad, and S. Gros, “Reinforcement learning based on MPC/MHE for unmodeled and partially observable dynamics,” in *American Control Conference (ACC)*. IEEE, 2021, pp. 2121–2126.
- [323] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, “Data-Driven Safety Filters: Hamilton-Jacobi Reachability, Control Barrier Functions, and Predictive Methods for Uncertain Systems,” *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, Oct. 2023, conference Name: IEEE Control Systems Magazine.
- [324] S. Gros, M. Zanon, and A. Bemporad, “Safe reinforcement learning via projection on a safe set: How to achieve optimality?” *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 8076–8081, 2020, publisher: Elsevier.
- [325] S. Gros and M. Zanon, “Learning for MPC with stability & safety guarantees,” *Automatica*, vol. 146, p. 110598, Dec. 2022.
- [326] A. B. Kordabad, R. Wisniewski, and S. Gros, “Safe Reinforcement Learning Using Wasserstein Distributionally Robust MPC and Chance Constraint,” *IEEE Access*, vol. 10, pp. 130 058–130 067, 2022, conference Name: IEEE Access.
- [327] A. B. Kordabad and S. Gros, “Verification of Dissipativity and Evaluation of Storage Function in Economic Nonlinear MPC using Q-Learning,” *IFAC-PapersOnLine*, vol. 54, no. 6, pp. 308–313, Jan. 2021.
- [328] M. Zanon, S. Gros, and M. Palladino, “Stability-constrained Markov Decision Processes using MPC,” *Automatica*, vol. 143, p. 110399, Sep. 2022.
- [329] M. Zanon, S. Gros, and A. Bemporad, “Practical reinforcement learning of stabilizing economic MPC,” in *18th European Control Conference (ECC)*. IEEE, 2019, pp. 2258–2263.
- [330] A. B. Kordabad, H. Nejatbakhsh Esfahani, and S. Gros, “Bias Correction in Deterministic Policy Gradient Using Robust MPC,” in *2021 European Control Conference (ECC)*, Jun. 2021, pp. 1086–1091.
- [331] K. Seel, S. Gros, and J. T. Gravdahl, “Combining Q-learning and Deterministic Policy Gradient for Learning-Based MPC,” in *2023 62nd IEEE Conference on Decision and Control (CDC)*, Dec. 2023, pp. 610–617, iSSN: 2576-2370.
- [332] Anand, Akhil S. Sawant, Shambhuraj, Reinhardt, Dirk, and Gros, Sébastien, “Data-Driven Predictive Control and MPC: Do we achieve optimality?” 2024.
- [333] D. Reinhardt, A. S. Anand, S. Sawant, and S. Gros, “Economic model predictive control as a solution to markov decision processes,” 2024.
- [334] E. Nikishin, R. Abachi, R. Agarwal, and P.-L. Bacon, “Control-oriented model-based reinforcement learning with implicit differentiation,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 7886–7894, issue: 7.
- [335] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, and T. Graepel, “Mastering atari, go, chess and shogi by planning with a learned model,” *Nature*, vol. 588, no. 7839, pp. 604–609, 2020, ISBN: 1476-4687 Publisher: Nature Publishing Group.
- [336] T. Salzmann, J. Arrizabalaga, J. Andersson, M. Pavone, and M. Ryall, “Learning for CasADI: Data-driven models in numerical optimization,” 2023, arXiv: 2312.05873 [eess.SY].
- [337] J. A. E. Andersson and J. B. Rawlings, “Sensitivity Analysis for Nonlinear Programming in CasADI*,” *IFAC-PapersOnLine*, vol. 51, no. 20, pp. 331–336, Jan. 2018.
- [338] A. Lahr, J. Náf, K. P. Wabersich, J. Frey, P. Siehl, A. Carron, M. Diehl, and M. N. Zeilinger, “L4acados: Learning-based models for acados, applied to Gaussian process-based predictive control,” Nov. 2024, arXiv: 2411.19258 tex.pubstate: prepublished.
- [339] S. Gros and M. Zanon, “Reinforcement Learning based on MPC and the Stochastic Policy Gradient Method,” in *American Control Conference (ACC)*, May 2021, pp. 1947–1952, iSSN: 2378-5861.

APPENDIX A DIFFERENTIATING THROUGH A PARAMETERIZED MPC LAYER

In the following, we discuss how to compute gradients concerning parameters θ within the architectures introduced in Fig. 3. The availability of gradient information is a crucial prerequisite for the learning approaches introduced in the following sections. We restrict the discussion to the parameterized

MPC formulation of Fig. 3, where the corresponding solution map is defined by

$$z^*(s, \theta) := \underset{z}{\operatorname{argmin}} L(z; \theta) \text{ s.t. } g(z; s, \theta) \geq 0. \quad (31)$$

Assuming that the objective is twice, the constraints once, continuously differentiable, the implicit function theorem (IFT) guarantees the existence of the solution sensitivities $\frac{\partial z^*}{\partial \theta}(\bar{s}, \bar{\theta})$ at $(\bar{s}, \bar{\theta})$ if the linear independence constraint qualification, second-order sufficient conditions, and strict complementarity are satisfied at the solution $z^*(\bar{\theta})$. Under these conditions, the IFT furthermore implies that the solution sensitivity can be computed as

$$\begin{bmatrix} \frac{\partial z^*}{\partial \theta}(\bar{s}, \bar{\theta}) \\ \frac{\partial \mu_A^*}{\partial \theta}(\bar{s}, \bar{\theta}) \end{bmatrix} = - \left(\frac{\partial r}{\partial(z, \mu_A)} \right)^{-1} \frac{\partial r}{\partial \theta} \quad (32)$$

where μ_A are the dual variables associated with the active constraints at the solution $z^*(\bar{s}, \bar{\theta})$ and where the partial derivatives of the residual map $r(z, \mu_A; s, \theta)$ are evaluated at $z = z^*(\bar{\theta}, \bar{s})$, $\mu_A = \mu_A^*(\bar{s}, \bar{\theta})$, $s = \bar{s}$, and $\theta = \bar{\theta}$. The residual map is defined as

$$r(z, \mu_A; s, \theta) = \begin{bmatrix} \nabla_z L(z; \theta) + \nabla_z g_A(z; s, \theta) \mu_A \\ g_A(z, \mu_A; s, \theta) \end{bmatrix} \quad (33)$$

where $g_A(z; s, \theta)$ denotes all equalities as well as all inequalities that are active at $z^*(\bar{s}, \bar{\theta})$.

At an active set changes, the solution map is not differentiable. Within the stochastic optimization framework, this nondifferentiability of the solution map for some values of θ and s is, in general, not problematic, as the gradient needs to be well-defined only almost everywhere.

A standard approximation replaces the residual map in (33) with a smoothed approximation of the KKT conditions associated with (31), as is done within an interior point solver which leads to a natural smoothing of the solution map alleviating the problem of nondifferentiability at active set changes [224].

For the interested reader, we provide some further references: In [337], the authors discuss differentiating through a parameterized MPC for active-set methods, whereas in [283], interior point methods for MPC with box constraints on the actions are considered. The factorization of the Karush-Kuhn-Tucker (KKT) system required for solving the parameterized MPC formulation can be reused to also derive the sensitivities for the parameters if an exact Hessian is used.

APPENDIX B

GRADIENTS IN THE HIERARCHICAL ARCHITECTURE

The following discussion is restricted to the hierarchical NN-MPC architecture of Fig 3. At each decision step, the NN predicts the parameter ϕ , which is then processed by the MPC-optimization layer to generate a control $u_0^*(\phi)$. We distinguish the learning approaches based on the actor's feedback. The value function, i.e., the critic, is evaluated for the predicted parameter ϕ or the resulting control $u_0^*(\phi)$. Note that this distinction is first and foremost important for the optimization properties for the NN component or when safe

exploration is required during training.

1) Parameter Critic: Assuming that the critic gives feedback based on the predicted parameter ϕ , the optimization layer can be considered part of the environment [214]. Thus, an altered version of the original MDP can be defined by introducing a modified MDP transition model

$$P^{\text{MPC}}(s^+|s, a) := P(s^+|s, u_0^*(a)). \quad (34)$$

One of the benefits of this approach is that implementations of RL methods can be directly used without any further adjustments, as the MPC optimization is only done when generating new samples by forward simulating the system from a given state to the next using (34).

2) Control Critic: Assuming that the critic gives feedback based on the control of the MPC optimization layer u_0^* , there are multiple ways to obtain a gradient. One can avoid differentiating through the MPC optimization layer

$$\nabla_\theta J(\theta) = \mathbb{E}_{\psi \sim \pi_\theta(s), s \sim \mathcal{D}} [Q(s, u_0^*(\psi)) \nabla_\theta \log \pi_\theta(\psi)],$$

by extending the stochastic policy gradient [339]. As there is no gradient information, using such a policy gradient can be seen as black box optimization with respect to the MPC optimization layer. This can be especially problematic for high-dimensional parameter spaces, as discussed in [27].

Alternatively, one can differentiate through the MPC [224] extending the deterministic policy gradient (16) to derive a policy gradient with

$$\nabla_\theta J^\pi(\theta) = \mathbb{E}_{s \sim \mathcal{D}} \left[\nabla_\theta \mu_\theta(s) \nabla_\psi Q^\mu(s, u_0^*(\psi)) \Big|_{\psi=\mu_\theta(s)} \right].$$

Doing the same for a stochastic policy, using the reparameterization trick, is an extension that has not been considered yet in the literature to the author's best knowledge.