# Relaxed Actor-Critic with Convergence Guarantees for Continuous-Time Optimal Control of Nonlinear Systems

Jingliang Duan, Jie Li, Qiang Ge, Shengbo Eben Li, Monimoy Bujarbaruah, Fei Ma, Dezhao Zhang

***Abstract*—This paper presents the Relaxed Continuous-Time Actor-critic (RCTAC) algorithm, a method for finding the nearly optimal policy for nonlinear continuous-time (CT) systems with known dynamics and infinite horizon, such as the path-tracking control of vehicles. RCTAC has several advantages over existing adaptive dynamic programming algorithms for CT systems. It does not require the "admissibility" of the initialized policy or the input-affine nature of controlled systems for convergence. Instead, given any initial policy, RCTAC can converge to an admissible, and subsequently nearly optimal policy for a general nonlinear system with a saturated controller. RCTAC consists of two phases: a warm-up phase and a generalized policy iteration phase. The warm-up phase minimizes the square of the Hamiltonian to achieve admissibility, while the generalized policy iteration phase relaxes the update termination conditions for faster convergence. The convergence and optimality of the algorithm are proven through Lyapunov analysis, and its effectiveness is demonstrated through simulations and real-world path-tracking tasks.**

***Index Terms*—reinforcement learning, continuous-time optimal control, nonlinear systems**

## I. Introduction

Dynamic Programming (DP) offers a systematic way to solve Continuous Time (CT) infinite horizon optimal control problems with known dynamics for unconstrained linear systems. It does this by using the principle of Bellman optimality and the solution of the underlying Hamilton-Jacobi-Bellman (HJB) equation [1], yielding the well-known Linear Quadratic Regulator (LQR) [2]. In this case, the optimal control policy is an affine state feedback. However, solving an infinite horizon optimal control problem analytically becomes more difficult when the system is subject to operating constraints or has nonlinear dynamics. This is because it becomes difficult to obtain an analytical solution of the HJB equation, which is typically a nonlinear partial differential equation [3]. This is known as the *curse of dimensionality*, as the computation burden grows exponentially with the dimensionality of the system [4]. Traditional DP methods are therefore not applicable in these cases.

To find a nearly optimal approximation of the optimal control policy for nonlinear dynamics, Werbos proposed the Adaptive DP (ADP) method [5], also known as Reinforcement Learning (RL) in the field of machine learning [6]–[10]. A distinct characteristic of ADP is that it utilizes a critic parameterized function, such as a Neural Network (NN), to approximate the value function, and an actor parameterized function to approximate the policy. The classical Value Iteration (VI) framework, which approximates the value function through one-step back-up operation, is commonly used for building ADP methods for discrete-time systems [11]. However, for CT systems, the value update law of VI requires integrating the running cost over a finite time horizon, which can lead to learning inaccuracies due to discretization error [12]. Most ADP methods adopt an iterative technique called Policy Iteration (PI) to find suitable approximations of both the value function and policy, which directly updates the value function by solving the corresponding Lyapunov equation without the need for discretization [7]. PI consists of two-step process: 1) policy evaluation, in which the value function moves towards its true value associated with an admissible control policy, and 2) policy improvement, in which the policy is updated to reduce the corresponding value function.

In recent years, there has been a significant amount of research on the use of ADP (or RL) techniques for the control of autonomous systems [13]–[17]. One example of CT optimal control is the ADP algorithm proposed by Abu-Khalaf and Lewis, which seeks to find a nearly optimal constrained state-feedback controller for nonlinear systems by using a non-quadratic cost function [18]. The value function, parameterized by a linear function of hand-crafted features, is trained by the least square method at the policy evaluation step, and the policy is expressed as an analytic function of the value function. Utilizing the same single approximator scheme, Dierks and Jagannathan developed a novel online parameter tuning law that ensures the optimality of both the value function and control policy, as well as maintaining bounded system states during the learning process [19]. [20] proposed a synchronous PI algorithm with an actor-critic framework for nonlinear CT systems without input constraints. Both the value function and policy are approximated by linear methods and tuned

Jingliang Duan and Jie Li contributed equally to this work. All correspondences should be sent to S. Li with email: lisb04@gmail.com.

J. Duan is with the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, 100083, China, and also with the School of Vehicle and Mobility, Tsinghua University, Beijing, 100084, China. Email:duanjl15@163.com.

J. Li, Q. Ge and S. E. Li are with the School of Vehicle and Mobility, Tsinghua University, Beijing, 100084, China. Email: jie-li18@mails.tsinghua.edu.cn, gq17@mails.tsinghua.edu.cn, lisb04@gmail.com.

M. Bujarbaruah is with the Department of Mechanical Engineering, University of California Berkeley, Berkeley, CA 94720, USA. Email: monimoyb@berkeley.edu.

F. Ma is with the School of Mechanical Engineering, University of Science and Technology Beijing, Beijing, 100083, China. Email: yeke@ustb.edu.cn.

D. Zhang is with Beijing Idriverplus Technology Co., Ltd., Beijing, 100192, China. Email: zhangdezhao@idriverplus.com.

simultaneously online. Furthermore, Vamvoudakis introduced an event-triggered ADP algorithm that reduces computation cost by only updating the policy when a specific condition is violated [21], and Dong *et al.* expanded upon this concept for use in nonlinear systems with saturated actuators [22]. In addition, ADP methods have also been widely applied in the optimal control of incompletely known dynamic systems [23]–[27] and multi-agent systems [28]–[30].

It is worth noting that most existing ADP techniques for CT systems are valid on the basis of one or both of the following two requirements:

- *A1: Admissibility of Initial Policy*: One is the requirement for an admissible initial policy, meaning that the policy must be able to stabilize the system. This is because the infinite horizon value function can only be evaluated for stabilizing control policies. However, it can be challenging to obtain an admissible policy, particularly for complex systems.
- *A2: Input-Affine Property of System*: Most ADP methods are limited to input-affine systems because these methods require that the optimal policy can be represented by the value function. This means that the minimum point of the Hamilton function can be solved analytically when the value function is given. However, this is not possible for input non-affine systems, making it difficult to directly solve the optimal policy.

Note that while there are certain methods, such as integral VI [12] and parallel-control-based ADP methods [31], which do not require initial admissible control policies, they are only suitable for linear or input-affine systems. For example, the differentiation term of the time derivative of the Lyapunov function with respect to the control policy can be added to its updating rule to make the initial admissible policies no longer necessary [19], [23], [32], which requires the input-affine property. [33] goes one step further and eliminates the restriction of the affine property by transforming the general nonlinear system into an affine system. However, this approach generally leads to some loss in policy performance. Additionally, the studies mentioned above approximate the value function or the policy using a single NN (i.e., the linear combination of predetermined hand-crafted basis functions). This means that the performance of these methods is heavily dependent on the quality of hand-crafted features, limiting their generality since it is challenging to design such features for high-dimensional nonlinear systems. Therefore, this paper aims to address these two requirements without sacrificing optimality guarantees and generality.

In this paper, we propose a relaxed continuous-time actor-critic (RCTAC) algorithm with the guarantee of convergence and optimality for solving optimal control problems of general nonlinear CT systems with known dynamics, which overcomes the limitation of the above two central requirements. Our main contributions are summarized as follows:

1) The warm-up phase of RCTAC is developed to relax the requirement of A1. It is proved that given any initial policy, when the activation function of the value network meets certain requirements, the warm-up phase can converge to an admissible policy by continuously minimizing the square of the Hamiltonian. Unlike the policy tuning rule used in [19], [23], [32], which also relaxes A1 but is restricted to input-affine systems and single-NN-based policies, the developed warm-up phase is applicable to general non-affine systems. Moreover, RCTAC obviates the need for designing hand-crafted basis functions by utilizing multi-layer neural networks to approximate both the actor and critic, which create mappings from the system states to control inputs and the value function, respectively.
2) Different from [12], [19], [23] which require input-affine systems because the policy must be represented analytically by the value function, the policy network in the RCTAC algorithm is updated by directly minimizing the associated Hamiltonian using gradient descent methods in the generalized PI phase. This allows the RCTAC algorithm to be applied to arbitrary nonlinear dynamics with bounded and non-affine inputs, thereby relaxing the requirement of A2. Compared to the method of transforming general nonlinear systems into affine systems by creating augmented systems [33], our method does not sacrifice the guarantee of theoretical optimality.
3) We introduce novel update termination conditions for the policy evaluation and improvement processes, resulting in a faster convergence speed than the corresponding PI methods. We also provide a Lyapunov analysis to prove the convergence and optimality of RCTAC.

Throughout the paper, we provide two detailed numerical experiments to show the generality and efficacy of the proposed RCTAC algorithm, including a linear optimal control problem and a path-tracking control task for vehicles with nonlinear and non-input-affine dynamics. Besides, we demonstrate the practical application performance of our algorithm through an actual longitudinal and lateral vehicle control task.

**Organization:** Section II provides the formulation of the optimal control problem and the description of PI. In Section III, we describe the RCTAC algorithm and analyze its convergence and optimality. In Section IV, we present simulation examples that show the generality and effectiveness of the RCTAC algorithm for CT system. In Section V, we conduct experiments to verify the effectiveness of the method in practical applications. Section VI concludes this paper.

## II. Mathematical Preliminaries

### A. HJB Equation

This study considers the time-invariant system

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

where $x \in \mathbb{R}^n$ denotes the state, $u \in \mathbb{R}^m$ denotes the control input, and $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ denotes the system dynamics. The dynamics $f(x, u)$ is assumed to be Lipschitz continuous on a compact set $\Omega$ that contains the origin. We suppose a continuous policy $u = \pi(x)$ on $\Omega$ that asymptotically stabilizes the system exists. We assume the full information of $f(x(t), u(t))$ is available. It can be represented by a nonlinear and input non-affine function or a Neural Network (NN) only if $\frac{\partial f(x,u)}{\partial u}$ is accessible. The control signal $u(t)$ can be saturated or unsaturated. The value function (cost-to-go) of policy $\pi(x)$ is calculated as

$$V^\pi(x) = \int_t^\infty l(x(\tau), \pi(x(\tau)))\mathrm{d}\tau\Big|_{x(t)=x}, \tag{2}$$

where $l : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is positive definite, i.e., if and only if $(x, u) = (0, 0)$, $l(x, u) = 0$; otherwise, $l(x, u) > 0$. For dynamics in (1) and the corresponding value function in (2), we give the associated Hamilton function

$$H(x, u, \frac{\partial V^\pi(x)}{\partial x}) := l(x, u) + \frac{\partial V^\pi(x)}{\partial x^\top} f(x, u). \tag{3}$$

**Definition 1.** *(Admissible Policy [34]). A controller $\pi(x)$ is called the admissible policy, denoted by $\pi(x) \in \Psi(\Omega)$, if it is continuous on $\Omega$ with $\pi(0) = 0$, and stabilizes* (1) *on $\Omega$.*

Given an admissible policy $\pi(x) \in \Psi(\Omega)$, the differential equivalent to (2) on $\Omega$ is called the Lyapunov equation

$$H(x, \pi(x), \frac{\partial V^\pi(x)}{\partial x}) = 0, \tag{4}$$

where $V^\pi(0) = 0$. Therefore, given a policy $\pi(x) \in \Psi(\Omega)$, we can obtain the value function (2) of system (1) by solving the Lyapunov equation. Then the optimal control problem for Continuous Time (CT) system can be formulated as solving a policy $\pi(x) \in \Psi(\Omega)$ such that the corresponding value function is minimum. The minimized or optimal value function $V^\star(x(t))$ defined by

$$V^\star(x(t)) = \min_{\pi(x)\in\Psi(\Omega)} \int_t^\infty l(x(\tau), \pi(x(\tau))\mathrm{d}\tau, \tag{5}$$

satisfies the classical Hamilton-Jacobi-Bellman (HJB) equation

$$\min_u H(x, u, \frac{\partial V^\star(x)}{\partial x}) = 0, \quad V^\star(0) = 0. \tag{6}$$

Meanwhile, the optimal control policy $\pi^\star(x)$ can be calculated as

$$\pi^\star(x) = \arg\min_u H(x, u, \frac{\partial V^\star(x)}{\partial x}), \forall x \in \Omega. \tag{7}$$

which is the globally optimal solution to (5). Inserting $V^\star(x)$ and $\pi^\star(x)$ in (4), one can reformulate (6) as

$$l(x, \pi^\star(x)) + \frac{\partial V^\star(x)}{\partial x^\top} f(x, \pi^\star(x)) = 0$$

with $V^\star(0) = 0$. The optimal value function is shown to exist and be unique in [35]. To obtain the optimal policy, we first need to solve the HJB equation (6) to find the value function, and then use it to calculate the optimal policy using (7). However, due to the nonlinear property of the HJB equation, it is often challenging or even impossible to find a solution.

## B. Policy Iteration

Rather than attempting to solve the HJB equation directly, most Adaptive Dynamic Programming (ADP) methods adopt an iterative technique, called Policy Iteration (PI), to approximate both the value function and the policy [11]. PI consists of alternating between policy evaluation using (4) and policy improvement using (7). The algorithm proposed in this paper is also based on PI, as shown in the pseudocode in Algorithm 1.

**Algorithm 1** PI for CT optimal control

```
Initial with policy $\pi^0 \in \Psi(\Omega)$
Given any small positive number $\epsilon$ and let $i = 0$
while $\max_{x\in\Omega} |V^i(x) - V^{i+1}(x)| \geq \epsilon$ do
    1. Find value function $V^i(x)$ for all $x \in \Omega$ by
       $l(x, \pi^i(x)) + \frac{\partial V^i(x)}{\partial x^\top} f(x, \pi^i(x)) = 0, \quad V^i(0) = 0$   (8)
    2. Find new policy $\pi^{i+1}(x)$ by
       $\pi^{i+1}(x) = \arg\min_u [l(x, u) + \frac{\partial V^i(x)}{\partial x^\top} f(x, u)]$   (9)
    $i = i + 1$
end while
```

As shown in Algorithm 1, the first step of PI is to find an initial policy $\pi^0(x) \in \Psi(\Omega)$, because the associated value function $V^0(x)$ is finite only when the system is asymptotically stable. Algorithm 1 then iteratively refines both the optimal control policy and the optimal value function. The convergence and optimality of the algorithm are proven in [18].

## C. Value Function and Policy Approximation

In previous ADP research for CT systems, the value function $V^i(x)$ and policy $\pi^i(x)$ are usually approximated by linear methods, which requires a large number of artificially designed basis functions [24]. In recent years, NNs are favored in many fields, such as deep learning and Reinforcement Learning (RL), due to their better generality and higher fitting ability [36]. In our work, we represent the value function and policy using multi-layer neural networks (NNs), referred to as the value NN $V(x; \omega)$ ($V_\omega(x)$ for short) and the policy NN $\pi(x; \theta)$ ($\pi_\theta(x)$ for short), where $w$ and $\theta$ denote network weights. The two NNs in this case are used to map the original system states to the estimated value function and control inputs, respectively.

By inserting the value and policy network in (3), we can obtain an approximated Hamiltonian expressed in terms of the parameters $w$ and $\theta$

$$H(x, \omega, \theta) = l(x, \pi_\theta(x)) + \frac{\partial V_\omega(x)}{\partial x^\top} f(x, \pi_\theta(x)).$$

We refer to the algorithm combining PI and multi-layer NN as approximate PI (API), which involves processes alternatively tuning each of the two networks to find nearly optimal parameters $\omega^\star$ and $\theta^\star$ satisfying $V_{\omega^\star}(x) \approx V^\star(x)$, $\pi_{\theta^\star}(x) \approx \pi^\star(x)$.

Given any policy $\pi_\theta \in \Psi(\Omega)$, the value network is tuned by solving the Lyapunov equation (8) during the policy evaluation phase of API, which is equivalent to finding parameters $w$ to minimize the following critic loss function

$$L_c(\omega, \theta) = \mathbb{E}_{x\sim d_x}\big[H(x, \omega, \theta)^2\big], \tag{10}$$

where $d_x$ represents the state distribution over $\Omega$. It is important to note that $d_x$ can be any distribution that satisfies the requirement of a positive probability density for all $x \in \Omega$, such as the uniform distribution. The condition $V(x;\omega) \equiv 0$ can be easily guaranteed by selecting proper activation function $\sigma_V(\cdot)$ for the value network. Based on (9), the policy improvement process is carried out by tuning the policy network to minimize the expected Hamiltonian in each state, which is also called actor loss function here

$$L_a(\omega, \theta) = \mathbb{E}_{x \sim d_x}\big[H(x, \omega, \theta)\big]. \tag{11}$$

The benefit of updating the policy network by minimizing $L_a(\omega, \theta)$ is that the tuning rule is applicable to any nonlinear dynamics with non-affine and constrained inputs. This relaxes the requirement of A2 (from Introduction).

Since the state $x$ is continuous, it is difficult to directly compute the expectation in (10) and (11). In practice, these two loss functions can be estimated by the sample average. We can directly utilize existing NN optimization methods to adjust the parameters of value and policy NNs, such as Stochastic Gradient Descent (SGD). The value network and policy network usually require multiple updating iterations to make (8) and (9) hold respectively. Therefore, compared with PI algorithm, two inner updating loops would be introduced to update the value and policy NNs until convergence. Taking SGD as an example, we give the pseudocode of API in Algorithm 2.

**Algorithm 2** API for CT optimal control

```
Initial with $\theta^0$ such that $\pi_{\theta^0}(x) \in \Psi(\Omega)$ and arbitrary $\omega^0$
Choose the appropriate learning rates $\alpha_\omega$ and $\alpha_\theta$
Given any small positive number $\epsilon$ and set $i = 0$
while $\max_{x\in\Omega} |V_{\omega^i}(x) - V_{\omega^{i-1}}(x)| \geq \epsilon$ do
    1. Estimate $V_{\omega^{i+1}}(x)$ using $\pi_{\theta^i}(x)$
        $\omega^{i+1} = \omega^i$
        repeat
            $\omega^{i+1} = \omega^{i+1} - \alpha_\omega \frac{\mathrm{d}L_c(\omega^{i+1}, \theta^i)}{\mathrm{d}\omega^{i+1}}$    (12)
        until $L_c(\omega^{i+1}, \theta^i) \leq \epsilon$
    2. Find improved policy $\pi_{\theta^{i+1}}(x)$ using $V_{\omega^{i+1}}(x)$
        $\theta^{i+1} = \theta^i$
        repeat
            $L_{a,\mathrm{old}} = L_a(\omega^{i+1}, \theta^{i+1})$
            $\theta^{i+1} = \theta^{i+1} - \alpha_\theta \frac{\mathrm{d}L_a(\omega^{i+1}, \theta^{i+1})}{\mathrm{d}\theta^{i+1}}$    (13)
        until $|L_a(\omega^{i+1}, \theta^{i+1}) - L_{a,\mathrm{old}}| \leq \epsilon$
    $i = i + 1$
end while
```

## III. Relaxed Continuous-Time Actor-Critic

Algorithm 2 alternately update the value and policy network by minimizing (10) and (11), respectively. Note that while one NN is being adjusted, the other remains constant. Besides, each NN generally needs multiple iterations to satisfy the terminal condition, which is referred to as the protracted iterative computation problem [11]. This problem usually leads to the admissibility requirement because the initial policy network needs to satisfy that $\pi(x;\theta^0) \in \Psi(\Omega)$ to have a finite and converged value function $V(x;\omega^1)$. Many previous studies used trials and errors process to obtain feasible initial weights for the policy network to guarantee the stability of the system [1], [21]. However, this method usually takes a lot of time, especially for complex systems. On the other hand, the protracted problem often results in slower learning [11].

### *A. Description of the RCTAC Algorithm*

Inspired by the idea of generalized PI framework, which is widely utilized in discrete-time dynamic RL problems [11], we present the relaxed continuous-time actor-critic (RCTAC) algorithm for CT systems to relax the requirement A1 (from Introduction) and improve the learning speed by truncating the inner loops of Algorithm 2 without losing the convergence guarantees. The pseudocode of RCTAC algorithm is shown in Algorithm 3.

**Algorithm 3** RCTAC algorithm

```
Initialize arbitrary $\theta^0$ and $\omega^0$
Choose the appropriate learning rates $\alpha$, $\alpha_\omega$ and $\alpha_\theta$
Given any small positive number $\epsilon$ and set $i = 0$
Phase 1: Warm-up
    while $\max_{x\in\Omega} H(x, \omega^i, \theta^i) \geq 0$ do
        Update $\omega$ and $\theta$ using:
            $\{\omega^{i+1}, \theta^{i+1}\} = \{\omega^i, \theta^i\} - \alpha \frac{\mathrm{d}L_c(\omega^i, \theta^i)}{\mathrm{d}\{\omega^i, \theta^i\}}$    (14)
        $i = i + 1$
    end while
Phase 2: PI with relaxed termination conditions
    while $\max_{x\in\Omega} |V_{\omega^i}(x) - V_{\omega^{i-1}}(x)| \geq \epsilon$ do
        1. Estimate $V_{\omega^{i+1}}(x)$ using $\pi_{\theta^i}(x)$
            $\omega^{i+1} = \omega^i$
            repeat
                Update $\omega^{i+1}$ using (12)
            until $H(x, \omega^i, \theta^i) \leq H(x, \omega^{i+1}, \theta^i) \leq 0, \forall x \in \Omega$
        2. Find improved policy $\pi_{\theta^{i+1}}(x)$ using $V_{\omega^{i+1}}(x)$
            $\theta^{i+1} = \theta^i$
            repeat
                Update $\theta^{i+1}$ using (13)
            until $\max_{x\in\Omega} H(x, \omega^{i+1}, \theta^{i+1}) \leq 0$
        $i = i + 1$
    end while
```

### *B. Convergence and Optimality Analysis*

The solution to (8) may be non-smooth for general nonlinear and input non-affine systems. In keeping with other research in the literature [20], we make the following assumptions.

**Assumption 1.** *If $\pi(x) \in \Psi(\Omega)$, its corresponding value function is smooth, i.e. $V^\pi(x) \in C^1(\Omega)$ (cf. [18], [20]).*

Multiple theoretical analyses and experimental studies have demonstrated that optimization algorithms like SGD are capable of locating the global minimum of the training objective for multi-layer NNs in polynomial time, provided that the NN is over-parameterized (with a sufficiently large number of hidden neurons) [37], [38]. Based on this fact, our second assumption is:

**Assumption 2.** *Optimization algorithms like SGD can locate the global minimum of the critic loss function* (10) *and the actor loss function* (11) *for over-parameterized NNs in polynomial time.*

In the following section, we will demonstrate that the nearly optimal value function and policy can be obtained using Algorithm 3. To do so, we will first introduce some necessary lemmas.

**Lemma 1.** *(Universal Approximation [39]). For any continuous function $F(x)$ defined on a compact set $\Omega$, it is possible to construct a feedforward NN with one hidden layer that uniformly approximates $F(x)$ with arbitrarily small error $\epsilon \in \mathbb{R}^+$.*

Lemma 1 allows us to ignore the NN approximation errors when proving the convergence of Algorithm 3.

**Lemma 2.** *Consider the CT dynamic optimal control problem for* (1) *and* (2)*. Suppose the solution ($V^\pi(x) \in C^1 : \mathbb{R}^n \to \mathbb{R}$ ) of the HJB equation* (6) *is smooth and positive definite. The control policy $\pi(x)$ is given by* (7)*. Then we have that $V^\pi(x) = V^\star(x)$ and $\pi(x) = \pi^\star(x)$ (cf. [3]).*

The next lemma shows how Algorithm 3 can be leveraged to obtain an admissible policy $\pi(x;\theta) \in \Psi(\Omega)$ given any initial policy $\pi(x;\theta^0)$.

**Lemma 3.** *Consider the CT dynamic optimal control problem for* (1) *and* (2)*. The value function (cost-to-go) $V_\omega$ and policy $\pi_\theta$ are represented by over-parameterized NNs. The parameters $w$ and $\theta$ are initialized randomly, i.e., the initial policy $\pi(x;\theta^0)$ can be inadmissible. These two NNs are updated with Algorithm 3. Let Assumption 1 and 2 hold, and suppose all the hyper-parameters (such as $\alpha$, $\alpha_w$ and $\alpha_\theta$) and NN optimization method are properly selected. The NN approximation errors are ignored according to Lemma 1. Suppose all the activation functions $\sigma_V(\cdot)$ and biases $b_V$ of the value network $V(x;\omega)$ are set to $\sigma_V(0) = 0$ and $b_V(\cdot) \equiv 0$. At the same time, the output layer activation function $\sigma_{V_{\text{out}}}$ also needs to satisfy $\sigma_{V_{\text{out}}}(\cdot) \geq 0$. We have that: $\exists N_a \in \mathbb{Z}^+$, if the iteration index $i \geq N_a$, then $\pi(x;\theta^i) \in \Psi(x)$ for the system* (1) *on $\Omega$.*

*Proof.* According to (4) and Lemma 1, there $\exists(\omega^\dagger, \theta^\dagger)$, such that $\pi(x;\theta^\dagger) \in \Psi(\Omega)$ and $H(x,\omega^\dagger,\theta^\dagger) = 0$ for all $x \in \Omega$, which implies that

$$\min_{\omega,\theta} L_c(\omega,\theta) = \min_{\omega,\theta} \mathbb{E}_{x\sim d_x}\Big[H(x,\omega,\theta)^2\Big] = 0.$$

Since Algorithm 3 updates $\omega$ and $\theta$ using (14) to continuously minimize $L_c(\omega,\theta)$ in Phase 1, according to Assumption 2, the global minima of $L_c(\omega,\theta)$ can be obtained in polynomial time. Before the global minima is found, there exists $N_a \in \mathbb{Z}^+$, such that

$$H(x,\omega^{N_a},\theta^{N_a}) \leq 0, \quad \forall x \in \Omega. \tag{15}$$

Taking the time derivative of $V(x;\omega)$, we can derive that

$$\begin{aligned}\frac{\mathrm{d}V(x;\omega)}{\mathrm{d}t} &= \frac{\partial V(x;\omega)}{\partial x^\top} f(x,\pi(x;\theta)),\\ &= H(x,\omega,\theta) - l(x,\pi(x;\theta)).\end{aligned} \tag{16}$$

Using (15) and (16), one has

$$\frac{\mathrm{d}V(x;\omega^{N_a})}{\mathrm{d}t} \leq -l(x,\pi(x;\theta^{N_a})), \quad \forall x \in \Omega.$$

As the running cost $l(x,\pi(x;\theta))$ is positive definite, it follows

$$\frac{\mathrm{d}V_{\omega^{N_a}}(x)}{\mathrm{d}t} < 0, \quad \forall x \in \Omega\backslash\{0\}. \tag{17}$$

Since $\sigma_V(0) = 0$, $b_V(\cdot) \equiv 0$ and $\sigma_{V_{\text{out}}}(\cdot) \geq 0$, we have

$$\begin{cases} V(x;\omega) = 0 \text{ if } x = 0 \ \&\ \forall\omega,\\ V(x;\omega) \geq 0 \text{ if } \forall x \in \Omega\backslash\{0\} \ \&\ \forall\omega. \end{cases} \tag{18}$$

From (17) and (18), we have

$$V(x;\omega^{N_a}) > \min_{z\in\Omega} V(z;\omega^{N_a}) = 0, \quad \forall x \in \Omega\backslash\{0\}. \tag{19}$$

From (18) and (19), we infer that the value function $V(x;\omega^{N_a})$ is positive definite. Then, according to (17), $V(x;\omega^{N_a})$ is a Lyapunov function for the closed-loop dynamics obtained from (1) when policy $\pi(x;\theta^{N_a})$ is used. Therefore, the policy $\pi(x;\theta^{N_a}) \in \Psi(\Omega)$ for the system (1) on $\Omega$ [40], that is, it is a stabilizing admissible policy.

At this point, Algorithm 3 enters Phase 2. According to (4), one has

$$\min_{\omega} L_c(\omega,\theta^{N_a}) = \min_{\omega} \mathbb{E}_{x\sim d_x}\Big[H(x,\omega,\theta^{N_a})^2\Big] = 0.$$

So, from Assumption 2 and Lemma 1, we can always find $\omega^{N_a+1}$ by continuously applying (12), such that

$$H(x,\omega^{N_a},\theta^{N_a}) \leq H(x,\omega^{N_a+1},\theta^{N_a}) \leq 0, \quad \forall x \in \Omega.$$

Again, from Lemma 1, there always $\exists\theta^\#$, such that

$$H(x,\omega^{N_a+1},\theta^\#) = \min_\theta H(x,\omega^{N_a+1},\theta), \quad \forall x \in \Omega.$$

Since

$$\min_\theta L_a(\omega^{N_a+1},\theta) \geq \mathbb{E}_{x\sim d_x}\Big[\min_\theta H(x,\omega^{N_a+1},\theta)\Big],$$

we have

$$\begin{aligned} L_a(\omega^{N_a+1},\theta^\#) &= \min_\theta L_a(\omega^{N_a+1},\theta)\\ &= \mathbb{E}_{x\sim d_x}\Big[\min_\theta H(x,\omega^{N_a+1},\theta)\Big]. \end{aligned}$$

Utilizing the fact that the global minima of $L_a(\omega^{N_a+1},\theta)$ can be obtained, Hamiltonian $H(x,\omega^{N_a+1},\theta)$ can be taken to global minimum for all $x \in \Omega$ by minimizing over $\theta$. Then, we can also find $\theta^{N_a+1}$ through (13), such that

$$H(x,\omega^{N_a+1},\theta^{N_a+1}) \leq H(x,\omega^{N_a+1},\theta^{N_a}) \leq 0, \quad \forall x \in \Omega.$$

This implies that like the case with $V(x;\omega^{N_a})$, $V(x;\omega^{N_a+1})$ is also a Lyapunov function. So, $\pi(x;\theta^{N_a+1}) \in \Psi(\Omega)$. Applying this for all subsequent time steps, $V(x;\omega^i)$ is a Lyapunov function for $\forall i \geq N_a$, and it is obvious that

$$H(x,\omega^i,\theta^i) \leq H(x,\omega^{i+1},\theta^i) \leq 0, \quad \forall i \geq N_a \ \& \ \forall x \in \Omega, \tag{20}$$

and

$$\pi(x;\theta^i) \in \Psi(\Omega), \quad \forall i \geq N_a. \tag{21}$$

Thus, we have proved that starting from any initial policy, the RCTAC algorithm in Algorithm 3 converges to an admissible policy. As claimed previously, this relaxes the requirement A1, which is typical for most other ADP algorithms. □

We now present our main result, which shows that the value NN $V(x;\omega)$ and policy NN $\pi(x;\theta)$ converge to optimum uniformly by applying Algorithm 3.

**Definition 2.** *(Uniform Convergence). A function sequence $\{f_n\}$ converges uniformly to $f$ on a set $\mathbb{K}$ if $\forall \epsilon > 0$, $\exists N(\epsilon) \in \mathbb{Z}^+$, s.t. $\forall n > N$, $\sup_{x\in\mathbb{K}} |f_n(x) - f(x)| < \epsilon$.*

**Theorem 1.** *For arbitrary initial value network $V(x;\omega^0)$ and policy network $\pi(x;\theta^0)$, where all the activation functions $\sigma_V(\cdot)$ and biases $b_V$ of the value network are set to $\sigma_V(0) = 0$ and $b_V(\cdot) \equiv 0$, and the output layer activation function $\sigma_{V_{\text{out}}}$ also satisfies $\sigma_{V_{\text{out}}}(\cdot) \geq 0$, if these two NNs are tunned with Algorithm 3, $V_{\omega^i}(x) \to V^\star(x)$, $\pi_{\theta^i}(x) \to \pi^\star(x)$ uniformly on $\Omega$ as $i \to \infty$.*

*Proof.* From Lemma 3, it can be shown by induction that the policy $\pi(x;\theta^i) \in \Psi(\Omega)$ for system (1) on $\Omega$ when $i \geq N_a$. Furthermore, according to (16) and (20), given policy $\pi(x;\theta^i)$, we get

$$\frac{\mathrm{d}V(x;\omega^i)}{\mathrm{d}t} \leq \frac{\mathrm{d}V(x;\omega^{i+1})}{\mathrm{d}t} \leq 0, \quad \forall x \in \Omega \ \& \ i \geq N_a. \tag{22}$$

From Newton-Leibniz formula,

$$V_\omega(x(t)) = V_\omega(x(\infty)) - \int_t^\infty \frac{\mathrm{d}V_\omega(x(\tau))}{\mathrm{d}\tau}\mathrm{d}\tau. \tag{23}$$

According to (18) and (21),

$$V_\omega(x(\infty)) = V_\omega(0) = 0, \quad i \geq N_a \ \& \ \forall\omega. \tag{24}$$

So, from (18), (22), (23) and (24), we have

$$0 \leq V(x;\omega^{i+1}) \leq V(x;\omega^i), \quad \forall x \in \Omega \ \& \ i \geq N_a. \tag{25}$$

As such, $V(x;\omega^i)$ is pointwise convergent as $i$ goes to $\infty$. One can write $\lim_{i\to\infty} V(x;\omega^i) = V(x;\omega^\infty)$. Then, the compactness of $\Omega$ directly leads to uniform convergence by Dini's theorem [41].

Next, from Definition 2, it is not hard to show that

$$\lim_{i\to\infty} \sup_{x\in\Omega} |V(x;\omega^i) - V(x;\omega^{i+1})| = 0.$$

Furthermore, since

$$\begin{aligned} &V(x;\omega^i) - V(x;\omega^{i+1}) \\ &\quad = \int_t^\infty \frac{\mathrm{d}(V(x(\tau);\omega^{i+1}) - V(x(\tau);\omega^i))}{\mathrm{d}\tau}\mathrm{d}\tau, \\ &\quad = \int_t^\infty \Big[H(x(\tau),\omega^{i+1},\theta^i) - H(x(\tau),\omega^i,\theta^i)\Big]\mathrm{d}\tau, \end{aligned}$$

we have

$$\lim_{i\to\infty} \sup_{x\in\Omega} |H(x,\omega^{i+1},\theta^i) - H(x,\omega^i,\theta^i)| = 0. \tag{26}$$

From Lemma 1, (4) and (21),

$$\min_\omega L_c(\omega,\theta^i) = \min_\omega \mathbb{E}_{x\sim d_x}\Big[H(x,\omega,\theta^i)^2\Big] = 0, \quad \forall i \geq N_a.$$

Since $\omega^{i+1}$ is obtained by minimizing $L_c(\omega^i,\theta^i)$ using (12), according to (26), it is true that

$$\lim_{i\to\infty} H(x,\omega^i,\theta^i) = \lim_{i\to\infty} H(x,\omega^{i+1},\theta^i) = 0, \quad \forall x \in \Omega. \tag{27}$$

Therefore, $V(x;\omega^\infty)$ is the solution of the Lyapunov equation (4) when a policy $\pi(x;\theta^\infty)$ is given, and it leads to that

$$V_{\omega^\infty}(x) = V^{\pi_{\theta^\infty}}(x).$$

The policy $\pi(x;\theta^i) \in \Psi(\Omega)$ for $i \geq N_a$, so the generated state trajectory is unique due to the locally Lipschitz continuity of the dynamics [18]. Since $V(x;\omega^i)$ converges uniformly to $V(x;\omega^\infty)$, its obvious that the system trajectory converges for all $x \in \Omega$. Therefore, $\pi(x;\theta^i)$ also converges uniformly to $\pi(x;\theta^\infty)$ on $\Omega$. Since $\theta^{i+1}$ is obtained by minimizing $L_a(\omega^{i+1},\theta^i)$ using (13), it is obvious from (27) that

$$\lim_{i\to\infty} L_a(\omega^{i+1},\theta^{i+1}) = \lim_{i\to\infty} L_a(\omega^{i+1},\theta^i) = 0, \tag{28}$$

which implies that

$$\lim_{i\to\infty} \min_\theta H(x,\omega^i,\theta) = 0, \quad \forall x \in \Omega. \tag{29}$$

From (27), (29), and Lemma 2, one has $\lim_{i\to\infty} V(x;\omega^i) = V^\star(x)$ and $\lim_{i\to\infty} \pi(x;\theta^i) = \pi^\star(x)$. Therefore, one can conclude that $V_{\omega^i}(x)$ goes to $V^\star(x)$ and $\pi_{\theta^i}(x)$ goes to $\pi^\star(x)$ uniformly on $\Omega$ as $i \to \infty$, which means the global optimal solution is obtained. □

This demonstrates that the RCTAC algorithm uniformly converges to the optimal value function $V^\star(x)$ and the optimal policy $\pi^\star(x)$.

**Remark 1.** *The warm-up phase of RCTAC is developed to find the initial admissible policy, whose purpose is akin to that of the relaxing initial condition of discrete-time systems [42]. The difference is that the latter needs to repeatedly select an initial positive value function until the initial admissible policy can be constructed. It is proved that under mild restrictions of the activation functions of the value function, given any initial policy, the warm-up phase can converge to an admissible policy by continuously minimizing the square of the Hamiltonian.*

**Remark 2.** *According to the implementation process* (12) *and* (13) *of the proposed RCTAC algorithm, the dynamic $f(x,u)$ can be any analytic function such that $\frac{\partial f(x,u)}{\partial u}$ is available. As a result, RCTAC can be applied to any nonlinear system with non-affine and bounded control inputs, unlike [12], [19], [23] which are only applicable to systems with affine control inputs due to the requirement for an analytical expression of the control policy using the value function. Control constraints can be easily incorporated by using a saturated activation function (such as* $\tanh(\cdot)$*) at the output of the policy network.*

**Remark 3.** *Since the state $x$ is continuous, it is intractable to check the value of $H(x,\omega,\theta)$ for every $x \in \Omega$. Therefore, in practical applications, we usually use the expected value of $H(x,\omega,\theta)$ to judge whether each termination condition in Algorithm 3 is satisfied. So, the RCTAC algorithm can also be formulated as Algorithm 4. Fig. 1 shows the frameworks of API Algorithm 2 and RCTAC Algorithm 4.*

**Algorithm 4** RCTAC algorithm: Tractable Relaxation

```
Initialize arbitrary $\theta^0$ and $\omega^0$
Choose the appropriate learning rates $\alpha$, $\alpha_\omega$ and $\alpha_\theta$
Given any small positive number $\epsilon$ and set $i = 0$
Phase 1: Warm-up
  while $L_a(\omega^i, \theta^i) \geq \epsilon$ do
      Update $\omega$ and $\theta$ using (14)
      $i = i + 1$
  end while
Phase 2: PI with relaxed termination conditions
  while $\mathbb{E}_{x\sim d_x}|V_{\omega^i}(x) - V_{\omega^{i-1}}(x)| \geq \epsilon$ do
      Update $w^{i+1}$ using (12)
      Update $\theta^{i+1}$ using (13)
      $i = i + 1$
  end while
```

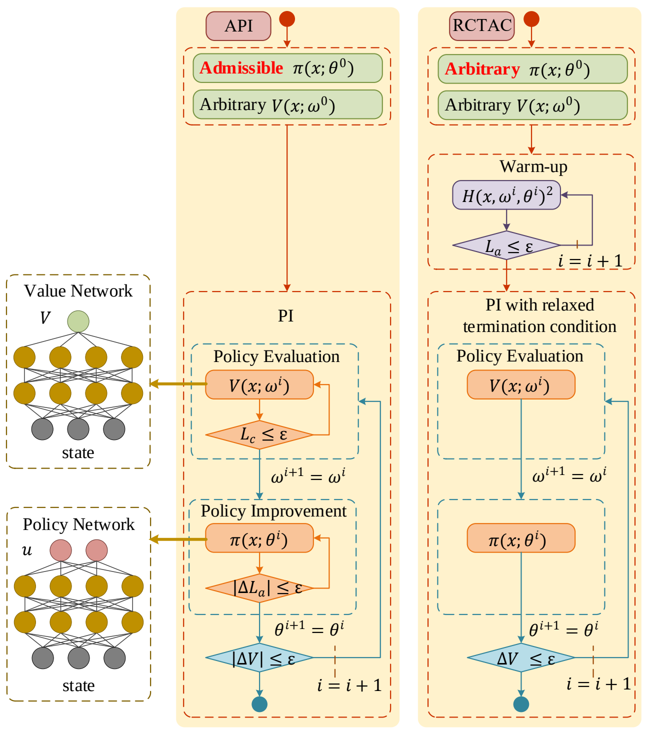

API
Admissible $\pi(x;\theta^0)$
Arbitrary $V(x;\omega^0)$
PI
Policy Evaluation
$V(x;\omega^i)$
$L_c \leq \varepsilon$
$\omega^{i+1} = \omega^i$
Policy Improvement
$\pi(x;\theta^i)$
$|\Delta L_a| \leq \varepsilon$
$\theta^{i+1} = \theta^i$
$|\Delta V| \leq \varepsilon$
$i = i + 1$
RCTAC
Arbitrary $\pi(x;\theta^0)$
Arbitrary $V(x;\omega^0)$
Warm-up
$H(x,\omega^i,\theta^i)^2$
$L_a \leq \varepsilon$
$i = i + 1$
PI with relaxed termination condition
Policy Evaluation
$V(x;\omega^i)$
$\omega^{i+1} = \omega^i$
$\pi(x;\theta^i)$
$\theta^{i+1} = \theta^i$
$\Delta V \leq \varepsilon$
$i = i + 1$
Value Network
$V$
state
Policy Network
$u$
state

Fig. 1: API and RCTAC algorithm framework diagram.

It is worth noting that for Algorithm 4, sometimes skipping Phase 1 and directly using Phase 2 can also obtain a nearly optimal policy. This is because even if $L_a(\omega^i,\theta^i) \geq \epsilon$, Phase 2 would continuously make $L_a(\omega^i,\theta^i)$ approach a non-positive number by alternately using (12) to optimize $V(x;\omega)$ and using (13) to optimize $\pi(x;\theta)$. Note that the Phase 2 of Algorithm 4 is not a CT version of the Value Iteration (VI) method. As Lee *et al.* pointed out in [12], all VI methods for CT optimal control can be deemed as a variant of integral VI. The integral ADP methods, such as integral VI, integral PI and integral generalized PI, iteratively perform policy evaluation and policy improvement updates relying on the running cost during a finite time interval [12], which is clearly different from Algorithm 3 and Algorithm 4. Besides, these integral ADP methods are usually subject to input-affine systems since these methods require that the optimal policy can be directly represented by the value function, which means that the minimum point of the Hamilton function could be solved analytically when the value function is given. This manner is difficult to extend to input non-affine systems.

**Remark 4.** *In the previous analysis, the utility function $l(x,u)$ is limited to positive definite functions, i.e., the equilibrium state (denoted by $x_e$) of the system must be $x_e = 0$. If we take $x - x_e$ as the input of value network $V(x;\omega)$, the RCTAC Algorithm 4 can be extended to problems with non-zero $x_e$, where $l(x,u) = 0$ only if $x = x_e$ & $u = 0$. The corresponding convergence and optimality analysis is similar to the problem of $x_e = 0$.*

**Remark 5.** *According to Lemma 3, all activation functions $\sigma_V$ and biases $b_V$ of $V(x;\omega)$ are set to $\sigma_V(0) = 0$ and $b_V(\cdot) \equiv 0$ to ensure $V(x_e;\omega) \equiv 0$. To remove these restrictions for value networks, we propose another effective method that drives $V(x_e;\omega)$ to gradually approach 0 by adding an equilibrium term to the critic loss function* (10)

$$L_c{}'(\omega,\theta) = \mathbb{E}_{x\sim d_x}\left[H(x,\omega,\theta)^2\right] + \eta V(x_e;\omega)^2,$$

*where $\eta$ balances the importance of the Hamiltonian term and the equilibrium term.*

## IV. Results

To verify the proposed RCTAC Algorithm 4, we offer two numerical examples, one with linear dynamics, and the other one with a nonlinear input non-affine system (the vehicle path-tracking control). We apply Algorithm 4 and Algorithm 2 to find the optimal solutions for these two systems. Results indicate that our algorithm outperforms Algorithm 2 in both cases.

### A. *Example I: Linear Time Invariant System*

*1) Description:* Consider the CT aircraft plant control problem used in [20], [21], [43], which can be formulated as

$$\min_u \quad \int_0^\infty (x^\top Q x + u^\top R u)\mathrm{d}t$$

$$\text{s.t.} \quad \dot{\mathrm{x}} = \begin{bmatrix} -1.01887 & 0.90506 & -0.00215 \\ 0.82225 & -1.07741 & -0.17555 \\ 0 & 0 & -1 \end{bmatrix} \mathrm{x} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \mathrm{u},$$

where $Q$ and $R$ are identity matrices of proper dimensions. For this linear case, the optimal analytic policy $\pi^\star(x) =$

$0.1352x_1 + 0.1501x_2 - 0.4329x_3$ and the optimal value function $V^\star(x) = x^\top Px$ can be easily found by solving the algebraic Riccati equation, where

$$P = \begin{bmatrix} 1.4245 & 1.1682 & -0.1352 \\ 1.1682 & 1.4349 & -0.1501 \\ -0.1352 & -0.1501 & 0.4329 \end{bmatrix}.$$

*2) Algorithm Details:* This system is very special, in particular, if the weights of the policy NN are randomly initialized around 0, which is a very common initialization method, then the initialized policy $\pi(x;\theta^0) \in \Psi(\Omega)$. Therefore, to compare the learning speed of Algorithm 2 and Algorithm 4, both algorithms are implemented to find the nearly optimal policy and value function. We approximate the value function and policy using 3-layer fully-connected NNs. Each NN contains 2 hidden layers with 256 units per layer, activated by exponential linear units (ELUs). The outputs of the value and policy network are $V(x;\omega)$ and $\pi(x;\theta)$, using softplus unit and linear unit as activation functions, respectively. The training set consists of 256 states which are randomly chosen from the compact set $\Omega$ at each iteration. We set the stepsizes $\alpha_\omega$ and $\alpha_\theta$ to 0.01 and employ Adam for NN optimization.

*3) Result:* Fig. 2 displays the learning accuracy of the policy and value function at each iteration, measured by

$$e_\pi = \mathbb{E}_{x\in X_0}\left[\frac{|\pi^\star(x) - \pi_\theta(x)|}{\max\limits_{x\in X_0} \pi^\star(x) - \min\limits_{x\in X_0} \pi^\star(x)}\right],$$

$$e_V = \mathbb{E}_{x\in X_0}\left[\frac{|V^\star(x) - V_\omega(x)|}{\max\limits_{x\in X_0} V^\star(x) - \min\limits_{x\in X_0} V^\star(x)}\right],$$

where there are 500 states in the test set $X_0$, which are randomly chosen from $\Omega$. We also draw violin plots in different iterations to show the precision distribution and 4-quartiles.

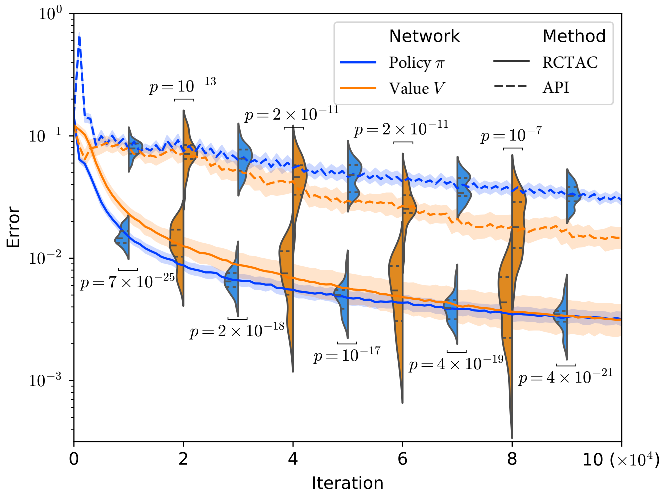
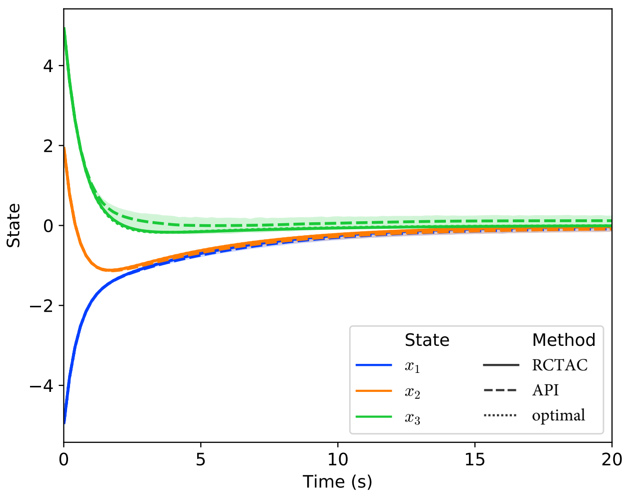
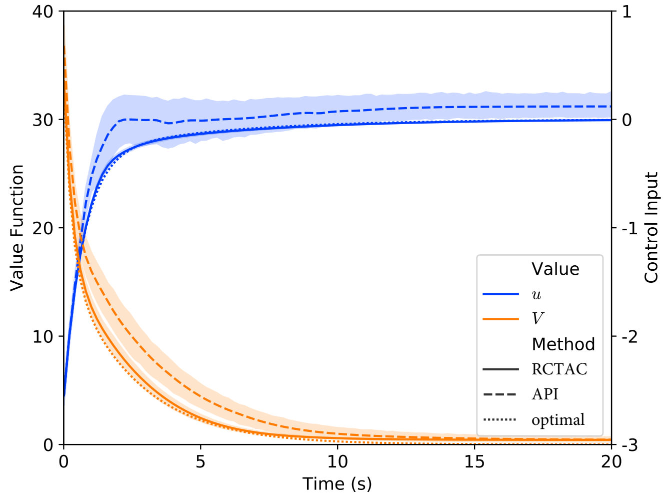


Fig. 2: RCTAC vs API performance comparison: Example I. Solid lines and shaded regions correspond to average values and 95% confidence interval over 20 runs. One iteration corresponds to one NN update.

It is clear from Fig. 2 that both two algorithms can make the value and policy network approximation errors ($e_\pi$ and $e_V$) fall with iteration. And after $10^5$ iterations, both errors of Algorithm 4 are less than 0.4%. This indicates that Algorithm 4 has the ability to converge value function and policy to nearly optimal solutions. In addition, the t-test results in Fig. 2 show that both $e_\pi$ and $e_V$ of Algorithm 4 are significantly smaller than those of Algorithm 2 ($p < 0.001$) under the same number of iterations. From the perspective of convergence speed, Algorithm 4 requires only about $10^4$ iterations to make both approximation errors less than 0.03, while Algorithm 2 requires around $10^5$ steps. Based on this, Algorithm 4 is about 10 times faster than Algorithm 2.



(a)



(b)

Fig. 3: Simulation results of different methods over 20 runs: Example I. (a) State trajectory. (b) Value function and control input.

Fig. 3 shows the simulation results of the learned policy (after $10^5$ iterations) over 20 runs. It is obvious that policies learned by RCTAC perform much better than API. The curves of state, control input, and cost generated by RCTAC are almost consistent with the optimal solution. In summary, Algorithm 4 is able to attain nearly optimal solutions and enjoys a significantly faster convergence speed compared to

Algorithm 2.

## B. Example II: Nonlinear and Input Non-Affine System

*1) Description:* Consider the vehicle path tracking task with nonlinear and input non-affine vehicle system derived in [44], [45]. The desired velocity is 12 m/s and the desired tracking path is shown in Fig. 5. Table I provides a detailed description of the states and inputs for this task, and Table II lists the vehicular parameters. The vehicle's movement is controlled by a bounded actuator, where $a_x \in [-3, 3]\ \mathrm{m/s}^2$ and $\delta \in [-0.35, 0.35]$ rad. The dynamics of the vehicle is given by

$$x = \begin{bmatrix} v_y \\ r \\ v_x \\ \phi \\ y \end{bmatrix}, u = \begin{bmatrix} \delta \\ a_x \end{bmatrix}, f(x,u) = \begin{bmatrix} \frac{F_{y\mathrm{r}}+F_{y\mathrm{f}}\cos\delta}{m} - v_x r \\ \frac{-bF_{y\mathrm{r}}+aF_{y\mathrm{f}}\cos\delta}{I_z} \\ v_y r + a_x - \frac{F_{y\mathrm{f}}\sin\delta}{m} \\ r \\ v_y\cos\phi + v_x\sin\phi \end{bmatrix},$$

where $F_{y\ddagger}$ represents the lateral tire force and the subscript $\ddagger \in \{\mathrm{f,r}\}$ corresponds to the front or rear wheels. It is calculated using the Fiala model:

$$\begin{aligned} F_{y\ddagger} =& -\operatorname{sgn}(\alpha_\ddagger)\min\Bigg\{|\mu_\ddagger F_{z\ddagger}|, \\ & \left|\tan(\alpha_\ddagger)C_\ddagger\left(1+\frac{C_\ddagger^2(\tan\alpha_\ddagger)^2}{27(\mu_\ddagger F_{z\ddagger})^2}-\frac{C_\ddagger|\tan\alpha_\ddagger|}{3\mu_\ddagger F_{z\ddagger}}\right)\right|\Bigg\}, \end{aligned}$$

where $F_{z\ddagger}$ denotes the tire load, $\mu_\ddagger$ denotes the lateral friction coefficient, and $\alpha_\ddagger$ denotes the tire slip angle with

$$\alpha_\mathrm{f} = -\delta + \arctan(\frac{v_y + ar}{v_x}), \quad \alpha_\mathrm{r} = \arctan(\frac{v_y - br}{v_x}).$$

The lateral friction coefficient is estimated by:

$$\mu_\ddagger = \frac{\sqrt{(\mu F_{z\ddagger})^2 - F_{x\ddagger}^2}}{F_{z\ddagger}},$$

where $F_{x\ddagger}$ represents the longitudinal tire force, expressed as

$$F_{x\mathrm{f}} = \begin{cases} 0, & a_x \geq 0 \\ \frac{ma_x}{2}, & a_x < 0 \end{cases}, \quad F_{x\mathrm{r}} = \begin{cases} ma_x, & a_x \geq 0 \\ \frac{ma_x}{2}, & a_x < 0 \end{cases}.$$

The vertical loads on the front and rear wheels are approximated by

$$F_{z\mathrm{f}} = \frac{b}{a+b}mg, \quad F_{z\mathrm{r}} = \frac{a}{a+b}mg.$$

The control objective is to minimize output tracking errors, which is formulated as

$$\begin{aligned} \min_u \quad & \int_0^\infty \left\{0.4(v_x - 12)^2 + 80y^2 + u^\top \begin{bmatrix} 280 & 0 \\ 0 & 0.3 \end{bmatrix} u\right\}\mathrm{d}t \\ & \text{subject to} \quad \dot{x} = f(x,u). \end{aligned}$$

TABLE I: List of state and input

| | | |
|---|---|---|
| Lateral velocity | $v_y$ | [m/s] |
| Yaw rate at CG (center of gravity) | $r$ | [rad/s] |
| Heading angle between trajectory & vehicle | $\phi$ | [rad] |
| Longitudinal velocity | $v_x$ | [m/s] |
| Distance between trajectory & CG | $y$ | [m] |
| Front wheel angle | $\delta$ | [rad] |
| Longitudinal acceleration | $a_x$ | [m/s$^2$] |

TABLE II: Vehicle parameters

| | | |
|---|---|---|
| Cornering stiffness at front wheel | $C_\mathrm{f}$ | -88000 [N/rad] |
| Cornering stiffness at rear wheel | $C_\mathrm{r}$ | -94000 [N/rad] |
| Mass | $m$ | 1500 [kg] |
| Distance from front axle to CG | $a$ | 1.14 [m] |
| Distance from rear axle to CG | $b$ | 1.40 [m] |
| Polar moment of inertia of CG | $I_z$ | 2420 [kg·m$^2$] |
| Tire-road friction coefficient | $\mu$ | 1.0 |

*2) Algorithm Details:* We employ 6-layer fully-connected NNs to represent $V_\omega$ and $\pi_\theta$, and the state input layer of each NN is followed by 5 fully-connected hidden layers, 32 units per layer. The activation function selection for the policy network is similar to that in Example I, with the exception that the output layer is activated by $\tanh(\cdot)$ with two units, multiplied by the vector $[0.35, 3]$ to accommodate the constrained control inputs. The training set consists of the current states of 256 parallel independent vehicles with different initial states, thereby obtaining a more realistic state distribution. We use Adam method to update two NNs, while the learning rates of value network and policy network are set to $8 \times 10^{-4}$ and $2 \times 10^{-4}$, respectively. Besides, we use $\eta = 0.1$ to trade off the Hamiltonian term and the equilibrium term of the critic loss function (Remark 5).

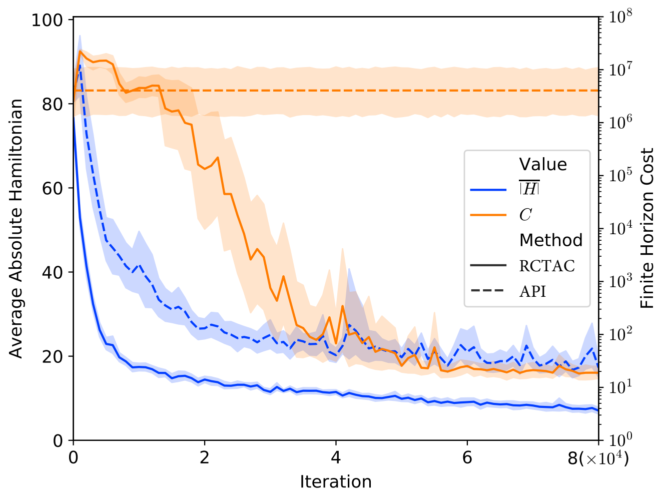


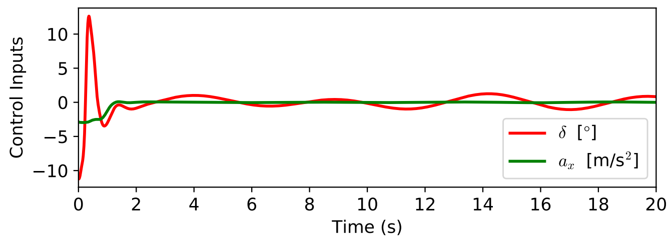
Fig. 4: RCTAC vs API performance comparison over 20 runs: Example II.

*3) Result:* Fig. 4 shows the evolution of the average absolute Hamiltonian $\overline{|H|}$ of 256 random states and the training performance. The policy performance at each iteration is calculated by the cumulative running cost in 20s time domain

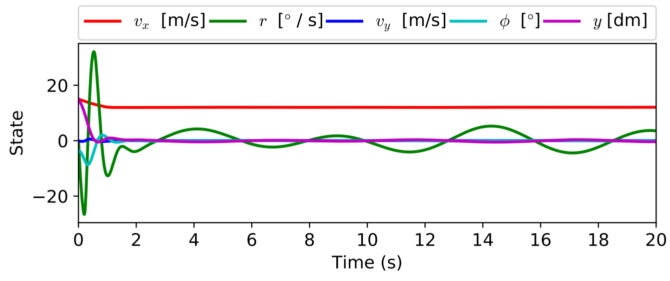
$$C = \int_0^{20} l(x(\tau), u(\tau))\mathrm{d}\tau,$$

where the initial state is randomly selected for each run. Since the initial policy is *not* admissible, i.e., $\pi(x;\theta^0) \notin \Psi(\Omega)$, Algorithm 2 can never make $\overline{|H|}$ close to 0, hence the terminal condition of policy evaluation can never be satisfied. Therefore, the finite horizon cost $C$ has no change during the entire learning process, i.e., Algorithm 2 can never converge to an admissible policy if $\pi(x;\theta^0) \notin \Psi(\Omega)$.

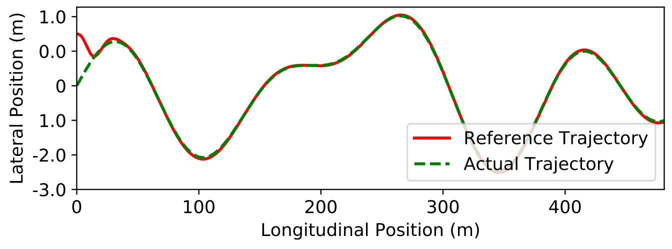
On the other hand, $\overline{|H|}$ of Algorithm 4 can gradually converge to 0, while the finite horizon cost $C$ is also reduced to a small value during the learning process. Fig. 5 shows the control inputs and state trajectory controlled by the learned RCTAC policy. It can quickly guide the vehicle to the desired state, taking less than 0.5s for the case in Fig. 5. The results of Example II show that Algorithm 4 can solve the CT dynamic optimal control problem for general nonlinear and input non-affine CT systems with saturated actuators and handle inadmissible initial policies.



(a)



(b)



(c)

Fig. 5: Simulation results: Example II. (a) Control inputs. (b) State trajectory. (c) Vehicle trajectory.

In conclusion, these two examples demonstrate that the proposed RCTAC method is able to learn the nearly optimal policy and value function for general nonlinear and input non-affine CT systems without reliance on initial admissible policy. In addition, if the initial policy $\pi(x;\theta^0) \in \Psi(\Omega)$, the learning speed of Algorithm 4 is also faster than that of Algorithm 2.

## V. Experimental Validation

In this section, we demonstrate the practical effectiveness of RCTAC by using the real-world path-tracking task [46], [47]. The experiment pipeline is shown in Fig. 6. The test vehicle is GEELY Geometry C, which is equipped with a driving mode button, enabling it to switch between manual driving mode and automatic driving mode. We deploy the learned policy on the onboard industrial personal computer (IPC) with a 3.60 GHz Intel Core i7-7700 CPU. The deployed policy network is trained by the proposed RCTAC algorithm, where the expected velocity is set to 15 km/h, while other vehicle and algorithm parameters are consistent with Example II in Section IV-B. Provided with an ASENSING INS570D high-precision vehicle-mounted positioning system, the state of the vehicle can be accurately measured. After receiving the vehicle state information, the IPC records and calculates control instructions every 80ms, and sends the desired front wheel angle and longitudinal acceleration through the Controller Area Network (CAN), so as to realize the lateral and longitudinal control of the vehicle.

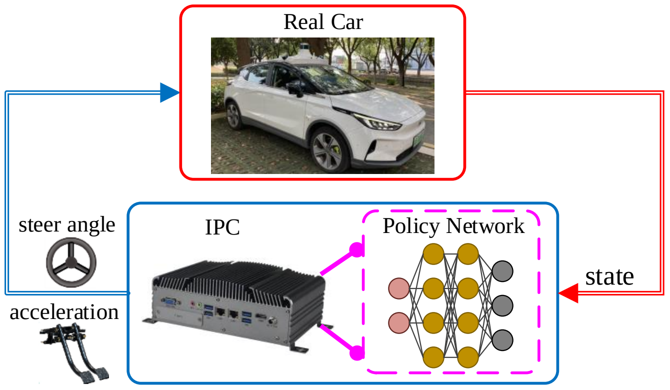


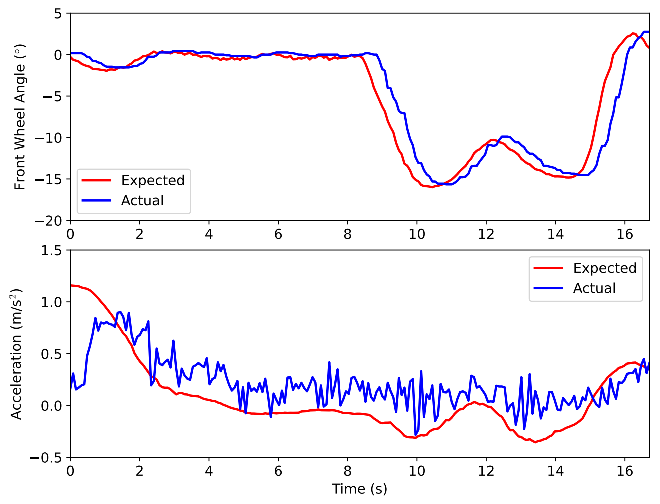
Fig. 6. Real vehicle test pipeline.

The control inputs and state trajectory controlled by the deployed RCTAC policy are shown in Fig. 7. From Fig. 7a, there exists a system response delay of about 0.5s between the expected and the actual control signal. Besides, the actual acceleration signals are noisy due to hardware limitations. Despite these difficulties, our policy still makes the vehicle track the desired speed and trajectory smoothly and well (Fig. 7b, 7c).

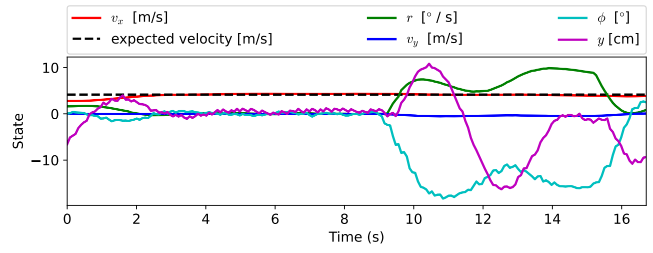
For comparison, model predictive control (MPC) with a prediction time domain of 20 steps is also deployed on the IPC to carry out practical experiments [48], where the well-known nonlinear programs solver IPOPT [49] is employed to solve the constructed MPC problem. The average results of 10 independent experiments are shown in Table III. The root mean squared error (RMSE) of the heading angle between vehicle & trajectory and that of the distance between CG & trajectory

$$\phi_{\text{RMSE}} = \sqrt{\frac{1}{m}\sum_{k=1}^{m}\phi_k^2},\ y_{\text{RMSE}} = \sqrt{\frac{1}{m}\sum_{k=1}^{m}y_k^2},$$

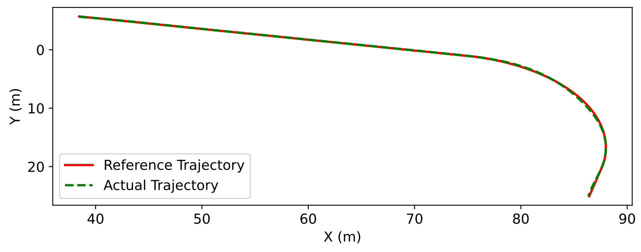
are calculated to quantify the tracking performance. It can be found that RCTAC matches MPC in terms of tracking



(a)



(b)



(c)

Fig. 7: Experiment results: (a) Control inputs. The expected value corresponds to the policy output. (b) State trajectory. (c) Vehicle trajectory.

performance. Moreover, RCTAC shows great advantages in online decision-making efficiency, whose average single-step decision time is 91.17% less than that of MPC.

TABLE III: Comparison Results

| | $\phi_{\text{RMSE}}$ [rad] | $y_{\text{RMSE}}$ [m] | single-step decision time [ms] |
|---|---|---|---|
| RCTAC | 0.0797 | 0.0568 | 3.67 |
| MPC | 0.0809 | 0.0601 | 41.57 |

In conclusion, the vehicle experiment verifies the control effect of the proposed RCTAC algorithm in practical applications. It performs as well as MPC in our path-tracking task. Moreover, the way of offline training and online application makes the online calculation time of RCTAC much shorter than that of online optimization methods, such as MPC. This property is of significant importance for systems with limited computing resources.

## VI. Conclusion

The paper proposed the relaxed continuous-time actor-critic (RCTAC) Algorithm 4, along with the proof of convergence and optimality, for solving nearly optimal control problems of general nonlinear CT systems with known dynamics. The proposed algorithm can circumvent the requirements of "admissibility" and input-affine system dynamics (described in A1 and A2 of Introduction), quintessential to previously proposed counterpart algorithms. As a result, given an arbitrary initial policy, the RCTAC algorithm is shown to eventually converge to a nearly optimal policy, even for general nonlinear input non-affine system dynamics. The convergence and optimality are mathematically proven by using detailed Lyapunov analysis. We further demonstrate the efficacy and theoretical accuracy of our algorithm via two numerical examples, which yields a faster learning speed of the nearly optimal policy starting from an admissible initialization, as compared to the traditional approximate policy iteration (API) algorithm (Algorithm 2). In addition, a real-world path-tracking experiment is conducted to verify the practical performance of our method. Results show that compared with the MPC method, RCTAC has reduced the single-step decision time by 91.17% without losing tracking performance. This indicates that our method has the potential to be applied in practical systems with limited computing resources.

## Appendix

We would like to acknowledge Prof. Francesco Borrelli, Ms. Ziyu Lin, Dr. Yiwen Liao, Dr. Jiatong Xu, and Dr. Xiaojing Zhang for their valuable suggestions.

## References

[1] D. Liu, Q. Wei, D. Wang, X. Yang, and H. Li, *Adaptive Dynamic Programming with Applications in Optimal Control*. London, UK: Springer, 2017.
[2] T. Pappas, A. Laub, and N. Sandell, "On the numerical solution of the discrete-time algebraic riccati equation," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 631–641, 1980.
[3] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*, 3rd ed. New York USA: Wiley, 2012.
[4] F. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Computational Intelligence Magazine*, vol. 4, no. 2, pp. 39–47, 2009.
[5] P. Werbos, "Approximate dynamic programming for realtime control and neural modelling," *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, pp. 493–525, 1992.
[6] J. Duan, S. Eben Li, Y. Guan, Q. Sun, and B. Cheng, "Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data," *IET Intelligent Transport Systems*, vol. 14, no. 5, pp. 297–305, 2020.
[7] S. E. Li, *Reinforcement Learning for Sequential Decision and Optimal Control*. Singapore: Springer Verlag, 2023.
[8] J. Duan, Y. Guan, S. E. Li, Y. Ren, Q. Sun, and B. Cheng, "Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 11, pp. 6584–6598, 2021.
[9] X. He, H. Yang, Z. Hu, and C. Lv, "Robust lane change decision making for autonomous vehicles: An observation adversarial reinforcement learning approach," *IEEE Transactions on Intelligent Vehicles*, to be published, doi: 10.1109/TIV.2022.3165178.

[10] Y. Li, S. He, Y. Li, L. Ge, S. Lou, and Z. Zeng, "Probabilistic charging power forecast of evcs: Reinforcement learning assisted deep learning approach," *IEEE Transactions on Intelligent Vehicles*, to be published, doi: 10.1109/TIV.2022.3168577.

[11] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA, USA: MIT Press, 2018.

[12] J. Y. Lee, J. B. Park, and Y. H. Choi, "On integral value iteration for continuous-time linear systems," in *American Control Conference (ACC)*. Washington, DC, USA: IEEE, 2013, pp. 4215–4220.

[13] K. Min, H. Kim, and K. Huh, "Deep distributional reinforcement learning based high-level driving policy determination," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 3, pp. 416–424, 2019.

[14] 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, 2022.

[15] B. Fan, Q. Yang, X. Tang, and Y. Sun, "Robust ADP design for continuous-time nonlinear systems with output constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2127–2138, 2018.

[16] J. Na, B. Wang, G. Li, S. Zhan, and W. He, "Nonlinear constrained optimal control of wave energy converters with adaptive dynamic programming," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 10, pp. 7904–7915, 2018.

[17] J. Sun and C. Liu, "Backstepping-based adaptive dynamic programming for missile-target guidance systems with state and input constraints," *Journal of the Franklin Institute*, vol. 355, no. 17, pp. 8412–8440, 2018.

[18] M. Abu-Khalaf and F. L. Lewis, "Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach," *Automatica*, vol. 41, no. 5, pp. 779–791, 2005.

[19] T. Dierks and S. Jagannathan, "Optimal control of affine nonlinear continuous-time systems," in *American Control Conference (ACC)*. Baltimore, MD, USA: IEEE, 2010, pp. 1568–1573.

[20] K. G. Vamvoudakis and F. L. Lewis, "Online actor critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[21] K. G. Vamvoudakis, "Event-triggered optimal adaptive control algorithm for continuous-time nonlinear systems," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 3, pp. 282–293, 2014.

[22] L. Dong, X. Zhong, C. Sun, and H. He, "Event-triggered adaptive dynamic programming for continuous-time systems with control constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1941–1952, 2017.

[23] X. Yang, D. Liu, and Q. Wei, "Online approximate optimal control for affine non-linear systems with unknown internal dynamics using adaptive dynamic programming," *IET Control Theory & Applications*, vol. 8, no. 16, pp. 1676–1688, 2014.

[24] Y. Jiang and Z.-P. Jiang, "Global adaptive dynamic programming for continuous-time nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 60, no. 11, pp. 2917–2929, 2015.

[25] S. Xue, B. Luo, D. Liu, and Y. Li, "Adaptive dynamic programming based event-triggered control for unknown continuous-time nonlinear systems with input constraints," *Neurocomputing*, vol. 396, pp. 191–200, 2020.

[26] H. Jiang and B. Zhou, "Bias-policy iteration based adaptive dynamic programming for unknown continuous-time linear systems," *Automatica*, vol. 136, pp. 1–12, 2022.

[27] N. Wang, Y. Gao, H. Zhao, and C. K. Ahn, "Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 3034–3045, 2020.

[28] K. G. Vamvoudakis, F. L. Lewis, and G. R. Hudas, "Multi-agent differential graphical games: Online adaptive learning solution for synchronization with optimality," *Automatica*, vol. 48, no. 8, pp. 1598–1611, 2012.

[29] J. Li, H. Modares, T. Chai, F. L. Lewis, and L. Xie, "Off-policy reinforcement learning for synchronization in multiagent graphical games," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2434–2445, 2017.

[30] J. Shi, D. Yue, and X. Xie, "Data-based optimal coordination control of continuous-time nonlinear multi-agent systems via adaptive dynamic programming method," *Journal of the Franklin Institute*, vol. 357, no. 15, pp. 10 312–10 328, 2020.

[31] J. Lu, Q. Wei, T. Zhou, Z. Wang, and F.-Y. Wang, "Event-triggered near-optimal control for unknown discrete-time nonlinear systems using parallel control," *IEEE Transactions on Cybernetics*, 2022.

[32] X. Yang, Z. Zeng, and Z. Gao, "Decentralized neurocontroller design with critic learning for nonlinear-interconnected systems," *IEEE Transactions on Cybernetics*, to be published, doi: 10.1109/TCYB.2021.3085883.

[33] J. Lu, Q. Wei, and F.-Y. Wang, "Parallel control for optimal tracking via adaptive dynamic programming," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 6, pp. 1662–1674, 2020.

[34] R. W. Beard, G. N. Saridis, and J. T. Wen, "Galerkin approximations of the generalized Hamilton-Jacobi-Bellman equation," *Automatica*, vol. 33, no. 12, pp. 2159–2177, 1997.

[35] S. Lyashevskiy, "Constrained optimization and control of nonlinear systems: new results in optimal control," in *Proceedings of 35th IEEE Conference on Decision and Control (CDC)*. Kobe, Japan: IEEE, 1996, pp. 541–546.

[36] Y. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[37] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*. Long Beach, CA, USA: PMLR, 2019, pp. 242–252.

[38] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*. Long Beach, CA, USA: PMLR, 2019, pp. 1675–1685.

[39] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, no. 5, pp. 551–560, 1990.

[40] A. M. Lyapunov, "The general problem of the stability of motion," *International Journal of Control*, vol. 55, no. 3, pp. 531–534, 1993.

[41] R. G. Bartle and D. R. Sherbert, *Introduction to real analysis*, 3rd ed. New York, NY, USA: Wiley, 2000.

[42] D. Liu, Q. Wei, and P. Yan, "Generalized policy iteration adaptive dynamic programming for discrete-time nonlinear systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 12, pp. 1577–1591, 2015.

[43] B. L. Stevens and F. L. Lewis, *Aircraft control and simulation*. New York, NY, USA: Wiley, 2004.

[44] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *IEEE Intelligent Vehicles Symposium (IV)*. Seoul, South Korea: IEEE, 2015, pp. 1094–1099.

[45] R. Li, Y. Li, S. Li, E. Burdet, and B. Cheng, "Driver-automation indirect shared control of highly automated vehicles with intention-aware authority transition," in *IEEE Intelligent Vehicles Symposium (IV)*. Los Angeles, CA, USA: IEEE, 2017, pp. 26–32.

[46] L. Li, W.-L. Huang, Y. Liu, N.-N. Zheng, and F.-Y. Wang, "Intelligence testing for autonomous vehicles: A new approach," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 158–166, 2016.

[47] S. Ge, F.-Y. Wang, J. Yang, Z. Ding, X. Wang, Y. Li, S. Teng, Z. Liu, Y. Ai, and L. Chen, "Making standards for smart mining operations: Intelligent vehicles for autonomous mining transportation," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 413–416, 2022.

[48] Y. Lin, J. McPhee, and N. L. Azad, "Comparison of deep reinforcement learning and model predictive control for adaptive cruise control," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 221–231, 2021.

[49] A. Wachter and L. T. Biegler, "Biegler, l.t.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. mathematical programming 106, 25-57," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.