Reinforcement Learning-Based Model Predictive Control for Discrete-Time Systems

Min Lin, Zhongqi Sun, *Member, IEEE*, Yuanqing Xia, *Senior Member, IEEE*, and Jinhui Zhang

***Abstract*— This article proposes a novel reinforcement learning-based model predictive control (RLMPC) scheme for discrete-time systems. The scheme integrates model predictive control (MPC) and reinforcement learning (RL) through policy iteration (PI), where MPC is a policy generator and the RL technique is employed to evaluate the policy. Then the obtained value function is taken as the terminal cost of MPC, thus improving the generated policy. The advantage of doing so is that it rules out the need for the offline design paradigm of the terminal cost, the auxiliary controller, and the terminal constraint in traditional MPC. Moreover, RLMPC proposed in this article enables a more flexible choice of prediction horizon due to the elimination of the terminal constraint, which has great potential in reducing the computational burden. We provide a rigorous analysis of the convergence, feasibility, and stability properties of RLMPC. Simulation results show that RLMPC achieves nearly the same performance as traditional MPC in the control of linear systems and exhibits superiority over traditional MPC for nonlinear ones.**

***Index Terms*— Discrete-time systems, model predictive control, policy iteration (PI), reinforcement learning (RL).**

I. Introduction

THE development of modern unmanned systems places higher demands on the performance of the controllers. Typically, these systems are constrained, and violation of constraints may lead to unsafe behaviors that can severely affect system operation. Model predictive control (MPC) is capable of providing an optimal solution while handling the constraints explicitly, which has seen significant success in recent decades with wide applications in diverse fields [1], [2], [3], [4], [5], [6].

The successful applications of MPC attracted tremendous academic interest, and soon a rigorous stability-centered MPC theoretical foundation was established. The early stability results on MPC employ a zero-state terminal equality constraint [7], [8]. They were subsequently extended to the use of a terminal inequality constraint by taking a control Lyapunov function as the terminal cost [9], [10], [11], which established the currently well-known framework for stability. Based on this framework, a commonly adopted MPC design paradigm involves finding an appropriate terminal cost and a terminal controller as well as a terminal set that strictly meet specific conditions. While this is relatively easy for linear systems, it is generally a challenging task for nonlinear ones. Even if a qualified terminal cost, a terminal controller, and the corresponding terminal set are found, the conditions for stability and recursive feasibility are usually quite conservative. In practice, however, the theoretically well-established MPC approaches are prohibitively difficult to apply due to the high computational burden. Instead, MPC without terminal cost and terminal constraint enjoys wide popularity. Its closed-loop stability and recursive feasibility can be guaranteed under some conditions [12], [13]. But the absence of terminal cost entails that the residual costs outside the truncated prediction horizon are completely ignored, which makes it difficult to achieve optimal performance under a very limited prediction horizon. Motivated by the analysis above, we aim to develop a learning-based MPC scheme to learn an appropriate terminal cost function without complex offline design procedures while improving the performance of the controller.

In recent years, striking progress of reinforcement learning (RL), such as AlphaGo [14] and Atari games [15], has drawn the attention of the control community. Different from MPC's offline design, RL optimizes the control policy through online data-based adaptation [16], [17]. Observed state transitions and costs (or rewards) are the only inputs to the RL agent, and no prior knowledge of the system dynamics is needed [18], [19]. A central idea in RL is temporal-difference (TD) learning [20], [21], [22], which estimates the value function directly from raw experience in a bootstrapping way, without waiting for a final outcome. The value function is a prediction of the expected long-term reward at each state. When the optimality is reached, it encodes the global optimal information so that the infinite-horizon optimal control policy can be obtained.

Nevertheless, to approximate the global optimal policy, the RL agent tends to try different policies and learns via trial and error, thereby struggling to provide safe guarantees on the resulting behaviors. This problem becomes prominent when confronted with some safety-critical systems, and safety in this context is defined in terms of stability. There have been some achievements centered on safe RL recently [23], [24], [25], but it remained largely an open field.

Manuscript received 30 March 2022; revised 31 January 2023; accepted 3 May 2023. Date of publication 19 May 2023; date of current version 1 March 2024. This work was supported in part by the Beijing Municipal Science Foundation under Grant 4222052; and in part by the National Natural Science Foundation of China under Grant 62003040, Grant 61836001, and Grant 61720106010. *(Corresponding author: Zhongqi Sun.)*

Min Lin, Yuanqing Xia, and Jinhui Zhang are with the School of Automation, Beijing Institute of Technology, Beijing 100081, China.

Zhongqi Sun is with the School of Automation, Beijing Institute of Technology, Beijing 100081, China, and also with the Yangtze Delta Region Academy of Beijing Institute of Technology, Jiaxing 314019, China (e-mail: sunzhongqi12@gmail.com).

Digital Object Identifier 10.1109/TNNLS.2023.3273590

In view of the powerful data-driven optimization capability of RL, if combined with MPC to optimize the controller design with running data, it can be expected to improve the control performance. At the same time, benefiting from the solid theoretical foundation of MPC, the properties of the behaviors produced by the RL agent would be easier to analyze. This motivates the attempts to integrate the best of both worlds. However, only limited pioneering work has been reported in this field. For example, in [26], a novel "plan online and learn offline" framework was proposed by taking MPC as the trajectory optimizer of RL. It is shown that MPC enables a more efficient exploration, thereby accelerating the convergence and reducing the approximation errors in the value function. The related work was extended to the framework of actor-critic (AC) in [27] to handle the case of sparse and binary reward, which shows that MPC is an important sampler that effectively propagates global information. These results were further applied to the tasks defined by simple and easily interpretable high-level objectives on a real unmanned ground vehicle in [28], and the expression of the value function was improved. Yet, the focus of these works is on the improvements that MPC brings to RL practice, with no reference to stability. The stability issue was highlighted in [29] where an economic MPC was used as a function approximator in RL. This scheme was further analyzed in [30] under an improved algorithmic framework. These two papers paved the way for [31], where the constraint satisfaction concern was addressed with the help of robust linear MPC. The practicality of these approaches, however, remains to be investigated and verified. To the best of the authors' knowledge, few studies can balance the theoretical and practical guarantees of the "RL + MPC" approaches, that is, to provide a theoretically safe and operationally efficient scheme, and this article aims to fill this gap.

We propose an "RL + MPC" scheme from a new perspective: policy iteration (PI). Considering that MPC's ability to handle constraints can provide constraint-enforcing policies for RL agents, *safety* referred to in this article is not only limited to stability, but also constraint satisfaction. The main contributions of this article contain threefold as follows.

1) We provide a new idea for combining RL and MPC by bridging them through PI. In this way, a complete RL-based model predictive control (RLMPC) scheme is developed for discrete-time systems. In each iteration, the current policy is evaluated through learning to obtain the value function. Then it is employed as the terminal cost to compensate for the suboptimality induced by the truncated prediction horizon of MPC, thus improving the policy. This solves the challenge of complex offline design procedures while progressively improving the performance to (near-)optimum.
2) The convergence of learning, recursive feasibility, and closed-loop stability of the proposed RLMPC are closely investigated, thereby theoretically guaranteeing its safety. We demonstrate that no constraint is violated even before the optimal value function is well-learned, which verifies the ability of MPC to effectively constrain the RL-produced policies within safe limits.
3) We incorporate the value function approximation (VFA) technique into the developed RLMPC approach to approximate the global optimal value function with high computational efficiency. The effect of the prediction horizon on the control performance is scrutinized. Results show that the proposed RLMPC scheme can achieve nearly the same optimal performance as traditional MPC in linear system control and outperform it in nonlinear cases, which is due to the conservativeness of the offline MPC design for nonlinear systems. In addition, thanks to the elimination of the terminal controller and terminal constraint, the RLMPC scheme is particularly useful in dealing with systems where it is difficult or even impossible to design a terminal controller, such as nonholonomic systems, while reducing computational burden by flexibly adjusting its prediction horizon.

The rest of this article is organized as follows. In Section II, we formally outline the problem formulation. The RLMPC scheme is developed in Section III. Its convergence, optimality, feasibility, and stability properties are analyzed in Section IV. Section V discusses the implementation of the overall scheme. We test the proposed scheme in simulations of both linear and nonlinear examples in Section VI. Section VII provides final conclusion.

Notation: We use the following convention: $\mathbb{R}$ denotes the set of reals and $\mathbb{R}^n$ is an n-dimensional set of real numbers. $\mathbb{N}$ is the set of natural numbers. For some $r_1, r_2 \in \mathbb{R}$, we use $\mathbb{R}_{\geq r_1}$, $\mathbb{N}_{>r_2}$, and $\mathbb{N}_{[r_1,r_2]}$ to represent the sets $\{r \in \mathbb{R} | r \geq r_1\}$, $\{r \in \mathbb{N} | r > r_2\}$, and $\{r \in \mathbb{N} | r_1 \leq r \leq r_2\}$, respectively. We label the variables of the optimal solution with $\cdot^*$, feasible solution with $\tilde{\cdot}$, and estimated ones with $\hat{\cdot}$, respectively. Moreover, the notations $x_{i|k}$ and $u_{i|k}$ indicate the state and input prediction i steps ahead from the current time k, respectively. The sequence $\{u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}\}$ is denoted by $\boldsymbol{u}_k$ or by $\boldsymbol{u}_{N\|k}$ if we want to emphasize its length N.

II. Problem Formulation

A. Optimal Control Problem

Consider a dynamic system described by the following state space difference equation:

$$x_{k+1} = f(x_k, u_k) \tag{1}$$

where $x_k \in \mathbb{R}^n$ and $u_k \in \mathbb{R}^m$ are the system state and control input at time $k \in \mathbb{N}$, respectively. It is assumed that the system is subject to the constraints

$$x_k \in \mathbb{X}, \quad u_k \in \mathbb{U} \tag{2}$$

where $\mathbb{X}$ and $\mathbb{U}$ are compact sets and contain the origin as an interior point. Also, system (1) is assumed to satisfy the following conditions.

Assumption 1: The function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is continuous with $f(0, 0) = 0$. Under the constraints (2), the system (1) is stabilizable with respect to the equilibrium at $x_k = 0$, $u_k = 0$, and the system state x_k is measurable.

Remark 1: Assumption 1 is a nonintrusive and common assumption in the MPC community and can be found in numerous literatures (e.g., [9], [10], [11], [12], [13]). For a

system $\breve{x}_{k+1} = f(\breve{x}_k, \breve{u}_k)$ with a general equilibrium (x^e, u^e), one can always make variable changes $x_k = \breve{x}_k - x^e$ and $u_k = \breve{u}_k - u^e$ to translate it to system (1) with the equilibrium $(0, 0)$.

For the particular setup considered above, the infinite-horizon optimal control problem at time k with the initial state $x_k \in \mathbb{X}$ is then described by Problem 1.

Problem 1: Find a control policy $u_k = \pi(x_k)$, which is defined as a map from state space to control space $\pi : \mathbb{R}^n \to \mathbb{R}^m$ and results in a control sequence $\boldsymbol{u}_\infty = \{u_k, u_{k+1}, \ldots\}$, such that it stabilizes the zero equilibrium of (1) and minimizes the infinite-horizon cost

$$J_\infty(x_k, \boldsymbol{u}_\infty) = \sum_{i=k}^{\infty} \ell(x_i, u_i) \tag{3}$$

subject to constraints (1) and (2), with the running cost $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ satisfying

$$\begin{aligned} &\ell(0, 0) = 0 \\ &\alpha_1^{\mathcal{K}_\infty}(\|x\|) \leq \breve{\ell}(x) \triangleq \inf_{u \in \mathbb{U}} \ell(x, u) \leq \alpha_2^{\mathcal{K}_\infty}(\|x\|) \end{aligned} \tag{4}$$

for $\forall x \in \mathbb{X}$, where $\alpha_1^{\mathcal{K}_\infty}$ and $\alpha_2^{\mathcal{K}_\infty}$ are $\mathcal{K}_\infty$ functions [32].

Remark 2: The running cost defined by (4) is a fairly general form and is common in many MPC research studies (e.g., [13], [33]). Its specific form is task-dependent, for example, it usually takes a quadratic form in regulation problems: $\ell(x, u) = x^{\mathrm{T}} Q x + u^{\mathrm{T}} R u$, where Q and R are set to be positive-definite matrices with proper dimensions.

B. Background of Model Predictive Control

Generally, there is no analytic solution to Problem 1, especially for constrained nonlinear systems. An efficient method to solve this problem is MPC [9], which provides an approximated solution by recursively solving the following OP 1 at each $x_k \in \mathbb{X}$.

OP 1: Find the optimal control sequence $\boldsymbol{u}_k^*$ by solving the finite-horizon minimization problem

$$\min_{\boldsymbol{u}_k} J(x_k, \boldsymbol{u}_k) = \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + F(x_{N|k}) \tag{5a}$$

$$\text{s.t. } x_{0|k} = x_k \tag{5b}$$

$$x_{i+1|k} = f(x_{i|k}, u_{i|k}), \quad 0 \leq i \leq N-1 \tag{5c}$$

$$u_{i|k} \in \mathbb{U}, \quad 0 \leq i \leq N-1 \tag{5d}$$

$$x_{i|k} \in \mathbb{X}, \quad 1 \leq i \leq N-1 \tag{5e}$$

$$x_{N|k} \in \mathbb{X}_\Omega \tag{5f}$$

Where $N \in \mathbb{N}_{>0}$ is the prediction horizon, $F(\cdot)$ is the *terminal cost*, $\boldsymbol{u}_k = \{u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}\}$ is the predicted control sequence, $\mathbb{X}_\Omega$ is the *terminal set*, and (5f) is the *terminal constraint*.

After solving OP 1, apply only the first element in $\boldsymbol{u}_k^*$, that is, $u_k = u_{0|k}^*$, to system (1) and repeat this procedure.

To guarantee the recursive feasibility and stability, it is required that N is chosen properly so that $x_{N|k}$ is guaranteed to enter a positively invariant set $\mathbb{X}_\Omega$ (under a terminal controller $\kappa(x_k) \in \mathbb{U}$) containing the origin. Moreover, parameters of the running and terminal costs should be designed to satisfy $F(x_{N|k}) \geq \sum_{i=N}^{\infty} \ell(x_{i|k}, \kappa(x_{i|k}))$ for $\forall x_k \in \mathbb{X}_\Omega$, such that $F(\cdot)$ is a local Lyapunov function for $\kappa(x_k)$ in $\mathbb{X}_\Omega$.

As mentioned in Section I, it is generally a difficult task to design such terminal conditions (i.e., terminal cost, terminal constraint, and terminal controller) for nonlinear systems. Although MPC without terminal condition (MPCWTC) is proposed as an option to circumvent this challenge, its performance under a very limited (but necessary) prediction horizon tends to be inferior to that of the kind with terminal cost. Therefore, we propose to introduce RL into MPC to learn the terminal cost online, thereby generating a policy that is closer to the optimal one in the sense of an infinite horizon.

C. Background of RL

An important class of RL techniques aims to learn the optimal policy $\pi^*(x)$ by iteratively obtaining the evaluation of the current policy and then improving the policy accordingly [34]. At state x_k, the value function $V_\pi : \mathbb{R}^n \to \mathbb{R}$, or the evaluation, for a given policy $u_k = \pi(x_k)$ is the accumulated rewards (or running costs) by following the policy from x_k, that is, $V_\pi(x_k) = \sum_{i=k}^{\infty} \ell(x_i, u_i)$, which can be rearranged into the well-known *Bellman equation*

$$V_\pi(x_k) = \ell(x_k, u_k) + V_\pi(x_{k+1}). \tag{6}$$

According to Bellman's principle, when the optimality is achieved, the optimal value function $V_\pi^*(x_k)$ satisfies the *Bellman optimality equation* [35]

$$V_\pi^*(x_k) = \min_{u_k} \left\{ \ell(x_k, u_k) + V_\pi^*(x_{k+1}) \right\} \tag{7}$$

and the corresponding optimal policy is given by

$$\pi^*(x_k) = \arg\min_{u_k} \left\{ \ell(x_k, u_k) + V_\pi^*(x_{k+1}) \right\}. \tag{8}$$

Nevertheless, the calculation of $V_\pi(x_k)$ is generally computationally difficult, and $V_\pi^*(x_k)$ is unknown before all the policies $\pi(x_k) = u_k \in \mathbb{U}$ are considered. This motivates the TD learning approach [36], which provides a more tractable computation. In TD learning, the running costs are observed to construct the TD target $V_{\mathrm{TD}}(x_k)$, thereby iteratively updating the value function as

$$V_\pi(x_k) \leftarrow V_\pi(x_k) + \alpha [V_{\mathrm{TD}}(x_k) - V_\pi(x_k)] \tag{9}$$

where $\alpha \in (0, 1)$ is a constant called the learning step size, and $V_{\mathrm{TD}}(x_k) - V_\pi(x_k)$ is known as the TD error. The Bellman equation (6) holds whenever the TD error is zero, thus obtaining the evaluation of policy $\pi(x_k)$. Here, to be consistent with MPC, the TD target can be constructed into the N-step form

$$V_{\mathrm{TD}}(x_k) = \sum_{i=k}^{k+N-1} \ell(x_i, u_i) + V_\pi(x_{k+N}). \tag{10}$$

It can be seen that this TD target has a similar form as the MPC cost function (5a), which builds up the connection between MPC and RL, and their combination gives rise to the RLMPC algorithm to be presented in Section III–V.

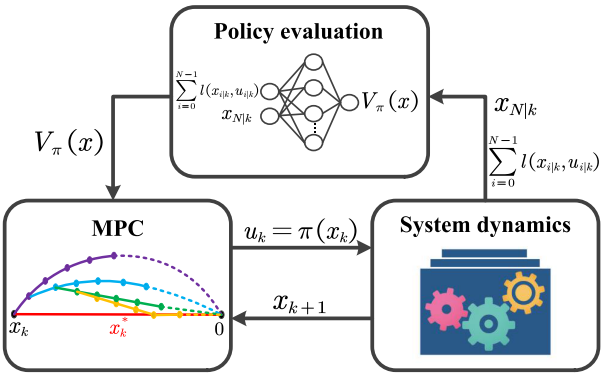

Fig. 1. RLMPC pipeline.

III. RL-Based MPC

In the proposed RLMPC, the value function and control policy are updated in the PI manner. Specifically, we take a currently obtained heuristic term as the terminal cost of MPC, and the corresponding control input is the current control policy. The costs incurred by following this policy, which can be regarded as the rewards for the controller interacting with the system dynamics, are used as training data for the learning of the value function. VFA technique is employed to approximate the underlying value function, that is, the evaluation of the current policy. With the latest learned value function, we take it as the heuristic term in the cost function, thus updating the MPC controller, that is, the current policy. Fig. 1 illustrates the RLMPC pipeline.

A. Policy Generator—MPC

In this section, we formulate the MPC problem to be solved at each step (see OP 2). Since MPC only looks N steps forward at each control period, it ultimately produces a locally optimal policy for Problem 1 unless coupled with a terminal cost that propagates global information [26]. Motivated by this, we take the heuristic term learned through RL, namely the value function, as the terminal cost. Based on the results presented in [13], we modify the formulation of MPC as follows.

OP 2: Find the optimizer $\boldsymbol{u}_k$ of the following problem:

$$\min_{\boldsymbol{u}_k} J(x_k, \boldsymbol{u}_k) = \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi(x_{N|k}) \tag{11a}$$

$$\text{s.t. } x_{0|k} = x_k \tag{11b}$$

$$x_{i+1|k} = f(x_{i|k}, u_{i|k}), 0 \le i \le N-1 \tag{11c}$$

$$u_{i|k} \in \mathbb{U}, \quad 0 \le i \le N-1 \tag{11d}$$

$$x_{i|k} \in \mathbb{X}, \quad 1 \le i \le N \tag{11e}$$

where the control sequence $\boldsymbol{u}_k = \{u_{0|k}, u_{1|k}, \ldots, u_{N-1|k}\}$ is the decision variable, and $V_\pi(\cdot)$ is a heuristic term obtained through the learning technique to be presented in Section III-B. We define the MPC control policy generated under a fixed $V_\pi(\cdot)$ to be a fixed policy.

Up to this point, we obtain the policy $\pi(x_k)$ implicitly defined by OP 2: solving OP 2 at time k to yield a control sequence $\boldsymbol{u}_k^* = \{u_{0|k}^*, u_{1|k}^*, \ldots, u_{N-1|k}^*\}$, and the first element is applied to system (1).

Remark 3: A significant advantage of this formulation lies in the fact that it not only inherits the advantages of MPCWTC [13], but also has the potential to further improve the control performance by compensating the costs neglected by the truncated prediction horizon through the learned heuristic function term.

In this article, the initial policy $\pi^0(x_k)$ is generated by solving OP 2 with $V_\pi(\cdot) \equiv 0$ and implemented in the receding horizon fashion, which is exactly the MPCWTC. To ensure its stability and recursive feasibility, we make the following assumptions.

Assumption 2 (Controllability Assumption): There exists a positive constant $\beta \in \mathbb{R}_{>0}$ such that

$$V_\pi^*(x_k) \le \beta \cdot \check{\ell}(x_k) \quad \forall x_k \in \mathbb{X} \tag{12}$$

where $V_\pi^*(x_k)$ and $\check{\ell}(x_k)$ are defined in (7) and (4), respectively.

Lemma 1 (See [13]): If the prediction horizon N of the MPCWTC is chosen to satisfy

$$N > \frac{2\ln\beta}{\ln\beta - \ln(\beta - 1)} \tag{13}$$

it is guaranteed that the closed-loop system is asymptotically stable and the MPCWTC optimization problem is recursively feasible.

Assumption 3: In this context, the prediction horizon of the initial controller satisfies (13).

Remark 4: Assumption 2 is a fairly standard condition in the MPCWTC literature (e.g., [13], [37], [38]) and is also known as the boundedness condition of the optimal value function. It indicates the controllability of system (1) with respect to the equilibrium, since otherwise $V_\pi^*(x_k)$ would go to infinity [13]. Lemma 1 and Assumption 3 jointly determine that the initial controller is stable, which is consistent with the requirement on the initial policy for PI (to be introduced in Section III-C). The method to calculate a suitable β can be found in [39] and [40].

B. Learning of the Value Function

Now we focus on how to perform policy evaluation on *a fixed control policy* $\pi(x_k)$ at time k. As mentioned in Section II-C, the essence of policy evaluation is to obtain $V_\pi(x_k)$ for $\pi(x_k)$ on the set $\mathbb{X}$. For the continuous state space, however, it is impossible to obtain $V_\pi(x_k)$ at every $x_k \in \mathbb{X}$, because we could not traverse all states over the state space. Therefore, the VFA technique [36] is adopted in this article to approximate the existing true value function.

According to the higher-order Weierstrass approximation theorem [41], there exists a dense polynomial basis set $\{\Phi_i(x_k)\}$ such that one can approximate the true value function $V_\pi(x_k)$, $\forall x_k \in \mathbb{X}$ in the following form:

$$\begin{aligned} V_\pi(x_k) &= \sum_{i=1}^{\infty} W_i \Phi_i(x_k) = \sum_{i=1}^{p} W_i \Phi_i(x_k) + \sum_{i=p+1}^{\infty} W_i \Phi_i(x_k) \\ &= W^{\mathrm{T}} \Phi(x_k) + e(x_k) = \hat{V}_\pi(x_k, W) + e(x_k) \end{aligned} \tag{14}$$

where $\hat{V}_\pi(x_k, W)$ is an approximation of $V_\pi(x_k)$, $W = [W_1, \ldots, W_p]^{\mathrm{T}} \in \mathbb{R}^p$ is the weight vector, and $\Phi(x_k) = [\Phi_1(x_k), \ldots, \Phi_p(x_k)]^{\mathrm{T}}$ is the basis vector. The approximation error $e(x_k)$ converges uniformly to zero as $p \to \infty$. Note that this approximation is essentially a single-layer network, and all we have to do is to feed training data into this network.

Consider a set containing q ($q \in \mathbb{N}_{>0}$) states sampled in $\mathbb{X}$ at time k, denoted as $S_k = \{x_{k_1}, x_{k_2}, \ldots, x_{k_q}\} \subset \mathbb{X}$. Take each of its elements separately as initial states and propagate one step using this policy, then we record the incurred costs as training data. Specifically, for each $x_{k_j} \in S_k$, $j \in \mathbb{N}_{[1,q]}$, solve OP 2 (whose terminal cost is a fixed heuristic function corresponding to the current policy) with the subscript k replaced by k_j, obtain the solution $\boldsymbol{u}^*_{k_j}$, and calculate the cost $J(x_{k_j}, \boldsymbol{u}^*_{k_j})$ according to (11a). Note that we do not apply any $\boldsymbol{u}^*_{k_j}$ to the real system in this process, so the new subscript k_j is adopted here to distinguish it from the actual system state.

With all these costs $J(x_{k_j}, \boldsymbol{u}^*_{k_j})$, $j \in \mathbb{N}_{[1,q]}$, we are now able to train the network similar to the TD learning. Recall that the purpose of (9) is to approximate the TD target with $V_\pi(x_k)$, we now directly use $\hat{V}_\pi(x_{k_j}, W)$ to approximate the targets $J(x_{k_j}, \boldsymbol{u}^*_{k_j})$

$$\hat{V}_\pi(x_{k_j}, W) \leftarrow \hat{V}_\pi(x_{k_j}, W) + \alpha\left[J\left(x_{k_j}, \boldsymbol{u}^*_{k_j}\right) - \hat{V}_\pi(x_{k_j}, W)\right] \tag{15}$$

for $\forall j \in \mathbb{N}_{[1,q]}$, and we aim to minimize $E(x_{k_j}) = 1/2 e(x_{k_j})^2$, where $e(x_{k_j}) = J(x_{k_j}, \boldsymbol{u}^*_{k_j}) - \hat{V}_\pi(x_{k_j}, W)$. The stochastic gradient descent method [36] achieves this minimization by adjusting the weight vector W in a small step size $\alpha \in (0, 1)$ at a time in the direction of the negative gradient (with respect to W) of the squared error

$$\begin{aligned} W &\leftarrow W - \frac{1}{2}\alpha\nabla_W\left[J\left(x_{k_j}, \boldsymbol{u}^*_{k_j}\right) - \hat{V}_\pi\left(x_{k_j}, W\right)\right]^2 \\ &= W + \alpha\left[J\left(x_{k_j}, \boldsymbol{u}^*_{k_j}\right) - \hat{V}_\pi\left(x_{k_j}, W\right)\right]\nabla_W \hat{V}_\pi\left(x_{k_j}, W\right) \\ &= W + \alpha\left[J\left(x_{k_j}, \boldsymbol{u}^*_{k_j}\right) - W^{\mathrm{T}}\Phi\left(x_{k_j}\right)\right]\Phi\left(x_{k_j}\right), \quad j \in \mathbb{N}_{[1,q]} \end{aligned} \tag{16}$$

in which ∇_W denotes the partial derivatives with respect to the components of W.

After W converges for all the training data, that is, for $\forall x_{k_j} \in S_k$, $[J(x_{k_j}, \boldsymbol{u}^*_{k_j}) - \hat{V}_\pi(x_{k_j}, W)] \to 0$ holds, we obtain the (approximated) evaluation of the given fixed control policy. Provided with sufficient training data and a proper choice of the basis vector, we have $\hat{V}_\pi(x_k, W) \to V_\pi(x_k)$, $\forall x_k \in \mathbb{X}$. It is worth noting that with VFA, we can characterize the value function over the state space using only the storage space of a p-dimensional vector W, which enhances the utility of the proposed approach.

C. PI in RLMPC

In RLMPC, the value function and the control policy are updated by iterations with index $t \in \mathbb{N}$ increasing from zero to infinity. For convenience, we denote the heuristic term employed in the tth iteration as $V^t_\pi(\cdot)$, and the policy generated by the MPC with $V^t_\pi(\cdot)$ as the terminal cost is denoted as $\pi^t(\cdot)$. We are now in a position to present the PI mechanism in RLMPC.

1) *Initialization:* Given an initial state $x_k \in \mathbb{X}$ at time k, we start the iteration with an initial heuristic term $V^0_\pi(\cdot) \equiv 0$ and a corresponding stabilizing policy $\pi^0(x_k)$ whose existence is already guaranteed by Assumptions 2 and 3. For $t = 0, 1, 2, \ldots$, do the following two steps iteratively.
2) *Policy Evaluation Step:* Determine the value function $V^{t+1}_\pi(\cdot)$ for policy $\pi^t(\cdot)$ by following the steps presented in Section III-B: first, sample in $\mathbb{X}$ to form S_k. Second, generate training data $J^t(x_{k_j}, \boldsymbol{u}^t_{k_j})$, $x_{k_j} \in S_k$, with

$$\begin{aligned} J^t(x_{k_j}, \boldsymbol{u}^t_{k_j}) &= \min_{\boldsymbol{u}_{k_j} \in \mathbb{U}}\left\{\sum_{i=0}^{N-1} \ell(x_{i|k_j}, u_{i|k_j}) + V^t_\pi(x_{N|k_j})\right\} \\ &= \sum_{i=0}^{N-1} \ell\left(x^t_{i|k_j}, u^t_{i|k_j}\right) + V^t_\pi\left(x^t_{N|k_j}\right) \end{aligned} \tag{17}$$

where $x_{k_j} = x^t_{0|k_j}$, $x^t_{i|k_j}$, and $u^t_{i|k_j}$, $i \in \mathbb{N}_{[0,N-1]}$ are deduced from model (11c), satisfying (11d) and (11e). Third, train the network using (16) to obtain the value function

$$V^{t+1}_\pi(x_k) \to J^t(x_k, \boldsymbol{u}^t_k) \quad \forall x_k \in \mathbb{X}. \tag{18}$$

3) *Policy Improvement Step:* According to the descriptions in Section III-A, the updated policy $\pi^{t+1}(x_k)$ is yielded by solving OP 2 with terminal cost $V^{t+1}_\pi(\cdot)$

$$\boldsymbol{u}^{t+1}_k = \arg\min_{\boldsymbol{u}^{t+1}_k \in \mathbb{U}}\left\{\sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V^{t+1}_\pi(x_{N|k})\right\} \tag{19}$$

and $\boldsymbol{u}^{t+1}_k$ is implemented in a receding horizon manner.

In what follows, we focus on analyzing the theoretical properties of this RLMPC scheme.

IV. Convergence, Feasibility, and Stability Analysis

In this section, we analyze the convergence, feasibility, and stability properties. We show that for $\forall x_k \in \mathbb{X}$, the value function $V^t_\pi(x_k) \to V^*_\pi(x_k)$ and the control policy $\pi^t(x_k) \to \pi^*(x_k)$ as the iteration index $t \to \infty$. Also, we demonstrate that the policy obtained in each iteration is a feasible and stabilizing policy for the closed-loop system. The convergence proof is inspired by [42] which demonstrates the convergence of optimal control (unconstrained infinite horizon optimization) under PI. This article leverages its proof idea to extend the results to RLMPC. Before proceeding, we introduce the following definition.

Definition 1: For the RLMPC optimization problem OP 2, a control sequence $\tilde{\boldsymbol{u}}_k = \{\tilde{u}_{0|k}, \ldots, \tilde{u}_{N-1|k}\}$ is said to be feasible if $\tilde{u}_{i|k} \in \mathbb{U}$, $i = \mathbb{N}_{[0,N-1]}$ and the resulting system trajectory $\tilde{\boldsymbol{x}}_k = \{\tilde{x}_{0|k}, \ldots, \tilde{x}_{N|k}\}$ satisfies $\tilde{x}_{i|k} \in \mathbb{X}$, $i = \mathbb{N}_{[0,N]}$, rendering the cost function $J(\tilde{x}_k, \tilde{\boldsymbol{u}}_k)$ in the form of (11a) finite.

A. Convergence Analysis

We first present two theorems to respectively demonstrate that the value function $V_\pi^t(x_k)$ is uniformly monotonically nondecreasing with respect to the iteration index t and show that it has an upper bound. Then, based on the two theorems, it is shown in the third theorem that $V_\pi^t(x_k)$ converges to the optimal solution as $t \to \infty$.

Theorem 1: For $\forall t \in \mathbb{N}$ and $x_k \in \mathbb{X}$, let $V_\pi^t(x_k)$ and $\pi^t(x_k)$ be obtained by PI presented in Section III-C, then $V_\pi^t(x_k)$ is a uniformly monotonically nondecreasing sequence for t, that is, $\forall t : V_\pi^t(x_k) \le V_\pi^{t+1}(x_k)$.

Proof: Define a new value function $\Gamma^t(x_k)$ with $\Gamma^0(\cdot) \equiv 0$. It is followed by the update law that

$$\Gamma^t(x_k) = \sum_{i=0}^{N-1} \ell(\tilde{x}_{i|k}, \tilde{u}_{i|k}) + \Gamma^{t-1}(\tilde{x}_{N|k}) \tag{20}$$

where $\{\tilde{u}_{0|k}, \ldots, \tilde{u}_{N-1|k}\} = \tilde{\boldsymbol{u}}_k$ is an arbitrary feasible control sequence as defined in Definition 1, $\tilde{x}_{0|k} = x_k$, and $\tilde{x}_{i|k}, i = \mathbb{N}_{[1,N]}$, is the resulting trajectory of $\tilde{\boldsymbol{u}}_k$. Since $\Gamma^0(\cdot) = V_\pi^0(\cdot) = 0$, we have

$$\begin{aligned}
\Gamma^1(x_k) &= \sum_{i=0}^{N-1} \ell(\tilde{x}_{i|k}, \tilde{u}_{i|k}) + \Gamma^0(\tilde{x}_{N|k}) \\
&\ge \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + \Gamma^0(x_{N|k}) \right\} \\
&= \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^0(x_{N|k}) \right\} \\
&= \sum_{i=0}^{N-1} \ell\left(x_{i|k}^0, u_{i|k}^0\right) + V_\pi^0\left(x_{N|k}^0\right) = V_\pi^1(x_k).
\end{aligned} \tag{21}$$

The last two equations hold from (17) and (18), indicating the following update law:

$$\begin{aligned}
V_\pi^{t+1}(x_k) &= J^t\left(x_k, \boldsymbol{u}_k^t\right) \\
&= \sum_{i=0}^{N-1} \ell\left(x_{i|k}^t, u_{i|k}^t\right) + V_\pi^t\left(x_{N|k}^t\right) \\
&= \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^t(x_{N|k}) \right\}.
\end{aligned} \tag{22}$$

Assume that at $t = l$, $\Gamma^l(x_k) \ge V_\pi^l(x_k)$ holds for $\forall x_k \in \mathbb{X}$, then for $t = l + 1$

$$\begin{aligned}
\Gamma^{l+1}(x_k) &= \sum_{i=0}^{N-1} \ell(\tilde{x}_{i|k}, \tilde{u}_{i|k}) + \Gamma^l(\tilde{x}_{N|k}) \\
&\ge \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + \Gamma^l(x_{N|k}) \right\} \\
&= \sum_{i=0}^{N-1} \ell\left(\bar{x}_{i|k}^l, \bar{u}_{i|k}^l\right) + \Gamma^l\left(\bar{x}_{N|k}^l\right) \\
&\ge \sum_{i=0}^{N-1} \ell\left(\bar{x}_{i|k}^l, \bar{u}_{i|k}^l\right) + V_\pi^l\left(\bar{x}_{N|k}^l\right) \\
&\ge \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^l(x_{N|k}) \right\} \\
&= \sum_{i=0}^{N-1} \ell\left(x_{i|k}^l, u_{i|k}^l\right) + V_\pi^l\left(x_{N|k}^l\right) = V_\pi^{l+1}(x_k)
\end{aligned}$$

where $\bar{u}_{i|k}^l$, $\bar{x}_{i|k}^l$, $i = 0, 1, \ldots$, are the solutions to the problem $\min_{\boldsymbol{u}_k \in \mathbb{U}} \{\sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + \Gamma^l(x_{N|k})\}$. Through induction, we conclude that

$$\forall t : \Gamma^t(x_k) \ge V_\pi^t(x_k). \tag{23}$$

Next, we use this conclusion to prove that $V_\pi^t(x_k)$ is a uniformly monotonically nondecreasing sequence about t. Again, the mathematical induction method is adopted. First, since $V_\pi^1(x_k) = \sum_{i=0}^{N-1} \ell(x_{i|k}^0, u_{i|k}^0) \ge 0$ and $\Gamma^0(x_k) = 0$, it is obvious that $\Gamma^0(x_k) \le V_\pi^1(x_k)$. Second, if we let the arbitrary feasible policy $\tilde{\boldsymbol{u}}_k$ be $\boldsymbol{u}_k^t$ in (20), we have the following update law at $t = l$:

$$\Gamma^l(x_k) = \sum_{i=0}^{N-1} \ell\left(x_{i|k}^l, u_{i|k}^l\right) + \Gamma^{l-1}\left(x_{N|k}^l\right). \tag{24}$$

At this time, assume that $\Gamma^{l-1}(x_k) \le V_\pi^l(x_k)$ holds for $\forall x_k \in \mathbb{X}$, then at $t = l + 1$, from (24) and (22), we derive

$$\Gamma^l(x_k) - V_\pi^{l+1}(x_k) = \Gamma^{l-1}\left(x_{N|k}^l\right) - V_\pi^l\left(x_{N|k}^l\right) \le 0. \tag{25}$$

Hence, $\Gamma^l(x_k) \le V_\pi^{l+1}(x_k)$ holds. Therefore, we conclude that $\forall t : \Gamma^t(x_k) \le V_\pi^{t+1}(x_k)$. Combining (23), we can draw the conclusion that

$$\forall t : V_\pi^t(x_k) \le \Gamma^t(x_k) \le V_\pi^{t+1}(x_k) \tag{26}$$

which completes the proof. ■

Remark 5: Since $V_\pi^0(\cdot) = 0$ and the running cost (4) is positive-definite, Theorem 1 indicates that the value function $V_\pi^t(x_k)$ obtained in any iteration $t \in \mathbb{N}_{\ge 1}$ is positive-definite.

Theorem 2: For $\forall t \in \mathbb{N}$ and $x_k \in \mathbb{X}$, let $V_\pi^t(x_k)$ and $\pi^t(x_k)$ be obtained by PI presented in Section III-C, and $V_\pi^*(x_k)$ be the optimal value function satisfying the Bellman optimality equation

$$V_\pi^*(x_k) = \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^*(x_{N|k}) \right\} \tag{27}$$

then $V_\pi^*(x_k)$ is an upper bound of $V_\pi^t(x_k)$, that is,

$$\forall t : V_\pi^t(x_k) \le V_\pi^*(x_k). \tag{28}$$

Proof: We prove this by induction. First, at $t = 0$, $V_\pi^0(x_k) = 0 \le V_\pi^*(x_k)$. Second, we assume that at $t = l$, $V_\pi^l(x_k) \le V_\pi^*(x_k)$ holds for $\forall x_k \in \mathbb{X}$, then at $t = l + 1$, we have

$$\begin{aligned}
V_\pi^{l+1}(x_k) &= \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^l(x_{N|k}) \right\} \\
&\le \min_{\boldsymbol{u}_k \in \mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^*(x_{N|k}) \right\} \\
&= V_\pi^*(x_k).
\end{aligned} \tag{29}$$

Thus, $V_\pi^t(x_k) \le V_\pi^*(x_k)$ holds for $\forall t \in \mathbb{N}$, and $V_\pi^*(x_k)$ is an upper bound. The proof is completed. ■

Based on the properties revealed by Theorems 1 and 2, we give the following theorem to show the convergence of RLMPC.

Theorem 3: For $\forall t \in \mathbb{N}$ and $x_k \in \mathbb{X}$, let $V_\pi^t(x_k)$ and $\pi^t(x_k)$ be obtained by PI presented in Section III-C. When $t \to \infty$, $\lim_{t\to\infty} V_\pi^t(x_k) = V_\pi^\infty(x_k) = V_\pi^*(x_k)$ and $\lim_{t\to\infty} \pi^t(x_k) = \pi^\infty(x_k) = \pi^*(x_k)$.

Proof: From Theorems 1 and 2, we know that $V_\pi^t(x_k)$ is a uniformly monotonically nondecreasing and upper-bounded sequence about t. According to the monotone convergence theorem [43], $V_\pi^t(x_k)$ converges to a value, denoted by $V_\pi^\infty(x_k)$, as $t \to \infty$. Since $\lim_{t\to\infty} V_\pi^t(x_k) = \lim_{t\to\infty} V_\pi^{t+1}(x_k) = V_\pi^\infty(x_k)$, we obtain

$$\begin{aligned} V_\pi^\infty(x_k) &= \lim_{t\to\infty} V_\pi^{t+1}(x_k) \\ &= \lim_{t\to\infty} \min_{\boldsymbol{u}_k\in\mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^t(x_{N|k}) \right\} \\ &= \min_{\boldsymbol{u}_k\in\mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + \lim_{t\to\infty} V_\pi^t(x_{N|k}) \right\} \\ &= \min_{\boldsymbol{u}_k\in\mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + \lim_{t\to\infty} V_\pi^{t+1}(x_{N|k}) \right\} \\ &= \sum_{i=0}^{N-1} \ell\left(x_{i|k}^\infty, u_{i|k}^\infty\right) + V_\pi^\infty\left(x_{N|k}^\infty\right) \end{aligned}$$

which is the Bellman optimality equation. Thus, $V_\pi^\infty(x_k) = V_\pi^*(x_k)$ and the resulting policy $\lim_{t\to\infty} \pi^t(x_k) = \pi^\infty(x_k) = \pi^*(x_k)$. The proof is completed. ■

B. Stability and Feasibility Analysis

Theorem 4: For $\forall t \in \mathbb{N}$ and $x_k \in \mathbb{X}$, let $V_\pi^t(x_k)$ and $\pi^t(x_k)$ be obtained by PI presented in Section III-C, then for every $t \in \mathbb{N}$, the RLMPC policy $\pi^t(x_k)$ is feasible and the closed-loop system is asymptotically stable under $\pi^t(x_k)$.

Proof: For $t = 0$, Assumption 3 has already ensured the asymptotic stability and feasibility of the initial policy $\pi^0(x_k)$, $\forall x_k \in \mathbb{X}$. Before proceeding, we define

$$V_{\tilde{N}}(x_k) = \min_{\boldsymbol{u}_{\tilde{N}\|k}\in\mathbb{U}} \sum_{i=0}^{\tilde{N}-1} \ell(x_{i|k}, u_{i|k}) \tag{30}$$

subject to (11b)–(11e) with N replaced by $\tilde{N}$, where $\tilde{N}$ can be an arbitrary positive integer, and $\boldsymbol{u}_{\tilde{N}\|k} = \{u_{0|k}, u_{1|k}, \ldots, u_{\tilde{N}-1|k}\}$ is the decision variable of length $\tilde{N}$. By letting $\tilde{N} = N$, it is obvious that $V_N(x_k) = J^0(x_k, \boldsymbol{u}_{N\|k}^0) = V_\pi^1(x_k)$ holds for $\forall x_k \in \mathbb{X}$. Furthermore, we can also obtain

$$\begin{aligned} V_{2N}(x_k) &= \min_{\boldsymbol{u}_{2N\|k}\in\mathbb{U}} \sum_{i=0}^{2N-1} \ell(x_{i|k}, u_{i|k}) \\ &= \min_{\boldsymbol{u}_{2N\|k}\in\mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + \sum_{i=N}^{2N-1} \ell(x_{i|k}, u_{i|k}) \right\} \\ &= \min_{\boldsymbol{u}_{N\|k}\in\mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_N(x_{N|k}) \right\} \end{aligned} \tag{31}$$

where the last equation holds because of Bellman's optimality principle. As a result, invoking (22) leads to

$$\begin{aligned} V_{2N}(x_k) &= \min_{\boldsymbol{u}_{N\|k}\in\mathbb{U}} \left\{ \sum_{i=0}^{N-1} \ell(x_{i|k}, u_{i|k}) + V_\pi^1(x_{N|k}) \right\} \\ &= J^1(x_k, \boldsymbol{u}_{N\|k}^1) = V_\pi^2(x_k). \end{aligned} \tag{32}$$

This illustrates that the policy generated by RLMPC with $V_\pi^1(x_{N|k})$ as the terminal cost is equivalent to that of the MPCWTC with a prediction horizon length of $2N$. By induction, it is easy to verify

$$V_{(t+1)\cdot N}(x_k) = J^t(x_k, \boldsymbol{u}_{N\|k}^t) = V_\pi^{t+1}(x_k) \quad \forall t \in \mathbb{N} \tag{33}$$

which implies that the policy $\pi^t(x_k)$ is equivalent to the MPCWTC with prediction horizon $(t+1)\cdot N$. Given that N satisfies (13) and hence $(t+1)\cdot N \ge 2\ln\beta/[\ln\beta - \ln(\beta-1)]$ also holds, the asymptotic stability and feasibility of $\pi^t(x_k)$ can be guaranteed according to Lemma 1. The proof is completed. ■

Remark 6: The above proof reveals that the essence of RLMPC is to achieve the effect of accumulating the prediction horizon on the basis of the initial controller through iterations. As $t \to \infty$, $\pi^t(x_k)$ converges to the policy generated by the MPCWTC with an infinite prediction horizon, that is, the optimal policy. This property echoes the results given by Theorem 1–Theorem 3. In this sense, (33) serves as a performance indicator for each iteration, and N determines the convergence rate (CR).

Remark 7: For specific control tasks, the MPCWTC could generally achieve (near-)optimal performance with a very limited prediction horizon $0 < \bar{N} \ll \infty$, without requiring an infinite one. In other words, if the prediction horizon of MPCWTC is no less than $\bar{N}$, increasing its prediction horizon hardly improves the performance. This indicates that we do not have to iterate RLMPCs to $t \to \infty$, but can stop at $0 < \bar{t} \ll \infty$ if $V_\pi^{\bar{t}}(x_k)$ hardly changes with a larger $\bar{t}$. Furthermore, with $V_\pi^{\bar{t}}(x_k)$ being the terminal cost, one can flexibly adjust the prediction horizon of RLMPC and resulting in almost the same control policy $\pi^{\bar{t}}(x_k) \to \pi^*(x_k)$, as the Bellman optimality equation (7) inherently holds at any prediction horizon. Since the prediction horizon determines the dimensionality and hence the computational burden of the optimization problem, such a property provides a way to significantly reduce the computational burden by shortening the prediction horizon of RLMPC.

V. Implementation Issues on RLMPC

A. On the PI

Theorems 1 and 2 reveal that as t increases, $V_\pi^t(x_k)$ exhibits a monotonic approximation to $V_\pi^*(x_k)$. This implies that each iteration will necessarily lead to some degree of improvement in $V_\pi^t(x_k)$. More importantly, Theorem 4 demonstrates that $V_\pi^t(x_k)$ obtained at any $t \in \mathbb{N}$ can result in a stabilizing control policy. This allows us to stop the iteration at any t without affecting the stability of the closed-loop system.

As we can see in Section III-B, the update of $V_\pi^t(x_k)$ is determined by the weight vector W. Denote the weight vector

Algorithm 1 RLMPC Algorithm

Initialize: N, Q, R, α, ε, p, $\Phi(x_k)$, $W^0 = \mathbf{0}$, $t = 0$, $Flag = 0$;
for $k = 1, 2, 3, \ldots$ **do**
  Measure the current state x_k at time k;
  Solve OP 2 for x_k with terminal cost $(W^t)^{\mathrm{T}}\Phi(x_{N|k})$;
  Apply $\pi^t(x_k) = u^t_{0|k}$ to system (1);
  if $Flag == 0$ **then**
    $W^{tq}_{k_0} \leftarrow W^t$;
    Construct sample set $S_k = \{x^t_{k_1}, \ldots, x^t_{k_q}\} \subset \mathbb{X}$;
    for $j = 1, 2, \ldots, q$ **do**
      Solve OP 2 for $x^t_{k_j}$ with terminal cost $(W^t)^{\mathrm{T}}\Phi(\cdot)$;
      Calculate $J(x^t_{k_j}, u^t_{k_j})$ according to (17);
      $W^{tq}_{k_j} \leftarrow W^{tq}_{k_{j-1}}$;
      Update $W^{tq}_{k_j}$ using (16) until $\|\Delta W^{tq}_{k_j}\| \le \varepsilon$;
      if $\|W^{tq}_{k_j} - W^{tq}_{k_{j-1}}\| \le \varepsilon$ **then** break;
      end if
    end for
    $W^{t+1} \leftarrow W^{tq}_{k_j}$;
  end if
  if $\|W^{t+1} - W^t\| \le \varepsilon$ **then**
    $Flag == 1$;
  end if
  $t \leftarrow t + 1$;
end for

obtained in the tth iteration as W^t (which has converged for all the training data in this iteration) and select a small constant $\varepsilon > 0$ as the threshold of change. Instead of iterating infinitely many steps, if $\|W^{t+1} - W^t\| \le \varepsilon$ is satisfied for some t, then $V^t_\pi(x_k)$ can be regarded as converged to the neighborhood of $V^*_\pi(x_k)$ and the iteration can be stopped.

Remark 8: In contrast to PI, value iteration (VI) is considered less computationally expensive by reducing the policy evaluation step to a single iteration. Although it is possible to adapt RLMPC to the VI update manner, the properties presented in Section IV may be lost. This is due to the fact that the intermediate policies of VI are not guaranteed to be stable and the intermediate value functions may not correspond to any policy. Taking such value functions as the terminal costs, it is difficult to ensure that the corresponding RLMPC generates suitable policies for value function updates. How to ensure the convergence, stability, and feasibility in the case of VI would be our future research.

B. Overall Algorithm of the RLMPC Scheme

Given the state x_k, use the policy $\pi^0(x_k)$ to control the real system for $r^0 \ge 1$ steps. In the meanwhile, perform the policy evaluation for $\pi^0(\cdot)$ as presented in Section III-B. Once the value function $V^1_\pi(\cdot)$ is obtained, substitute it into OP 2 as terminal cost and update the policy to $\pi^1(\cdot)$. This procedure is repeated for $\forall t \ge 1$, and $\pi^t(\cdot)$ is applied to the real system for r^t steps ($r^t \in \mathbb{N}_{\ge 1}$ is a constant at a given t). This learning procedure is characterized by the fact that the policy is continually improved throughout the control process.

The pseudo-code of the RLMPC algorithm is shown in Algorithm 1, which is an example of updating the policy at every control step, that is, $r^t \equiv 1$, $\forall t \in \mathbb{N}$.

VI. Simulation Examples

To evaluate the effectiveness of the proposed RLMPC scheme, two examples are studied: one involving a linear system and the other involving a nonlinear one. As is known, traditional MPC (which refers to the one described in OP 1) can attain linear quadratic optimal performance for linear systems with properly designed terminal conditions, so the linear system example can be a benchmark for RLMPC performance. For nonlinear systems, however, it can only achieve suboptimal control performance, which is where RLMPC comes in. We will highlight the superiority of RLMPC in the nonlinear system example. In both examples, we investigate RLMPC (following Algorithm 1) for two episodes. The first episode (labeled as "RLMPC Epi. 1"), called the *learning episode*, is to learn from scratch, starting with the initial policy to control the system; while the second episode (labeled as "RLMPC Epi. 2"), also referred to as the *re-execution episode*, is to redo the task under the learned value function (LVF) to check the performance.

A. Linear System

Consider the following linear system:

$$x_{k+1} = Ax_k + Bu_k \tag{34}$$

where $x_k = [x_{1k}, x_{2k}]^{\mathrm{T}} \in \mathbb{R}^2$, $u_k \in \mathbb{R}^1$, $A = \begin{bmatrix} 1 & 0.5 \\ -0.1 & 0.9 \end{bmatrix}$, and $B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. We set the running cost to be $\ell(x_k, u_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k$ with weights $Q = I \in \mathbb{R}^2$ and $R = 0.5$. Then, according to the traditional MPC design procedure [9], we solve a Riccati equation and obtain $P = \begin{bmatrix} 1.4388 & -0.3409 \\ -0.3409 & 5.0793 \end{bmatrix}$ with terminal cost $F(x_{N|k}) = x^{\mathrm{T}}_{N|k} P x_{N|k}$, and the terminal controller $u_k = Kx_k$ with $K = [-0.7597 \; -0.2128]$ being the linear quadratic optimal. Correspondingly, we set RLMPC parameters to be $\alpha = 10^{-6}$, $\varepsilon = 10^{-8}$, $p = 9$, and $\Phi(x) = [x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_1^2x_2, x_2^3, x_1x_2^2]$. Let the prediction horizon be $N = 3$ and the initial state be $x_0 = [2.9, 2]^{\mathrm{T}}$. We collect 100 random samples in each iteration in the state space $x_{1k} \in [-0.5, 3.5]$, $x_{2k} \in [-0.5, 2.5]$, as well as the actual system trajectory, to form S_k. Our task is to drive the system states from x_0 to the origin while minimizing (3). We examine the performance of RLMPC and compare it with those of traditional MPC and MPCWTC (labeled as "LQR (MPC)" and "MPCWTC," respectively, in the following figures). To facilitate comparison with the optimal control performance in the infinite horizon, we do not impose any constraints on this control problem, at which point the above-designed traditional MPC is fully equivalent to LQR.

The relevant state trajectories and control inputs are reported in Fig. 2. It can be seen that the trajectories of "RLMPC Epi. 1" and "MPCWTC" are very close in the initial few steps, which is due to the fact that the initial policy of the learning episode is essentially the MPCWTC. As the policy of the learning episode improves with each control step k, its trajectory gradually deviates from that of "MPCWTC" and converges to that of "LQR (MPC)." The evolution curves of the weight vector W in the learning episode are shown in Fig. 3. We observe that W converges at the 11th step (at this moment, the system states are not yet regulated to the origin), indicating the convergence of the learning process.

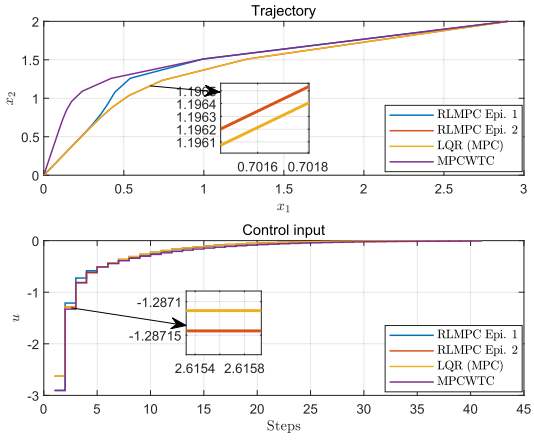

Fig. 2. Trajectories and inputs (linear system).

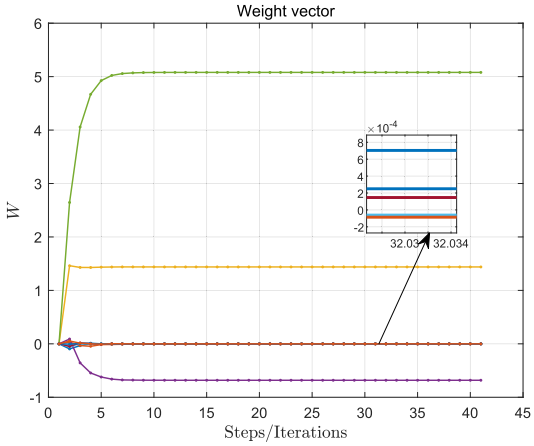

Fig. 3. Convergence of the weight vector (linear system).

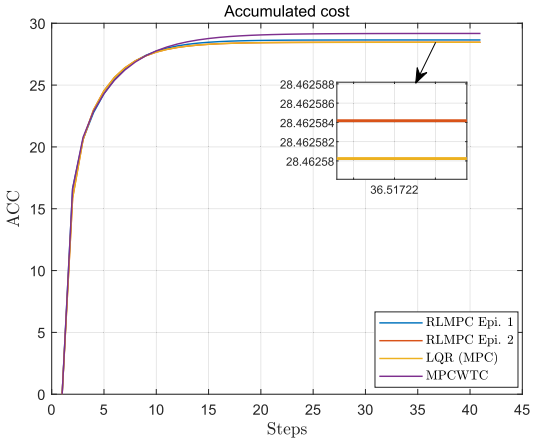

Fig. 4. ACCs (linear system).

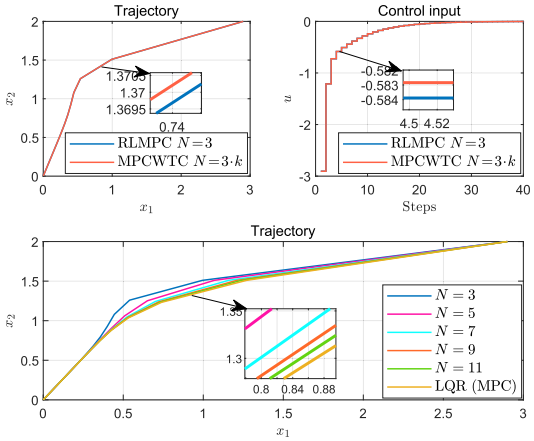

Fig. 5. Trajectories of RLMPC in the learning episode under different prediction horizons (linear system).

Since Algorithm 1 is an example of performing one iteration at each step, we know that the learning process converges at the 11th iteration. To examine the optimality property, we compare the performance of "LQR (MPC)" with that of "RLMPC Epi. 2." It can be seen that their trajectories basically coincide within a certain error range, which verifies the optimality of RLMPC, and the error mainly depends on the selection of ε for terminating the iterations.

Fig. 4 illustrates the accumulated costs (ACCs), which is defined as the sum of the running costs from the initial time to the current time k

$$\text{ACC} = \sum_{i=0}^{k} \ell(x_i, u_i). \tag{35}$$

As expected, the ACC of "MPCWTC" is the highest because it only looks at N steps ahead and ignores all the residual costs in each control period. The ACC of "RLMPC Epi. 1" ranks the second highest as its policy is gradually improving from "MPCWTC." "RLMPC Epi. 2" achieves almost the same minimum as "LQR (MPC)," while the slightly higher value in ACC (about 4×10^{-5}) is mainly due to the precision setting of the iteration termination. In conclusion, the proposed RLMPC approach can deliver the (near-)optimal policy in the sense of an infinite horizon for the control of a linear system.

To verify the statements in Remark 6, we first test the equivalence of RLMPC and the MPCWTC with an accumulating prediction horizon. Since the prediction horizon of RLMPC is set to be $N = 3$, following Algorithm 1 and the results in Section IV-B, it is theoretically equivalent to the MPCWTC with $N = 3 \cdot k$, where $k \in \mathbb{N}_{>0}$ is the step index. We observe from the two subgraphs in the upper part of Fig. 5 that their inputs and hence their trajectories are basically the same, which confirms this equivalence. We then investigate the effectiveness of increasing the prediction horizon by comparing the performance of RLMPC in the learning episode with $N = 3, 5, 7, 9, 11$, respectively. As can be seen from the lower subgraph of Fig. 5, the larger the prediction horizon, the closer the corresponding trajectory is to the optimal trajectory "LQR (MPC)." This can be explained by the fact that the equivalent MPCWTC has an increased accumulating prediction horizon in every iteration, so the policy at each step is closer to the optimal one. Furthermore, we present the ACCs and the CRs (which are measured by the number of iterations required for convergence) in Table I. It is obvious that a larger prediction horizon brings lower ACC and faster convergence in the learning episode, which is consistent with our theoretical results.

We finally investigate the data efficiency of the proposed scheme and compare it with Q-learning [44] and PILCO [45]. The former employs the action-value function and has a very

TABLE I
ACCs AND CRs OF RLMPC IN THE LEARNING EPISODE UNDER DIFFERENT PREDICTION HORIZONS

	$N=3$	$N=5$	$N=7$	$N=9$	$N=11$
ACC	28.6500	28.5454	28.4879	28.4700	28.4648
CR	11 iterations	7 iterations	5 iterations	4 iterations	3 iterations

Benchmark: the ACC of LQR (MPC) is 28.4626.

similar learning process to the proposed scheme, except that it is model-free. The latter is a model-based policy search scheme and is well known for its data efficiency. By implementing Q-learning and PILCO[1] in the optimal control of (34), we observe that the amount of training data required for them to converge to the optimal policy is 1.3×10^4 and 400 (10 trials and 40 training data in each trial), respectively. In contrast, RLMPC only needs 341 training data, far less than Q-learning, and achieves efficiency comparable to that of PILCO. This is mainly because Q-learning utilizes no prior knowledge of the system and therefore requires a large amount of data from interactions with the environment to provide a basis for policy improvement. While PILCO and RLMPC optimize the policy with the assistance of the model, thus significantly reducing the number of interactions and accelerating convergence. In addition, due to the combination with MPC, the significant advantage of RLMPC over PILCO is the theoretical guarantee for the stability and feasibility of the policies generated throughout the learning process.

B. Nonlinear System

Consider the regulation problem of a nonholonomic vehicle system

$$x_{k+1} = x_k + g(x_k)u_k \tag{36}$$

where $x_k = [\chi_k, y_k, \theta_k]^{\mathrm{T}}$ is the state vector, with χ_k, y_k, and θ_k representing the abscissa, ordinate, and yaw angle in the geodetic coordinate system, respectively; $u_k = [v_k, \omega_k]^{\mathrm{T}}$ is the control input, with v_k and ω_k representing the linear velocity and angular velocity, respectively; δ is the sampling interval; and $g(x_k) = \delta \begin{bmatrix} \cos\theta_k & 0 \\ \sin\theta_k & 0 \\ 0 & 1 \end{bmatrix}$. Set $\delta = 0.2$ s, the input constraint $|v_k| \leq 1$ m/s, $|\omega_k| \leq 4$ rad/s, and the state constraint $0 \leq \chi \leq 2$ m. The control objective is to drive the system from $x_0 = [1.98, 5, -\pi/3]^{\mathrm{T}}$ to the origin while minimizing (3) with running cost in the quadratic form $\ell(x_k, u_k) = x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k$, where $Q = \mathrm{diag}\{1, 2, 0.06\}$ and $R = \mathrm{diag}\{0.01, 0.005\}$.

Remark 9: According to Brockett's theory [46], there does not exist smooth and time-invariant feedback to stabilize system (36), which makes the widely adopted design method for terminal conditions, such as [10], no longer applicable to this system, thus posing a great challenge to the traditional MPC design here. Even if qualified terminal conditions can be found, they tend to be quite conservative, as evidenced by a small terminal set, a largely required prediction horizon, and an overestimated terminal cost. This can lead to the conservative performance of the designed traditional MPC.

1) Design of the Traditional MPC and RLMPC: In [47], a traditional MPC is proposed for (36): the terminal cost is $F(x_{N|k}) = \chi_{N|k}^2 + y_{N|k}^2 + \theta_{N|k}^2$, and the terminal set $\mathbb{X}_\Omega$ is defined as $\mathbb{X}_\Omega = \{x_{N|k} \in \mathbb{X} : F(x_{N|k}) \leq \varrho_v\}$, where $\varrho_v = c\varrho$, $\varrho = \min\{\bar{v}^2/\eta^2, \bar{\omega}^2/\xi^2\}$, $c = (1 + T^2 \max\{\eta^2, \xi^2\} - \max\{\eta T^2 + q_{11}/\eta + r_{11}\eta, 2\xi T\}) < 1$, and $\bar{v}$ and $\bar{\omega}$ are the upper bounds of v and ω, respectively, that is, $\bar{v} = 1$ and $\bar{\omega} = 4$, q_{11} and r_{11} are the first elements on the diagonal of Q and R, respectively. By choosing $\eta = \xi = 8$, $N \geq 30$, system (36) can be stabilized to the origin.

Remark 10: This traditional MPC design is computationally demanding because the prediction horizon should be at least 30 to meet the terminal constraint, which confirms Remark 9. Moreover, there is no guarantee that the control performance of the designed MPC is optimal. In contrast, RLMPC avoids artificially designing the terminal conditions and achieves (near-)optimal performance. With the learned (near-)optimal terminal cost, its prediction horizon can be flexibly adjusted without affecting its performance. Therefore, it is expected to show unique superiority in this problem.

Now we present the RLMPC design: $\alpha = 10^{-6}$, $\varepsilon = 10^{-7}$, $p = 30$, and the basis functions are chosen as polynomials of order one to six. We generate 100 random samples in each iteration in the state space $\chi_k \in [0, 2]$, $y_k \in [-1, 6]$, $\theta_k \in [-\pi, \pi]$, together with the actual system trajectory, to form S_k.

2) Comparison of the Traditional MPC and RLMPC: In the following simulations, the prediction horizons of the traditional MPC and RLMPC are set to 30 and 5 steps, respectively.

As we can see from Figs. 6 and 7 that no constraint is violated during both episodes, which demonstrates that RLMPC is able to restrict the policy within the input and state constraints and thus perform safely. Fig. 8 illustrates the convergence of the weight W. We observe that it takes 25 iterations to converge, and the evolution of the LVF is also visualized. Consistent with Theorem 1, this evolution exhibits the uniformly monotonically nondecreasing property in this process. To verify Theorem 4, we apply the policies obtained in the first 25 iterations of the learning episode, respectively, to the system and show the corresponding trajectories and ACCs [as defined in (35)] in Fig. 9. We observe that each policy generates a safe and stabilizing trajectory to the origin and that the ACCs show a gradual improvement, which also corroborates Theorem 1. The comparison of the two episodes of RLMPC and traditional MPC is shown in Fig. 7. Since the traditional MPC design for nonlinear systems does not ensure optimality, RLMPC could achieve better performance (15.09% less ACC) even in the learning episode. There is a further 16% reduction in ACC in the re-execution episode where a near-optimal policy is employed.

3) Prediction Horizon and Computational Burden: We now demonstrate the relationship between prediction horizon, control performance, convergence, and computational burden and use the following three metrics in the quantitative evaluation:

[1]The Q-learning scheme is based on the adaptation of the program provided in *https://github.com/XiaozhuFang/Simple-Comparison-between-Q-learning-and-MPC*, while the PILCO scheme is based on the adaptation of the "cart-pole" example provided in *https://github.com/UCL-SML/pilco-MATLAB*. We mainly modify the system and rewards to suit this example, leaving most of the parameters unchanged.

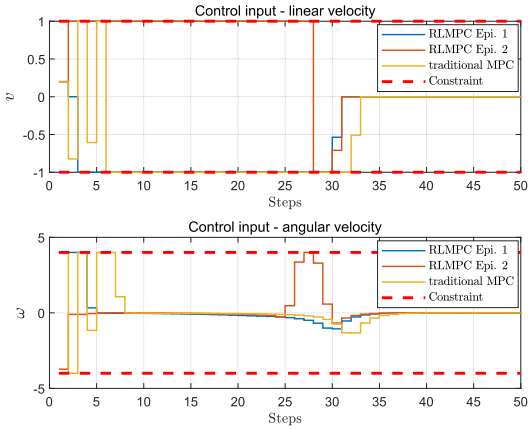

Fig. 6. Inputs of RLMPC (nonlinear system).

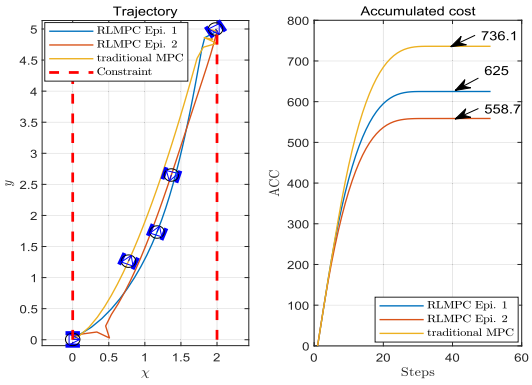

Fig. 7. System trajectories and ACCs of traditional MPC and RLMPC (nonlinear system).

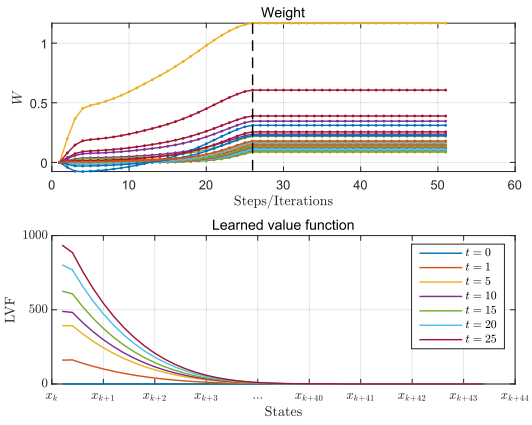

Fig. 8. Convergence of the weight vector and LVF (nonlinear system).

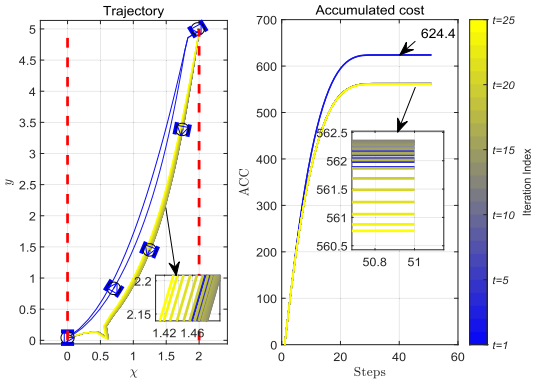

Fig. 9. System trajectories and ACCs of RLMPC policies in the 1st–25th iterations of the learning episode (nonlinear system).

TABLE II

COMPARISON OF TRADITIONAL MPC AND RLMPC IN THE LEARNING EPISODE WITH DIFFERENT PREDICTION HORIZONS

Method	RLMPC Epi. 1				Traditional MPC
	$N = 5$	$N = 7$	$N = 9$	$N = 30$	$N = 30$
ACC	625.0301	624.1901	623.7521	560.2312	736.0854
CR	25 iterations	24 iterations	23 iterations	8 iterations	\
ACT	0.0826s	0.0904s	0.1143s	0.3991s	0.4126s

The above simulation is implemented using Matlab R2018b on a computer equipped with a 2.60 GHz Intel Core i7-9750H CPU, 8 GB RAM and 64-bit Windows 10 operating system. The optimization problem is solved by IPOPT (version 3.11.8).

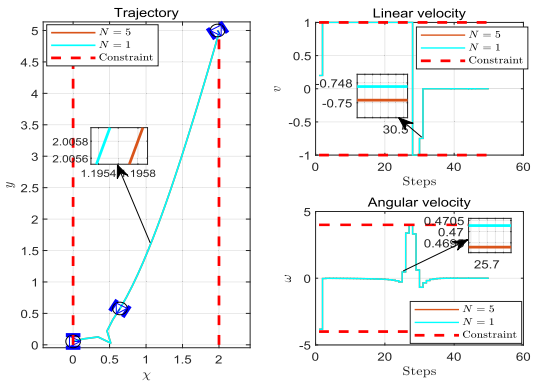

Fig. 10. Comparison of trajectories and inputs of RLMPC with the learned (near-)optimal value function under prediction horizons $N = 1$ and $N = 5$ (nonlinear system).

ACC, CR, and the average computational time (ACT) at each step. RLMPC in the learning episode with four different prediction horizons, namely $N = 5, 7, 9, 30$, are compared (settings of other parameters remain unchanged), as well as that of traditional MPC with $N = 30$. Corresponding numerical results are listed in Table II.

Consistent with the linear system example, increasing the prediction horizon of RLMPC effectively enhances CR and leads to an improved (lower ACC) trajectory in the learning episode. Conceivably, this comes at the cost of increased computation time at each step. However, we observe that even with the same prediction horizon of 30, the ACT of RLMPC is still slightly lower than that of traditional MPC. This is mainly due to the removal of the terminal constraint, which simplifies the optimization problem.

We now perform a verification of Remark 7. Taking the learned (near-)optimal value function as the terminal cost of RLMPC, we reduce the prediction horizon to 1 and re-execute the regulation task. As we can see from Fig. 10, the trajectories and input under $N = 1$ are basically coincide with those under $N = 5$, which confirms that shortening the prediction horizon does not affect the (near-)optimal performance. More importantly, the ACT time is reduced from 0.0826 to 0.0222 s (about an improvement of 73%), which illustrates the effectiveness of RLMPC in reducing the computational burden.

VII. Conclusion

To free the design of the terminal cost, terminal controller, and terminal set in traditional MPC and to improve the closed-loop performance, this article developed a new RLMPC scheme for discrete-time systems by combining RL and MPC through PI. The core of this scheme is to take a value function, which is obtained through learning, as the terminal cost of traditional MPC to stabilize the system. We showed that the value function monotonically converges to the optimal one under PI. Based on this property, the closed-loop stability of RLMPC was further proved. We tested the proposed scheme on a linear system example to show its convergence, safety, optimality, and stability. Furthermore, we also implemented it on a nonholonomic vehicle system to show that RLMPC outperforms traditional MPC in the nonlinear case. Finally, we discussed the impact of the prediction horizon on the learning process and the control performance of RLMPC, highlighting the ability of RLMPC to flexibly adjust the prediction horizon to reduce the computational burden.

References

[1] T. Wang, H. Gao, and J. Qiu, "A combined adaptive neural network and nonlinear model predictive control for multirate networked industrial process control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 416–425, Feb. 2016.

[2] M. L. Darby and M. Nikolaou, "MPC: Current practice and challenges," *Control Eng. Pract.*, vol. 20, no. 4, pp. 328–342, Apr. 2012.

[3] M. Neunert et al., "Whole-body nonlinear model predictive control through contacts for quadrupeds," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 1458–1465, Jul. 2018.

[4] Z. Li, Y. Xia, C.-Y. Su, J. Deng, J. Fu, and W. He, "Missile guidance law based on robust model predictive control using neural-network optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1803–1809, Aug. 2015.

[5] J. Kabzan et al., "AMZ driverless: The full autonomous racing system," *J. Field Robot.*, vol. 37, no. 7, pp. 1267–1294, 2020.

[6] Z. Li, W. Yuan, Y. Chen, F. Ke, X. Chu, and C. L. P. Chen, "Neural-dynamic optimization-based model predictive control for tracking and formation of nonholonomic multirobot systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 12, pp. 6113–6122, Dec. 2018.

[7] C. C. Chen and L. Shaw, "On receding horizon feedback control," *Automatica*, vol. 18, no. 3, pp. 349–352, May 1982.

[8] M. Alamir and G. Bornard, "On the stability of receding horizon control of nonlinear discrete-time systems," *Syst. Control Lett.*, vol. 23, no. 4, pp. 291–296, Oct. 1994.

[9] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Madison, WI, USA: Nob Hill Pub., 2009.

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

[11] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[12] M. Reble, *Model Predictive Control for Nonlinear Continuous-Time Systems With and Without Time-delays*. Berlin, Germany: Logos Verlag Berlin GmbH, 2013.

[13] A. Boccia, L. Grüne, and K. Worthmann, "Stability and feasibility of state constrained MPC without stabilizing terminal constraints," *Syst. Control Lett.*, vol. 72, pp. 14–21, Oct. 2014.

[14] D. Silver et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[15] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[16] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.

[17] R. Kamalapurkar, L. Andrews, P. Walters, and W. E. Dixon, "Model-based reinforcement learning for infinite-horizon approximate optimal tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 753–758, Mar. 2017.

[18] Y. Li, "Reinforcement learning applications," 2019, *arXiv:1908.06973*.

[19] M. Wiering and M. V. Otterlo, *Reinforcement Learning*, vol. 12. Berlin, Germany: Springer, 2012.

[20] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988.

[21] G. Tesauro, "Practical issues in temporal difference learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 257–277, May 1992.

[22] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Trans. Autom. Control*, vol. 42, no. 5, pp. 674–690, May 1997.

[23] J. García and F. Fernández, "A comprehensive survey on safe reinforcement learning," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 1437–1480, 2015.

[24] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 908–918.

[25] M. Gallieri et al., "Safe interactive model-based learning," 2019, *arXiv:1911.06556*.

[26] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, "Plan online, learn offline: Efficient learning and exploration via model-based control," 2018, *arXiv:1811.01848*.

[27] F. Farshidian, D. Hoeller, and M. Hutter, "Deep value model predictive control," 2019, *arXiv:1910.03358*.

[28] K. Napat, 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 *Proc. Annu. Conf. Learn. Dyn. Control*, 2020, pp. 211–224.

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

[30] M. Zanon, S. Gros, and A. Bemporad, "Practical reinforcement learning of stabilizing economic MPC," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2258–2263.

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

[32] S. V. Raković and W. S. Levine, *Handbook of Model Predictive Control*. Berlin, Germany: Springer, 2018.

[33] D. Limón, T. Alamo, F. Salas, and E. F. Camacho, "On the stability of constrained MPC without terminal constraint," *IEEE Trans. Autom. Control*, vol. 51, no. 5, pp. 832–836, May 2006.

[34] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits Syst. Mag.*, vol. 9, no. 3, pp. 32–50, Aug. 2009.

[35] D. E. Kirk, *Optimal Control Theory: An Introduction*. Chelmsford, MA, USA: Courier Corporation, 2004.

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

[37] L. Grüne, "NMPC without terminal constraints," *IFAC Proc. Volumes*, vol. 45, no. 17, pp. 1–13, 2012.

[38] L. Grüne, "Analysis and design of unconstrained nonlinear MPC schemes for finite and infinite dimensional systems," *SIAM J. Control Optim.*, vol. 48, no. 2, pp. 1206–1228, Jan. 2009.

[39] K. Worthmann, M. W. Mehrez, M. Zanon, G. K. I. Mann, R. G. Gosine, and M. Diehl, "Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs," *IEEE Trans. Control Syst. Technol.*, vol. 24, no. 4, pp. 1394–1406, Jul. 2016.

[40] M. W. Mehrez, K. Worthmann, J. P. V. Cenerini, M. Osman, W. W. Melek, and S. Jeon, "Model predictive control without terminal constraints or costs for holonomic mobile robots," *Robot. Auto. Syst.*, vol. 127, May 2020, Art. no. 103468.

[41] B. A. Finlayson, *The method of Weighted Residuals and Variational Principles*. Philadelphia, PA, USA: SIAM, 2013.

[42] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 3, pp. 621–634, Mar. 2014.

[43] J. Bibby, "Axiomatisations of the average and a further generalisation of monotonic sequences," *Glasgow Math. J.*, vol. 15, no. 1, pp. 63–65, Mar. 1974.

[44] J. Clifton and E. Laber, "Q-learning: Theory and applications," *Annu. Rev. Statist. Appl.*, vol. 7, pp. 279–301, Mar. 2020.

[45] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 408–423, Feb. 2015.

[46] R. W. Brockett, "Asymptotic stability and feedback stabilization," *Differ. Geometric Control Theory*, vol. 27, no. 1, pp. 181–191, Dec. 1983.

[47] Y. Zhu and U. Ozguner, "Robustness analysis on constrained model predictive control for nonholonomic vehicle regulation," in *Proc. Amer. Control Conf.*, 2009, pp. 3896–3901.

Min Lin received the B.S. degree in automation from the Beijing Institute of Technology, Beijing, China, in 2018, where he is currently pursuing the Ph.D. degree with the School of Automation.

His research interests cover model predictive control, machine learning, active disturbance rejection control, and robotic systems.

Zhongqi Sun (Member, IEEE) was born in Hebei, China, in 1986. He received the B.E. degree in computer and automation from Hebei Polytechnic University, Hebei, in 2010, and the Ph.D. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2018.

From 2018 to 2019, he was a Post-Doctoral Fellow with the Faculty of Science and Engineering, University of Groningen, Groningen, The Netherlands. He is currently an Assistant Professor with the School of Automation, Beijing Institute of Technology. His research interests include multiagent systems, model predictive control, machine learning, and robotic systems.

Yuanqing Xia (Senior Member, IEEE) received the Ph.D. degree in control theory and control engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001.

From 2002 to 2003, he was a Post-Doctoral Research Associate with the Institute of Systems Science, Academy of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing. From 2003 to 2004, he was with the National University of Singapore, Singapore, as a Research Fellow, where he worked on variable structure control. From 2004 to 2006, he was with the University of Glamorgan, Pontypridd, U.K., as a Research Fellow. From 2007 to 2008, he was a Guest Professor with Innsbruck Medical University, Innsbruck, Austria. Since 2004, he has been with the School of Automation, Beijing Institute of Technology, Beijing, first as an Associate Professor, then, since 2008, as a Professor. His current research interests are in the fields of networked control systems, robust control and signal processing, and active disturbance rejection control.

Jinhui Zhang received the Ph.D. degree in control science and engineering from the Beijing Institute of Technology, Beijing, China, in 2011.

He was a Research Associate with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong, in 2010, as a Senior Research Associate with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Hong Kong, from 2010 to 2011, and a Visiting Fellow with the School of Computing, Engineering and Mathematics, University of Western Sydney, Sydney, NSW, Australia, in 2013. He was an Associate Professor with the Beijing University of Chemical Technology, Beijing, from 2011 to 2016, and a Professor with the School of Electrical and Automation Engineering, Tianjin University, Tianjin, China, in 2016. He joined the Beijing Institute of Technology in 2016, where he is currently a Professor. His research interests include networked control systems, composite disturbance rejection control, and biomedical engineering.