Recurrent Model Predictive Control: Learning an Explicit Recurrent Controller for Nonlinear Systems

Zhengyu Liu, Jingliang Duan, Wenyuan Wang, Shengbo Eben Li, Senior Member, IEEE, Yuming Yin, Ziyu Lin, and Bo Cheng

Abstract—This article proposes an offline control algorithm, called recurrent model predictive control, to solve large-scale nonlinear finite-horizon optimal control problems. It can be regarded as an explicit solver of traditional model predictive control (MPC) algorithms, which can adaptively select appropriate model prediction horizon according to current computing resources, so as to improve the policy performance. Our algorithm employs a recurrent function to approximate the optimal policy, which maps the system states and reference values directly to the control inputs. The output of the learned policy network after \( N \) recurrent cycles corresponds to the nearly optimal solution of \( N \)-step MPC. A policy optimization objective is designed by decomposing the MPC cost function according to the Bellman’s principle of optimality. The optimal recurrent policy can be obtained by directly minimizing the designed objective function, which is applicable for general nonlinear and noninput-affine systems. Both simulation-based and real-robot path-tracking tasks are utilized to demonstrate the effectiveness of the proposed method.

Index Terms—Dynamic programming, model predictive control (MPC), recurrent function.

I. INTRODUCTION

MODEL predictive control (MPC) is a well-known method to solve finite-horizon optimal control problems online, which has been extensively investigated in various fields [1]–[3]. However, existing MPC algorithms still suffer from a major challenge: relatively low computation efficiency [4].

One famous approach to tackle this issue is the moving blocking technique, which assumes constant control input in a fixed portion of the prediction horizon. It increases the computation efficiency by reducing the number of variables to be optimized [5]. This solution cannot guarantee control performance, system stability, and constraint satisfaction. In addition, Wang and Boyd (2009) proposed an early termination interior-point method to reduce the calculation time by limiting the maximum number of iterations per time step [6]. However, these online methods are still unable to meet the online computing requirement for nonlinear and large-scale systems.

Some control algorithms choose to calculate a near-optimal explicit policy offline, and then implement it online. Bemporad et al. first proposed the explicit MPC method to increase the computation efficiency, which partitioned the constrained state space into several regions and calculated explicit feedback control laws for each region [7]. During online implementation, the onboard computer only needs to choose the corresponding state feedback control law according to the current system state, thereby reducing the burden of online calculation to some extent. Such algorithms are only suitable for small-scale systems, since the required storage capacity grows exponentially with the state dimension [8].

Furthermore, significant efforts have been devoted to approximation MPC algorithms, which can reduce polyhedral state regions and simplify explicit control laws. Geyer et al. provided an optimal merging approach to reduce partitions via merging regions with the same control law [9]. Jones et al. proposed a polytopic approximation method using double description and barycentric functions to estimate the optimal policy, which greatly reduced the partitions and could be applied to any convex problem [10]. Wen et al. proposed a piecewise continuous grid function to represent an explicit MPC solution, which reduced the requirements of storage capacity and improved online computation efficiency [11]. Borrelli et al. proposed an explicit MPC algorithm, which can be executed partially online and partially offline [12]. In addition, some MPC studies employed a parameterized function to approximate the MPC controller. They updated the function parameters...
by minimizing the MPC cost function with a fixed prediction horizon through supervised learning or reinforcement learning [13]–[15].

Note that the policy performance and the computation time for each step usually increase with the length of the prediction horizon. The above-stated algorithms usually have to make a tradeoff between control performance and computation time, and then select a conservative fixed prediction horizon to meet the requirement of real-time decision-making. However, onboard computing resources are usually dynamically changing, so these algorithms usually lead to calculation timeouts or resources waste. In other words, these algorithms cannot adapt to the dynamic allocation of computing resources and make full use of the available computing time to select the longest model prediction horizon.

In this article, we propose an offline MPC algorithm, called recurrent MPC (RMPC), for finite-horizon optimal control problems with large-scale nonlinearities and nonaffine inputs. Our main contributions can be summarized as follows:

1) A recurrent function is employed to approximate the optimal policy, which maps the system states and reference values directly to the control inputs. Compared to previous algorithms employing nonrecurrent functions (such as fully connected neural networks), which are only suitable for fixed-horizon predictive control [13]–[15], the inclusion of the recurrent structure allows the algorithm to select an appropriate prediction horizon according to current computing resources. In particular, the output of the learned policy function after \( N \) recurrent cycles corresponds to the nearly optimal solution of \( N \)-step MPC.

2) A policy optimization objective is designed by decomposing the MPC cost function according to the Bellman’s principle of optimality. The optimal recurrent policy can be obtained by directly minimizing the designed objective function. Therefore, unlike most explicit MPC algorithms [7]–[12] that can only handle linear systems, the proposed algorithm is applicable for nonlinear and noninput-affine systems.

3) RMPC calculates a near-optimal recurrent policy offline, and then implements online. Experiment shows that it is over five times faster than the traditional online MPC algorithms [16], [17] under the same problem scale.

This article is organized as follows. In Section II, we provide the preliminaries of the MPC problem. Section III presents the RMPC algorithm and proves its convergence. In Section IV, we perform a hardware-in-the-loop (HIL) simulation to demonstrate the generalizability and effectiveness of RMPC. Section V verifies the performance of RMPC in a four-wheeled robot, and Section VI concludes this article.

II. PRELIMINARIES

Consider the general time-invariant discrete-time dynamic system

\[
x_{i+1} = f(x_i, u_i)
\]

with state \( x_i \in \mathcal{X} \subset \mathbb{R}^n \), control input \( u_i \in \mathcal{U} \subset \mathbb{R}^m \), and the system dynamics function \( f : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n \). We assume that \( f(x_i, u_i) \) is Lipschitz continuous on a compact set \( \mathcal{X} \), and the system is stabilizable on \( \mathcal{X} \).

The \( N \)-step MPC problem without state constraints is given as

\[
\min_{u_0^N, \ldots, u_{N-1}^N} \ V(x_0, r_{1:N}, N) = \sum_{i=1}^{N} l(x_i, r_i, u_{i-1}^N(x_0, r_{1:N}))
\]

s.t., \( 1), \quad u \in \mathcal{U}, \quad (2) \]

where \( V(x_0, r_{1:N}, N) \) is the cost function, \( x_0 \) is initial state, \( N \) is length of prediction horizon, \( r_{1:N} = [r_1, r_2, \ldots, r_N] \) is reference trajectory, \( u_{i-1}^N \) is the control input of the \( i \)th step in \( N \)-step prediction, and \( l \geq 0 \) is the utility function. The purpose of MPC is to find the optimal control sequence to minimize the objective \( V(x_0, r_{1:N}, N) \), which can be denoted as

\[
\left[\begin{array}{c}
u_0^N(x_0, r_{1:N}), u_1^N(x_0, r_{1:N}), \ldots, u_{N-1}^N(x_0, r_{1:N})\end{array}\right] = \arg\min_{u_0^N, \ldots, u_{N-1}^N} V(x_0, r_{1:N}, N) \tag{3}
\]

where the superscript \(^*\) represents optimum.

III. RECURRENT MODEL PREDICTIVE CONTROL

A. Recurrent Policy Function

In practical applications, we only need to execute the first control input \( u_0^N(x_0, r_{1:N}) \) of the optimal sequence in (3) at each time step. Given a control problem, assume that \( N_{\text{max}} \) is the maximum feasible prediction horizon. We aim to make full use of computation resources and adaptively select the longest prediction horizon \( k \in [1, N_{\text{max}}] \), which means we need to calculate and store the optimal control input \( u_0^N(x, r_{1:k}) \) of \( \forall x \in \mathcal{X}, \forall r_{1:k}, \) and \( \forall k \in [1, N_{\text{max}}] \) in advance. This requires us to find an efficient way to represent the policy for different prediction horizon \( N \in [1, N_{\text{max}}] \) and solve it offline.

We first introduce a recurrent function, denoted as \( \pi^e(x_0, r_{1:c}; \theta) \), to approximate the control input \( u_0^N(x_0, r_{1:c}) \), where \( \theta \) is the vector of function parameters and \( c \) is the number of recurrent cycles of the policy function. The goal of the proposed RMPC algorithm is to find the optimal parameters \( \theta^* \), such that

\[
\pi^e(x_0, r_{1:c}; \theta^*) = u_0^N(x_0, r_{1:c}) \quad \forall x_0 \in \mathcal{X}, \forall r_{1:c}, \forall c \in [1, N_{\text{max}}]. \tag{4}
\]

The structure of the recurrent policy function is illustrated in Fig. 1. All recurrent cycles share the same parameters \( \theta \), where \( h_c \in \mathbb{R}^q \) is the vector of hidden states.

Each recurrent cycle is mathematically described as

\[
\begin{align*}
h_c &= \sigma_h(x_0, r_c, h_{c-1}; \theta_h) \\
\pi^e(x_0, r_{1:c}; \theta) &= \sigma_y(h_c; \theta_y) \\
c &\in [1, N_{\text{max}}], \quad \theta = \theta_h \cup \theta_y
\end{align*}
\]

(5)
where \( h_0 = 0 \), \( \sigma_h \), and \( \sigma_y \) are activation functions of hidden layer and output layer, respectively.

As shown in Fig. 1, the recurrent policy function outputs a control input at each recurrent cycle. Assuming that we have found the optimal parameters \( \theta^* \), it follows that the output of the \( c \)th cycle \( \pi^c(x_0, r_{1:c}; \theta^*) = u_0^{N,i} (x_0, r_{1:c}) \) for \( \forall c \in [1, N_{\text{max}}] \). This indicates that the more cycles, the longer the prediction horizon. In practical applications, the calculation time of each cycle \( t_c \) is different due to the dynamic change of computing resource allocation (see Fig. 2). At each time step, the total time assigned to the control input calculation is assumed to be \( T \). Denoting the final number of the recurrent cycles at each time step as \( k \), then the corresponding control input is \( \pi^k(x_0, r_{1:k}; \theta^*) \), where

\[
k = \begin{cases} 
N_{\text{max}}, & \sum_{c=1}^{N_{\text{max}}} t_c \leq T \\
p, & \sum_{c=1}^{p} t_c \leq T < \sum_{c=1}^{p+1} t_c 
\end{cases}
\]

Fig. 2. Maximum recurrent cycles in different cases.

Therefore, the recurrent policy is able to make full use of computing resources and adaptively select the longest prediction step \( k \). In other word, the more computing resources allocated, the longer prediction horizon will be selected, which would usually lead to better control performance.

Remark 1: Existing MPC algorithms usually employ nonrecurrent approximation functions to represent the policies [13]–[15], which must select a fixed prediction horizon in advance. When the prediction horizon changes, the optimization problem must be reconstructed to learn a new corresponding policy. Conversely, RMPC employs recurrent function to approximate the optimal policy, which maps the system states and reference values directly to the control inputs. The use of recurrent structure allows RMPC to select an appropriate model prediction horizon according to current computing resources. The output of the learned policy network after \( N \) recurrent cycles corresponds to the nearly optimal solution of \( N \)-step MPC.

B. Objective Function for Policy Learning

To find the optimal parameters \( \theta^* \) offline, we first need to represent the MPC cost function in (2) in terms of \( \theta \), denoted by \( V(x_0, r_{1:N}, N; \theta) \). From (2) and the Bellman’s principle of optimality, the global minimum \( V^*(x_0, r_{1:N}, N) \) can be expressed as

\[
V^*(x_0, r_{1:N}, N) = l(x_1, r_1, u_0^{N,i}(x_0, r_{1:N})) + V^*(x_1, r_{2:N}, N-1)
\]

\[
= \sum_{i=1}^{2} l(x_i, r_i, u_0^{N-i+1,i}(x_{i-1}, r_{i:N})) + V^*(x_{2}, r_{3:N}, N-2)
\]

\[
\vdots
\]

\[
= \sum_{i=1}^{N-1} l(x_i, r_i, u_0^{N-i+1,i}(x_{i-1}, r_{i:N})) + V^*(x_{N-1}, r_{N}, 1)
\]

\[
= \sum_{i=1}^{N} l(x_i, r_i, u_0^{N-i+1,i}(x_{i-1}, r_{i:N})).
\]

Furthermore, according to (2), one has

\[
V^*(x_0, r_{1:N}, N) = \sum_{i=1}^{N} l(x_i, r_i, u_{i-1}^{N,i}(x_0, r_{1:N}))
\]

\[
= \sum_{i=1}^{N} l(x_i, r_i, u_0^{N-i+1,i}(x_{i-1}, r_{i:N})).
\]

Therefore, for the same \( x_0 \) and \( r_{1:N} \), it is clear that

\[
u_{i-1}^{N,i}(x_0, r_{1:N}) = u_0^{N-i+1,i}(x_{i-1}, r_{i:N}), \quad \forall i \in [1, N].
\]

This indicates that the optimal control input \( u_{i-1}^{N,i}(x_0, r_{1:N}) \) in (3) can be regarded as the optimal control input of the \( N-i+1 \)-step MPC control problem with initial state \( x_{i-1} \). Hence, by replacing all \( u_{0}^{N,i}(x_0, r_{1:N}) \) in (2) with \( u_0^{N-i+1,i}(x_{i-1}, r_{i:N}) \), the cost function of \( N \)-step MPC can be rewritten as

\[
V(x_0, r_{1:N}, N) = \sum_{i=1}^{N} l(x_i, r_i, u_0^{N-i+1,i}(x_{i-1}, r_{i:N})).
\]

Immediately, we can obtain the \( N \)-step cost function in terms of \( \theta \)

\[
V(x_0, r_{1:N}, N; \theta) = \sum_{i=1}^{N} l(x_i, r_i, \pi^{N-i+1,i}(x_{i-1}, r_{i:N}; \theta)).
\]

Fig. 3 illustrates the reshaped \( N \)-step cost function intuitively.
Fig. 3. Illustration of the objective function of RMPC.

To find the optimal parameters $\theta^*$ that make (4) hold, we can construct the following objective function:

$$J(\theta) = \mathbb{E}_{x_0 \in X, r_1: N_{\text{max}}} \left\{ V(x_0, r_1: N_{\text{max}}, N_{\text{max}}^i; \theta) \right\}. \quad (9)$$

Therefore, we can update $\theta$ by directly minimizing $J(\theta)$. The policy update gradients can be derived as

$$\frac{dJ}{d\theta} = \mathbb{E}_{x_0 \in X, r_1: N_{\text{max}}} \left\{ \frac{dV(x_0, r_1: N_{\text{max}}, N_{\text{max}}; \theta)}{d\theta} \right\} \quad (10)$$

where

$$\frac{dV(x_0, r_1: N_{\text{max}}, N_{\text{max}}; \theta)}{d\theta} = \sum_{i=1}^{N_{\text{max}}} \frac{d}{d\theta} l(x_i, r_i, \pi_{N_{\text{max}}^i}; \theta).$$

Denoting $l(x_i, r_i, \pi_{N_{\text{max}}^i}; \theta)$ as $l_i$, $\frac{dx_i}{d\theta}$ as $\phi_i$ and $\frac{d\pi_{N_{\text{max}}^i}}{d\theta}$ as $\psi_i$, we further have

$$\frac{dV(x_0, r_1: N_{\text{max}}, N_{\text{max}}; \theta)}{d\theta} = \sum_{i=1}^{N_{\text{max}}} \left\{ \frac{\partial l_i}{\partial x_i} \phi_i + \frac{\partial l_i}{\partial \pi_{N_{\text{max}}^i}} \psi_i \right\}.$$

with $\phi_i = 0$, and

$$\psi_i = \frac{\partial \pi_{N_{\text{max}}^i}}{\partial x_i} \phi_i + \frac{\partial \pi_{N_{\text{max}}^i}}{\partial \theta}.$$

Fig. 4 visually shows the backpropagation path of the policy gradients. Taking the gradient descent (GD) method as an example, the policy update rule is

$$\theta_{K+1} = -\alpha_{\theta} \frac{dJ}{d\theta} + \theta_K \quad (11)$$

where $\alpha_{\theta}$ denotes the learning rate and $K$ indicates $K$th iteration.

Remark 2: Most existing explicit MPC algorithms [7]–[12] can only handle linear systems. As a comparison, RMPC is applicable for general nonlinear and noninput-affine systems, because the nearly optimal recurrent policy can be obtained by

Algorithm 1: RMPC Algorithm.

Given an appropriate learning rate $\alpha_{\theta}$ and an arbitrarily small positive number $\epsilon$.

Initial with arbitrary $\theta_0$

repeat

Randomly select $x_0 \in X$ and the corresponding $r_1: N_{\text{max}}$

Calculate $\frac{dJ(\theta_K)}{d\theta}$ using (10)

Update policy function using (11)

until $|J(\theta_{K+1}) - J(\theta_K)| \leq \epsilon$

End

Fig. 5. Training flowchart of RMPC.
directly minimizing the designed objective function using policy gradient methods.

Remark 3: The objective function of RMPC in (9) is identical to the traditional MPC problem (2). The only difference between them is that RMPC aims to find an explicit nearly optimal recurrent policy rather than numerical solutions. Therefore, RMPC can be regarded as a special explicit solver for the traditional MPC problem in (2).

C. Convergence and Optimality

There are many types of recurrent functions belonging to the structure defined in (5), and the recurrent neural network (RNN) is the most commonly used one. In recent years, deep RNNs have been successfully implemented in many fields, such as natural language processing and system control, attributing to their ability to process sequential data [18], [19]. Next, we assume that the iteration index \( K \to \infty \), the optimal policy \( \pi^* (x_0, r_{1:K}; \theta^*) \) that makes (4) hold can be achieved using Algorithm 1, as long as \( \pi^* (x_0, r_{1:K}; \theta^*) \) is an overparameterized RNN. The overparameterization means that the number of hidden neurons and layers is sufficiently large. Before the main theorem, the following lemma and assumption need to be introduced.

Lemma 1 (Universal Approximation Theorem [20]–[22]): Consider a sequence of finite functions \( \{ F^i (y^i) \}_{i=1}^n \), where \( n \) is the number of functions, \( y^i = [y_1, y_2, \ldots, y_i] \in \mathbb{R}^i, i \in [1, n] \) is the input dimension, and \( F^i (y^i) : \mathbb{R}^i \to \mathbb{R}^d \) is a continuous function on a compact set. Describe an RNN \( G^\theta (y^i; W, b) \) as

\[
\begin{align*}
   h_c &= \sigma_h (W_h y^i + U_h h_{c-1} + b_h) \\
   G^\theta (y^i; W, b) &= \sigma_y (W_h^T h_c + b_h)
\end{align*}
\]

where \( c \) is the number of recurrent cycles, \( W = W_h \cup W_y, b = b_h \cup b_y, \) and \( U_h, \sigma_h \) and \( \sigma_y \) are activation functions. Supposing \( G^\theta (y^i; W, b) \) is overparameterized, for any \( F^i (y^i) \}_{i=1}^n, \exists U_h, W, b, \) such that

\[
\| G^\theta (y^i; W, b) - F^i (y^i) \|_\infty \leq \epsilon, \quad \forall y^i \in \mathbb{R}^c, c \in [1, n]
\]

where \( \epsilon \in \mathbb{R}^+ \) is an arbitrarily small error.

The reported experimental results and theoretical proofs have shown that the straightforward optimization methods, such as GD and stochastic GD (SGD), can find global minima of most training objectives in polynomial time if the approximate function is an overparameterized NN or RNN [23], [24]. Based on this fact, we make the following assumption.

Assumption 1: If the approximate function is an overparameterized RNN, the global minimum of the objective function in (9) can be found using an appropriate optimization algorithm such as SGD [25].

Now, we are ready to show the convergence and optimality of RMPC.

Theorem 1 (RMPC): Suppose \( \pi^* (x_0, r_{1:K}; \theta^*) \) is an overparameterized RNN. Through Algorithm 1, any initial parameters \( \theta_0 \) will converge to \( \theta^* \), such that (4) holds.

Proof: By (9), we have

\[
\min_{\theta} J(\theta) = \min_{\theta} \mathbb{E}_{x_0 \in X, r_{1:N_{\text{max}}}} \left\{ V(x_0, r_{1:N_{\text{max}}}; \theta) \right\}
\]

\[
\geq \mathbb{E}_{x_0 \in X, r_{1:N_{\text{max}}}} \left\{ \min_{\theta} V(x_0, r_{1:N_{\text{max}}}; \theta) \right\}.
\]

By Lemma 1, there always \( \exists \theta^* \), such that

\[
\theta^* = \arg \min_{\theta} V(x_0, r_{1:N_{\text{max}}}; \theta), \quad \forall x_0 \in X, r_{1:N_{\text{max}}}
\]

Then, it directly follows that:

\[
J(\theta^*) = \min_{\theta} J(\theta).
\]

Furthermore, according to (6) and (7), and the Bellman’s principle of optimality, \( \theta^* \) can also make (4) hold, i.e., \( \theta^* = \theta^* \).

Note that \( \theta^* \) may not be unique. From Assumption 1, we can always find \( \theta^* \) by repeatedly minimizing \( J(\theta) \) in (9) using (11), which completes the proof.

Thus, we have proven that the RMPC algorithm can converge to \( \theta^* \). In other words, it can find the explicitly near-optimal policy of MPC with different prediction horizons, whose output after \( \epsilon \)th recurrent cycles corresponds to the nearly optimal solution of \( \epsilon \)-step MPC.

Remark 4: Theorem 1 shows that under mild assumptions, the output of the converged policy of RMPC is exactly the optimal solution of the traditional MPC problem in (2). This means that if RMPC converges to the nearly optimal solution, it would inherit the stability property of the original MPC problem. The stability conditions for the case where the approximate error of the learned recurrent policy cannot be ignored will be established in further studies.

IV. SIMULATION VERIFICATION

In order to evaluate the performance of the proposed RMPC algorithm, we choose the vehicle lateral control problem in the path tracking task as an example [26]. It is a nonlinear and nonaffine control problem and a widely used verification and application task for MPC [27]–[29].

A. Overall Settings

The recurrent policy network is trained offline on the PC, and then deployed to the industrial personal computer (IPC). The vehicle dynamics used for policy training are different from the controlled plant, which is provided by the Carsim simulator [30]. For online applications, the IPC-controller gives the control signal to the plant according to the state information and the reference trajectory. The plant feeds back the state information to the IPC-controller, so as to realize the closed-loop control process. The feedback scheme of the HIL experiment is depicted in Fig. 6. The type of IPC-controller is ADLINK MXE-5501, equipped with Intel i7-6820EQ CPU and 8 GB RAM, which is used as a vehicle onboard controller [31]. The plant is a real-time system, simulated by the vehicle dynamic model of CarSim. The longitudinal speed is assumed to be constant, \( v_z = 16 \text{ m/s} \), and the expected trajectory is shown in Fig. 10. The system states and control inputs of this problem
are listed in Table I, and the vehicle parameters are listed in Table II.

**B. Problem Description**

The offline policy is trained based on the nonliner and noninput-affine vehicle dynamics

\[
\begin{align*}
\dot{x} &= \begin{bmatrix} y \\ \phi \\ vy \\ \omega_r \\
\end{bmatrix}, \quad u = \delta, \quad x_{i+1} = \begin{bmatrix} vy \sin \phi + vy \cos \phi \\ \omega_r - \frac{F_{fy} \cos \delta + F_{fr}}{\mu_f} \frac{\alpha_f}{2} \frac{v_x}{\omega_r} - \frac{v_x \omega_r}{\alpha_f} \frac{1}{f} + x_i
\end{align*}
\]

where \( F_{fy} \) and \( F_{fr} \) are the lateral tire forces of the front and rear tires, respectively [32]. The lateral tire forces can be approximated according to the Fiala tire model

\[
F_{y\#} = \left\{ \begin{array}{ll}
-C_{\#} \tan \alpha_{\#} & \text{if } |\alpha_{\#}| \leq |\alpha_{\max, \#}|, \\
\mu_{\#} F_{z\#} & \text{if } |\alpha_{\#}| > |\alpha_{\max, \#}|
\end{array} \right.
\]

where \( \alpha_{\#} \) is the tire slip angle, \( F_{z\#} \) is the tire load, \( \mu_{\#} \) is the friction coefficient, and the subscript \( \# \in \{f, r\} \) represents the front or rear tires. The slip angles can be calculated from the relationship between the front/rear axle and the center of gravity (CG)

\[
\alpha_f = \arctan \left( \frac{v_y + a \omega_r}{v_x} \right) - \delta, \quad \alpha_r = \arctan \left( \frac{v_y - b \omega_r}{v_x} \right).
\]

The loads on the front and rear tires can be approximated by

\[
F_{zf} = \frac{b}{a + b} m g, \quad F_{zr} = \frac{a}{a + b} m g.
\]

The utility function of this problem is set to be

\[
l(x_i, r_i, u_{i-1}) = (1, 0, 0, 0)[x_i - r_i]^2 + 10 u_{i-1}^2 + (0, 0, 0, 1)[x_i]^2.
\]

Therefore, the policy optimization problem of this example can be formulated as

\[
\min_{\theta} \quad \mathbb{E} \quad \{V(x_0, r_1; N_{\max}; N_{\max}; \theta)\}
\]

\[
s.t. \quad x_i = f(x_{i-1}, u_{i-1})
\]

\[
\min u_{i-1} - u_{\max} \leq u_{i-1} \leq u_{\max} \quad i \in [1, N_{\max}].
\]

where \( V(x_0, r_1; N_{\max}; N_{\max}; \theta) = \sum_{i=1}^{N_{\max}} l(x_i, r_i, u_{i-1}), u_{i-1} = \pi_{N_{\max}}^{N_{\max}+1}(x_{i-1}, r_{i-1}; N_{\max}; \theta) \), \( N_{\max} = 15 \), \( u_{\min} = -0.2 \) rad, and \( u_{\max} = 0.2 \) rad.

**C. Algorithm Details**

The policy function is represented by a variant of RNN, called gated recurrent unit (GRU). The input layer is composed of the states, followed by four hidden layers using rectified linear units (RELU) as activation functions, with 128 units per layer. The output layer is set as a tanh layer, multiplied by 0.2 to confront bounded control inputs. We use the Adam optimization method to update the policy with the learning rate of \( 2 \times 10^{-4} \) and the batch size of 256.

**D. Result Analysis**

Given a nonlinear MPC problem, we can directly solve it with some optimization solvers, such as IPOPT [16] and BONMIN [17], whose numerical solutions can be approximately regarded as the optimal policy. In the sequel, both IPOPT and BONMIN are implemented in a symbolic framework, called CasADi [33].

Since RMPC is an explicit solver of the traditional MPC problem, if the control inputs of RMPC are approximately equal to the optimal numerical solutions under different prediction horizons, we can immediately show that RMPC can adaptively choose the longest prediction step. We run Algorithm 1 for 10 runs and calculate the policy error \( e_N \) between the solution of IPOPT and RMPC at each iteration with different prediction(boolean)
Fig. 7. Policy errors during training. Solid lines are average values over 10 runs. Shaded regions correspond to 95% confidence interval.

Fig. 8. RMPC versus other solvers in terms of online computation time.

Fig. 9. Performance comparison between RMPC and IPOPT. Solid lines are average values over 50 initialized states. Shaded regions correspond to 95% confidence interval.

Table III

| Method       | Average Tracking Errors with Different System Parameters |
|--------------|---------------------------------------------------------|
| m [kg]       | errors [cm]                                             |
| 1200         | 1.76                                                   |
| 1300         | 1.65                                                   |
| 1400         | 1.54                                                   |
| 1500         | 1.46                                                   |
| 1600         | 1.39                                                   |
| 1700         | 1.35                                                   |
| 1800         | 1.33                                                   |

For all prediction domains N, RMPC performs as well as the solution of the IPOPT solver. Besides, more recurrent cycles (or long prediction steps) help to reduce the cost-to-go L.

In detail, Fig. 10 intuitively presents the trajectory curves and the corresponding control inputs of RMPC and the IPOPT solver. Obviously, the trajectory and control inputs generated by the RMPC controller almost overlap with the IPOPT controller. And the trajectory tracking error decreases significantly with the number of recurrent cycles. This explains the importance of adaptively selecting the optimal control input with the longest prediction horizon in real applications.

RMPC is an explicit MPC method, whose policy is learned offline based on predetermined system parameters. However, in practical applications, system parameters may vary due to online changes or inaccuracy measurements. To evaluate the robustness of RMPC to different system parameters, we have changed certain system parameters and tested the performance of the fixed policy learned based on parameters in Table II. The average tracking errors under different parameters are shown in Table III. Results show that the final tracking performance is insensitive to the selected parameter changes. This implies that RMPC preserves good robustness property when the changes of system parameters are restricted within a reasonable range. This is mainly because the system usually behaves analogously as

\[
e_N = \frac{1}{N} \sum_{i=1}^{N} \left[ u_{0}^{N}(x_0, r_0, N) - \pi^N(x_0, r_0; \theta) \right] \]

where \( u_{\text{max}}^N \) and \( u_{\text{min}}^N \) are, respectively, the maximum and minimum value of \( u_{0}^{N}(x_0, r_1, N) \) for \( \forall x_0 \in X, \forall N \in [1, 15] \). Fig. 7 plots policy error curves during training with different prediction steps. It is clear that all the policy errors decrease rapidly to a small value during the training process. In particular, after \( 10^4 \) iterations, policy errors for all \( N \geq 5 \) reduce to less than 2%. This indicates that Algorithm 1 has the ability to find the nearly optimal explicit policy of MPC problems with different prediction horizons.

Fig. 8 compares the calculation efficiency of RMPC and optimization solvers in online applications. It is obvious that the calculation time of the optimization solvers is much longer than RMPC, and the gap increases with the number of prediction steps. Specifically, when \( N = 15 \), the IPOPT solver is about five times slower than RMPC (IPOPT for 26.2 ms, RMPC for 4.7 ms). This demonstrates the online effectiveness of the RMPC method.

Fig. 9 compares the policy performance of IPOPT and RMPC with different prediction horizons. The policy performance is measured by the cost-to-go of 200 steps (10 s) during simulation starting from randomly initialized states, i.e.,

\[
L = \sum_{i=1}^{200} l(x_i, r_i, u_{i-1}).
\]
Fig. 10. Simulation results: trajectory and control inputs curves with different recurrent cycles $c$. (a) Trajectories when $c = 7$. (b) Trajectories when $c = 11$. (c) Trajectories when $c = 15$. (d) Control inputs when $c = 7$. (e) Control inputs when $c = 11$. (f) Control inputs when $c = 15$.

Fig. 11. Experiment results: trajectory and control inputs curves with different recurrent cycles $c$. (a) Trajectories when $c = 3$. (b) Trajectories when $c = 9$. (c) Trajectories when $c = 15$. (d) Control inputs when $c = 3$. (e) Control inputs when $c = 9$. (f) Control inputs when $c = 15$.

long as parameters do not change a lot. One can also add some parametric noises during the training process to further improve robustness. For those frequently changing and easily detected parameters, simply setting them as policy inputs can be a good choice.

To summarize, this example demonstrates the optimality, efficiency, and generality of the RMPC algorithm.

V. EXPERIMENTAL VERIFICATION AND FUTURE WORK

A. Experimental Verification

As shown in Fig. 6, an IDRIVERPLUS four-wheeled robot is utilized to demonstrate the effectiveness of the proposed method in practical applications. For ease of understanding, this experiment is carried out by replacing the Carsim simulator in Fig. 6 with a real robot. Except for the vehicle parameters, all training details are the same as those in Section IV. Note that the actual vehicle parameters are usually quite different from the theoretical model used for learning, since some parameters, such as tire cornering stiffness, are difficult to measure accurately. Therefore, the experimental results can also reflect the robustness of RMPC to inaccurate vehicle parameters.

We deployed the learned policy of RMPC in the four-wheeled robot, aiming to follow a sine-shaped reference path. Fig. 11 shows the control results of RMPC and the IPOPT solver with different prediction steps. Although the variance of the steering wheel angle is larger than that in simulation due to the existence of system noise, both methods have achieved relatively good
tracking performance. Table IV compares the average tracking errors of these two methods under different number of prediction steps. Results show that RMPC achieved a smaller average tracking error than IPOPT in all cases. In particular, when \( c = 3 \), \( c = 9 \), and \( c = 15 \), RMPC reduces the tracking error by 56.2\%, 10.1\%, and 4.0\%, respectively. It is also obvious that the trajectory tracking error decreases significantly as the number of recurrent cycles increases, which provides evidence for the advantage of adaptively selecting the maximum prediction horizon. This real-world experiment demonstrates the efficacy of RMPC in practical applications.

B. Limitations and Future Work

In this article, the proposed RMPC method is only suitable for problems without state constraints. In the future, we will extend RMPC to constrained cases by combining constrained policy optimization techniques [27]. Besides, the performance of the proposed RMPC method is evaluated only by the path tracking task of four-wheeled vehicles. More subsequent experiments for different systems and tasks will be addressed in further studies. The future work also includes improving its robustness and investigating the stability of RMPC.

VI. Conclusion

This article proposes the RMPC algorithm to solve general nonlinear finite-horizon optimal control problems. Unlike traditional MPC algorithms, it can make full use of the current computing resources and adaptively select the longest model prediction horizon. Our algorithm employs an RNN to approximate the optimal policy, which maps the system states and reference values directly to the control inputs. The output of the learned policy network after \( N \) recurrent cycles corresponds to the nearly optimal solution of \( N \)-step MPC. A policy optimization objective was designed by decomposing the MPC cost function according to the Bellman’s principle of optimality. The optimal recurrent policy can be obtained by directly minimizing the designed objective function, which was applicable for general nonlinear and noninput-affine systems. The convergence and optimality of RMPC are further proved. We demonstrate its optimality, generality, and efficiency using a HIL experiment. Results showed that RMPC is over five times faster than the traditional MPC solver. The control performance of the learned policy can be further improved as the number of recurrent cycles increases. To prove its practicality, RMPC has also been applied to a real-world robot path-tracking task, and RMPC has achieved better path tracking accuracy than IPOPT in different prediction horizons.

TABLE IV

| Prediction step | RMPC | IPOPT |
|-----------------|------|-------|
| \( c = 3 \)     | 10.43cm | 23.79cm |
| \( c = 9 \)     | 9.74cm | 10.83cm |
| \( c = 15 \)    | 6.75cm | 7.03cm |

ACKNOWLEDGMENT

The authors would like to thank the Editor-in-Chief, the Associate Editor, and anonymous reviewers for their valuable comments.

REFERENCES

[1] S. J. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” Control Eng. Pract., vol. 11, no. 7, pp. 733–764, 2003.
[2] S. Vazquez et al., “Model predictive control: A review of its applications in power electronics,” IEEE Ind. Electron. Mag., vol. 8, no. 1, pp. 16–31, Mar. 2014.
[3] S. E. Li, Z. Jia, K. Li, and B. Cheng, “Fast online computation of a model predictive control algorithm and its application to fuel economy-oriented adaptive cruise control,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 3, pp. 1199–1209, Jun. 2015.
[4] J. H. Lee, “Model predictive control: Review of the three decades of development,” Int. J. Control, Automat. Syst., vol. 9, no. 3, pp. 415–424, 2011.
[5] R. Cagienard, P. Grieder, E. C. Kerrigan, and M. Morari, “Move blocking strategies in receding horizon control,” J. Process Control, vol. 17, no. 6, pp. 563–570, 2007.
[6] Y. Wang and S. Boyd, “Fast model predictive control using online optimization,” IEEE Trans. Control Syst. Technol., vol. 18, no. 2, pp. 267–278, Mar. 2010.
[7] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, “The explicit linear quadratic regulator for constrained systems,” Automatica, vol. 38, no. 1, pp. 3–20, 2002.
[8] B. Kouvaritakis, M. Cannon, and J. A. Rossiter, “Who needs QP for linear MPC anyway?,” Automatica, vol. 38, no. 5, pp. 879–884, 2002.
[9] T. Geyer, F. D. Torrisi, and M. Morari, “Optimal complexity reduction of polyhedral piecewise affine systems,” Automatica, vol. 44, no. 7, pp. 1728–1740, 2008.
[10] C. N. Jones and M. Morari, “Polytopic approximation of explicit model predictive controllers,” IEEE Trans. Autom. Control, vol. 55, no. 11, pp. 2542–2553, Nov. 2010.
[11] C. Wen, X. Ma, and B. E. Ydstie, “Analytical expression of explicit MPC solution via lattice piecewise-affine function,” Automatica, vol. 45, no. 4, pp. 910–917, 2009.
[12] F. Borrelli, M. Batoić, J. Pekar, and G. Stewart, “On the computation of linear model predictive control laws,” Automatica, vol. 46, no. 6, pp. 1035–1041, 2010.
[13] B. M. Åkesson, H. T. Toivonen, J. B. Waller, and R. H. Nyström, “Neural network approximation of a nonlinear model predictive controller applied to a pH neutralization process,” Comput. Chem. Eng., vol. 29, no. 2, pp. 323–335, 2005.
[14] B. M. Åkesson and H. T. Toivonen, “A neural network model predictive controller,” J. Process Control, vol. 16, no. 9, pp. 937–946, 2006.
[15] L. Cheng, W. Liu, Z.-G. Hou, J. Yu, and M. Tan, “Neural-network-based nonlinear model predictive control for piezoelectric actuators,” IEEE Trans. Ind. Electron., vol. 62, no. 12, pp. 7717–7727, Dec. 2015.
[16] A. Wächter and L. T. Biegler, “On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming mathematical programming,” Math. Program., vol. 106, no. 1, pp. 25–57, 2006.
[17] P. Bonami et al., “An algorithmic framework for convex mixed integer nonlinear programs,” Discrete Optim., vol. 5, no. 2, pp. 186–204, 2008.
[18] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur, “Recurrent neural network based language model,” in Proc. 11th Annu. Conf. Int. Speech Commun. Assoc., 2010, pp. 1045–1048.
[19] S. Li, H. Wang, and M. U. Rafique, “A novel recurrent neural network for manipulator control with improved noise tolerance,” IEEE Trans. Neural Netw. Learn. Syst., vol. 29, no. 5, pp. 1908–1918, May 2018.
[20] L. K. Li, “Approximation theory and recurrent networks,” in Proc. IEEE Int. Joint Conf. Neural Netw., vol. 2, 1992, pp. 266–271.
[21] A. M. Schäfer and H.-G. Zimmermann, “Recurrent neural networks are universal approximators,” Int. J. Neural Syst., vol. 17, no. 04, pp. 253–263, 2007.
[22] B. Hammer, “On the approximation capability of recurrent neural networks,” Neurocomputing, vol. 31, no. 1–4, pp. 107–123, 2000.
dynamic programming, and driver behavior analysis.

[23] Z. Allen-Zhu, Y. Li, and Z. Song, “A convergence theory for deep learning via over-parameterization,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 242–252.

[24] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, “Gradient descent finds global minima of deep neural networks,” in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1675–1685.

[25] Z. Allen-Zhu, Y. Li, and Z. Song, “On the convergence rate of training recurrent neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 6673–6685.

[26] R. Li, Y. Li, S. E. Li, E. Burdet, and B. Cheng, “Driver-automation indirect shared control of highly automated vehicles with intention-aware authority transition,” in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 26–32.

[27] J. Duan, Z. Liu, S. E. Li, Q. Sun, Z. Jia, and B. Cheng, “Adaptive dynamic programming for nonaffine nonlinear optimal control problem with state constraints,” *Neurocomputing*, vol. 484, pp. 128–141, 2021, doi: 10.1016/j.neucom.2021.04.134.

[28] S. Cheng *et al.*, “Model-predictive-control-based path tracking controller of autonomous vehicle considering parametric uncertainties and velocity-varying,” *IEEE Trans. Ind. Electron.*, vol. 68, no. 9, pp. 8698–8707, Sep. 2021.

[29] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, “Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints,” *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[30] R. F. Benekohal and J. Treiterer, “Carsim: Car-following model for simulation of traffic in normal and stop-and-go conditions,” *Transp. Res. Rec.*, vol. 1194, pp. 99–111, 1988.

[31] C. Chen, J. Pan, X. Chang, Q. Xu, and J. Wang, “System design and function verification of an extensible autonomous driving platform,” in *Proc. IEEE Int. Symp. Circuits Syst.*, 2019, pp. 1–5.

[32] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, “Kinematic and dynamic vehicle models for autonomous driving control design,” in *Proc. IEEE Intell. Veh. Symp.*, 2015, pp. 1094–1099.

[33] J. A. E. Andersson, G. Joris, H. Greg, J. B. Rawlings, and D. Moritz, “Casadi: A software framework for nonlinear optimization and optimal control,” *Math. Program. Comput.*, vol. 11, pp. 1–36, 2018.

Zhengyu Liu received the B.S. degree in automotive engineering from the College of Automotive Engineering, Jilin University, Changchun, China, in 2017, and the master’s degree in mechanical engineering from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2020. His current research interests include decision-making and control of autonomous vehicles, adaptive dynamic programming and reinforcement learning algorithms.

Jingliang Duan received the Ph.D. degree in mechanical engineering from the School of Vehicle and Mobility, Tsinghua University, Beijing, China, in 2021. He studied as a Visiting Student Researcher with the Department of Mechanical Engineering, University of California, Berkeley, CA, USA, in 2019. He is currently a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include decision and control of autonomous vehicle, reinforcement learning and adaptive dynamic programming, and driver behavior analysis.

Wenxuan Wang received the B.S. degree in vehicle engineering from the Beijing Institute Of Technology, Beijing, China, in 2018. He is currently a Member with the State Key Laboratory of Automotive Safety and Energy. School of Vehicle and Mobility, Tsinghua University. His current research interests include decision-making and control of automated vehicles, and reinforcement learning algorithms.

Shengbo Eben Li (Senior Member, IEEE) received the M.S. and Ph.D. degrees in mechanical engineering from Tsinghua University, Beijing, China, in 2008 and 2009, respectively. He worked with Stanford University, University of Michigan, and University of California, Berkeley. He is currently a Tenured Associate Professor with Tsinghua University. He is the Author of more than 100 journal/conference papers, and the Co-Inventor of over 20 Chinese patents. His active research interests include intelligent vehicles and driver assistance, reinforcement learning and distributed control, optimal control and estimation, etc. Prof. Li serves as an Associated Editor for the IEEE ITS and IEEE Trans. ITS, etc.

Yuming Yin received the M.S. degree in vehicle engineering from the University of Science and Technology Beijing, Beijing, China, in 2013, and the Ph.D. degree in mechanical engineering from Concordia University, Montreal, QC, Canada, in 2017, respectively. He is currently Associate Professor with the School of Mechanical Engineering, Zhejiang University of Technology, and the Visiting Scholar with the School of Vehicle and Mobility, Tsinghua University. His active research interests include ground vehicle system dynamics, RL and motion control.

Ziyu Lin received the B.S. degree in automotive engineering from China Agricultural University, Beijing, China, in 2017. She is currently working toward the Ph.D. degree in mechanical engineering with the School of Vehicle and Mobility, Tsinghua University, Beijing. Her current research interests include model-based reinforcement learning, approximate dynamic programming, and model predictive control and distributed control.

Bo Cheng received the B.S. and M.S. degrees in automotive engineering from Tsinghua University, Beijing, China, in 1985 and 1988, respectively, and the Ph.D. degree in mechanical engineering from the University of Tokyo, Tokyo, Japan, in 1998. He is currently a Professor with the School of Vehicle and Mobility, Tsinghua University, and the Dean of Tsinghua University Suzhou Autonomous Research Institute. His active research interests include autonomous vehicles, driver-assistance systems, active safety, and vehicular ergonomics.