Model-Based Actor-Critic Learning for Optimal Tracking Control of Robots With Input Saturation

Xingwei Zhao, Bo Tao, *Member, IEEE*, Lu Qian, and Han Ding, *Senior Member, IEEE*

***Abstract*—As robots normally perform repetitive work, reinforcement learning (RL) appears to be a promising tool for designing robot control. However, the learning cycle of control strategy tends to be long, thereby limiting the applications of RL for a real robotic system. This article proposes model-based actor-critic learning for optimal tracking control of robotic systems to address this limitation. A preconstructed critic is defined in the framework of linear quadratic tracker, and a model-based actor update law is presented on the basis of deterministic policy gradient algorithm to improve learning efficiency. A low gain parameter is introduced in the critic to avoid input saturation. Compared with neural network-based RL, the proposed method including the preconstructed critic and actor, has rapid, steady, and reliable learning process, which is friendly for the physical hardware. The performance and effectiveness of the proposed method are validated using a dual-robot test rig. The experimental results show that the proposed learning algorithm can train multiple robots to learn their optimal tracking control laws within a training time of 200 s.**

***Index Terms*—Actor-critic, linear quadratic tracker (LQT), reinforcement learning (RL), robot.**

I. Introduction

ROBOT manipulators can help human workers in complicated and repetitive tasks in various industrial processes, such as assembling [1], assisting [2], transportation [3], and machining [4]. As robots normally perform repetitive work, reinforcement learning (RL) is suitable for designing robot control. RL is a machine learning method used to find an optimal policy in an uncertain environment [5]–[7]. In the control system society, Werbos. first used the RL to seek solutions for the optimal regulator problem of discrete-time systems [8], [9]. Hagen and Krose [10] combined Q-learning and linear quadratic regulator (LQR) framework and proved that the learnt controller is equivalent to a model-based adaptive controller. Al-Tamimi *et al.* [11], [12] confirmed the convergence of Q-learning algorithm for the optimal control design of nonlinear discrete-time systems.

RL is widely used for robotic control problems, such as robotic biped dynamic walking [13], peg-in-hole insertion task [14], locomotion [15] human-robot interaction [16], and stable flight of an autonomous helicopter [17]. The studies use policy gradient algorithm to find the optimal policy in the continue action space. Peter *et al.* [18], [19] proposed a natural actor-critic approach on the basis of the policy gradient algorithm, where a critic is introduced to optimize the policy update step during policy gradient estimation. As an improvement of the policy gradient method, Silver *et al.* [20] presented the deterministic policy gradient algorithm, which can be considered as one of the core algorithms of deep RL. Deep RL shows excellent capabilities in complex tasks, such as playing a range of Atari games or defeating a human world champion in Go [21]–[23]. Lillicrap *et al.* demonstrated that deep RL can accomplish different dexterous manipulation tasks [24], such as robotic door opening task [25]. Parisi *et al.* [26] focused on the interaction between actor and critic learning and proposed a temporal difference regularized actor-critic method to improve the stability of the learning process.

Although RL has been effectively applied to solve various problems, it has not been widely used to deal with optimal tracking control. The design of optimal tracking control consists of two components, a feedback term and a feedforward term [5]. Considering that RL is based on the hypothesis that the system is causal, noncausal feedforward control poses a challenge for RL. Zhang *et al.* [27] transformed an optimal tracking problem into an optimal regulation problem to realize infinite horizon optimal tracking control of discrete-time nonlinear systems. Dierks and Jagannathan [28] directly used dynamic programming to achieve optimal tracking control of affine nonlinear systems. Wang *et al.* [29] proposed a neuro-optimal control strategy for nonlinear systems, where neural networks (NNs) act as parametric structures to approximate the cost function, control law, and error dynamics. In [30], RL was applied to design the output feedback control for nonlinear systems, where three NNs are constructed, namely, an observer NN, a critic NN, and an action NN, for the output feedback controller. In the abovementioned studies, although RL is used to solve the optimal tracking control, feedforward control is normally based on the dynamic inversion concept, which requires the control

Manuscript received July 24, 2019; revised November 20, 2019, February 9, 2020, and March 4, 2020; accepted April 20, 2020. Date of publication May 7, 2020; date of current version February 17, 2021. This work was supported in part by the National Science Foundation of China under Grant 91748204, Grant 51905184, and Grant 91948301, in part by the National Key Research and Development Program of China under Grant 2017YFB1301504. *(Corresponding author: Bo Tao.)*

Xingwei Zhao, Bo Tao, and Han Ding are with the State Key Laboratory of Digital Manufacturing Equipment and Technology, School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: zhaoxingwei@hust.edu.cn; taobo@hust.edu.cn; dinghan@hust.edu.cn).

Lu Qian is with the School of Logistics Engineering, Wuhan University of Technology, Wuhan 430063, China (e-mail: qianlu@whut.edu.cn).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2020.2992003

input matrix to be invertible and complete knowledge of system dynamics to be *a priori* identified. Kiumarsi *et al.* [31] assumed that the reference trajectory is generated using a linear command generator, and solved the optimal tracking problem through Q-learning by converting the linear quadratic tracker (LQT) into LQR. This article focuses on theoretical studies rather than on experimental investigations. Issues, such as environmental disturbances [32], input dead zone [33], input saturation [34], [35], and physical constraints [36], [37], which may occur in real robotic systems, are not considered in the study.

Motivated by the above observations, we propose a model-based actor-critic learning for the tracking control of robotic systems. To the best of our knowledge, the model-based actor update law for the LQT issue is first proposed in this article, which is on the basis of the deterministic policy gradient algorithm. In addition, to implement the model-based actor-critic learning on real robotic systems, we consider factors such as environmental disturbances, input saturation, and physical constraints in the algorithm. The LQT problem of robots can be solved by the model-based actor-critic learning without the knowledge of system parameters. The objectives of this article are to analyze the stability and reliability of the proposed model-based actor-critic learning for the LQT issue and to demonstrate the feasibility of the proposed learning algorithm for real robotic systems.

II. Review of Tracking Control With Input Saturation

The dynamics of a linear discrete system subject to input saturation can be written as

$$\boldsymbol{X}_{k+1} = \boldsymbol{A}\boldsymbol{X}_k + \boldsymbol{B}\sigma\left(\boldsymbol{u}_k\right) + \boldsymbol{\varepsilon}_k \tag{1}$$

where $\boldsymbol{X}_k$ is the state that indicates the angle and angular velocity of the joint with respect to the robotic system, $\boldsymbol{u}_k$ denotes the control input, $\boldsymbol{A}$ and $\boldsymbol{B}$ are constant matrices with compatible dimensions, $\boldsymbol{\varepsilon}_k$ represents the unknown noise that is assumed to be Gaussian, and $\sigma(\boldsymbol{u}_k)$ is a saturation function defined as

$$\begin{aligned}\sigma\left(\boldsymbol{u}_k\right) &= \left[\operatorname{sat}_s\left(u_k^1\right), \operatorname{sat}_s\left(u_k^2\right), \ldots, \operatorname{sat}_s\left(u_k^n\right)\right]\\ \operatorname{sat}_s\left(\boldsymbol{u}_k\right) &= \begin{cases} -s, \text{ if } \mathrm{u}_\mathrm{k}^\mathrm{j} < -\mathrm{s} \\ u_k, \text{ if } -\mathrm{s} \leq \mathrm{u}_\mathrm{k}^\mathrm{j} \leq \mathrm{s} \\ s, \text{ if } \mathrm{u}_\mathrm{k}^\mathrm{j} > \mathrm{s} \end{cases}\end{aligned} \tag{2}$$

where s is the saturation limit.

The reference trajectory is produced using a command generator model, which is expressed as

$$\boldsymbol{r}_{k+1} = \boldsymbol{F}\boldsymbol{r}_k \tag{3}$$

where $\boldsymbol{r}_k$ is the desired trajectory, $\boldsymbol{F}$ is a constant, Hurwitz matrix.

Then, the tracking error is defined as

$$\boldsymbol{e}_k = \boldsymbol{X}_k - \boldsymbol{r}_k. \tag{4}$$

Inspired by Zhang *et al.* [27], we define the tracking control corresponding to reference trajectory $\boldsymbol{r}_k$ as

$$\boldsymbol{u}_k = \sigma\left(\boldsymbol{u}_{b,k} + \boldsymbol{u}_{f,k}\right) \tag{5}$$

where the feedforward term is

$$\boldsymbol{u}_{f,k} = -\boldsymbol{B}^{-1}\left(\boldsymbol{A} - \boldsymbol{F}\right)\boldsymbol{r}_k \tag{6}$$

and the feedback term is

$$\boldsymbol{u}_{b,k} = \boldsymbol{K}_b\boldsymbol{e}_k \tag{7}$$

where $\boldsymbol{K}_b$ is a feedback gain.

The error dynamics can be written as

$$\boldsymbol{e}_{k+1} = \left(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_b\right)\boldsymbol{e}_k \tag{8}$$

which approaches zero when $\boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}_b$ is Hurwitz.

Remark 1: To minimize the tracking error, the optimal tracker control normally consists of two components: a feedback term (7) obtained by solving an algebraic Riccati equation (ARE) and a feedforward term (6) is generally solved based on the dynamic inversion concept. The disadvantage of the dynamic inversion concept is that the control input matrix should be invertible and complete knowledge of the system dynamics need to be *a priori* identified.

III. Model-Based Actor-Critic Learning Algorithm

In this section, a model-based actor-critic learning algorithm is proposed based on the LQT framework, where the system tracks a reference trajectory in an optimal sense by minimizing a predefined performance index. The LQT problem can be converted to an LQR problem by formulating the tracking control problem in a causal manner and be solved without known system parameters. Combining the robot system (1) with trajectory dynamics (3), the augmented system is given as

$$\begin{aligned}\begin{bmatrix}\boldsymbol{X}_{k+1}\\ \boldsymbol{r}_{k+1}\end{bmatrix} &= \bar{\boldsymbol{A}}\begin{bmatrix}\boldsymbol{X}_k\\ \boldsymbol{r}_k\end{bmatrix} + \bar{\boldsymbol{B}}\sigma\left(\boldsymbol{u}_k\right) + \bar{\boldsymbol{\varepsilon}}_k\\ \bar{\boldsymbol{A}} = \begin{bmatrix}\boldsymbol{A} & 0\\ 0 & \boldsymbol{F}\end{bmatrix}, \ \bar{\boldsymbol{B}} &= \begin{bmatrix}\boldsymbol{B}\\ 0\end{bmatrix}, \ \bar{\boldsymbol{\varepsilon}}_k = \begin{bmatrix}\boldsymbol{\varepsilon}_k\\ 0\end{bmatrix}.\end{aligned} \tag{9}$$

To minimize the tracking error, the LQT index is defined as

$$\begin{aligned}&\min c_k\\ &c_k = \tfrac{1}{2}\delta(\boldsymbol{X}_k - \boldsymbol{r}_k)^T\left(\boldsymbol{X}_k - \boldsymbol{r}_k\right) + \tfrac{1}{2}\boldsymbol{u}_k{}^T\boldsymbol{u}_k\end{aligned} \tag{10}$$

where $\delta \in (0, \infty)$ is a low gain parameter [38] to adjust the input gain for avoiding input saturation.

The value function can be defined as the accumulated cost

$$V\left(\boldsymbol{X}_k, \boldsymbol{r}_k\right) = \frac{1}{2}\sum_{i=k}^{\infty}\Big[\delta(\boldsymbol{X}_i - \boldsymbol{r}_i)^{\mathrm{T}}\left(\boldsymbol{X}_i - \boldsymbol{r}_i\right) + \boldsymbol{u}_i{}^T\boldsymbol{u}_i\Big]. \tag{11}$$

To design an LQT controller, we assume that the actor, that is, the control law of the augmented system (9) satisfies the form as

$$\begin{aligned}\boldsymbol{u}_k = \boldsymbol{K}_b\boldsymbol{X}_k + \boldsymbol{K}_f\boldsymbol{r}_k &= \begin{bmatrix}\boldsymbol{K}_b & \boldsymbol{K}_f\end{bmatrix}\begin{bmatrix}\boldsymbol{X}_k\\ \boldsymbol{r}_k\end{bmatrix}\\ \text{with} \quad \boldsymbol{K} &= \begin{bmatrix}\boldsymbol{K}_b & \boldsymbol{K}_f\end{bmatrix}.\end{aligned} \tag{12}$$

where $\boldsymbol{K}_b$ indicates the feedback control gain, $\boldsymbol{K}_f$ indicates the feedforward control gain, and $\boldsymbol{K}$ is their set.

Lemma 1: Given the augmented system (9) for the fixed actor in (12), the value function (11) can be written as

$$V\left(\boldsymbol{X}_k.\boldsymbol{r}_k\right) \approx \frac{1}{2}{\boldsymbol{\psi}_k}^T\boldsymbol{P}\boldsymbol{\psi}_k, \quad \text{with} \quad \boldsymbol{\psi}_k = \begin{bmatrix}\boldsymbol{X}_k\\ \boldsymbol{r}_k\end{bmatrix} \tag{13}$$

where $\boldsymbol{P} = \boldsymbol{P}^T > 0$ is a positive definite matrix.

On the basis of Bellman's optimality principle [5], value function (11) can be written recursively as

$$V\left(\boldsymbol{X}_k, \boldsymbol{r}_k\right) = c_k + V\left(\boldsymbol{X}_{k+1}, \boldsymbol{r}_{k+1}\right). \tag{14}$$

Then, the critic function is defined based on the value function [31]

$$Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right) = c_k + V\left(\boldsymbol{X}_{k+1}, \boldsymbol{r}_{k+1}\right). \tag{15}$$

On the basis of the quadratic form of the value function in Lemma 1 and the critic function defined in (15), Lemma 2 shows that the LQT problem can be converted to the LQR problem, and the LQT control can be obtained by solving an augmented ARE when the state variables and reference trajectory are written in an integrated form.

Lemma 2: Given the augmented system (9) and the value function given in (11), the critic function (15) can be written as

$$Q_v = \frac{1}{2}(\boldsymbol{\varphi}_k \otimes \boldsymbol{\varphi}_k)^{\mathrm{T}} vec\left(\boldsymbol{H}\right)$$

$$\text{with} \quad \boldsymbol{\varphi}_k = \begin{bmatrix}\boldsymbol{X}_k\\ \boldsymbol{r}_k\\ \boldsymbol{u}_k\end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix}\boldsymbol{H}_{xx} & \boldsymbol{H}_{xr} & \boldsymbol{H}_{xu}\\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr} & \boldsymbol{H}_{ru}\\ \boldsymbol{H}_{ux} & \boldsymbol{H}_{ur} & \boldsymbol{H}_{uu}\end{bmatrix}$$

$$\begin{bmatrix}\boldsymbol{H}_{xx} & \boldsymbol{H}_{xr}\\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr}\end{bmatrix} = \delta\bar{\boldsymbol{I}}+\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}}, \begin{bmatrix}\boldsymbol{H}_{xu}\\ \boldsymbol{H}_{ru}\end{bmatrix} = \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}$$

$$\left[\boldsymbol{H}_{ux}\ \boldsymbol{H}_{ur}\right] = \bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}},\ \boldsymbol{H}_{uu} = \boldsymbol{I} + \bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}, \bar{\boldsymbol{I}} = \begin{bmatrix}\boldsymbol{I} & -\boldsymbol{I}\\ -\boldsymbol{I} & \boldsymbol{I}\end{bmatrix} \tag{16}$$

where $\otimes$ is the Kronecker product. The optimal control for the LQT problem has the form

$$\boldsymbol{u}_k = \boldsymbol{K}\boldsymbol{\psi}_k, \quad \boldsymbol{K} = -\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}} \tag{17}$$

where $\boldsymbol{P}$ satisfies the following LQT ARE

$$\delta\bar{\boldsymbol{I}} - \boldsymbol{P} + \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}} - \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}} = 0. \tag{18}$$

On the basis of Lemmata 1 and 2, the model-based actor-critic learning algorithm for tracking control is given as follows.

A. Critic Update

On the basis of the definition of critic in (15), the critic evaluation can be recursively performed using the Bellman equation

$$\begin{aligned}&Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right) - Q_v\left(\boldsymbol{X}_{k+1}, \boldsymbol{r}_{k+1}, \boldsymbol{u}_{k+1}\right)\\ &\quad = c_k + V\left(\boldsymbol{X}_{k+1}, \boldsymbol{r}_{k+1}\right) - \left(c_{k+1} + V\left(\boldsymbol{X}_{k+2}, \boldsymbol{r}_{k+2}\right)\right)\\ &\quad = \frac{1}{2}\delta(\boldsymbol{X}_k - \boldsymbol{r}_k)^T\left(\boldsymbol{X}_k - \boldsymbol{r}_k\right) + \frac{1}{2}{\boldsymbol{u}_k}^T\boldsymbol{u}_k - \eta,\end{aligned}$$

$$\begin{aligned}\eta = &\frac{1}{2}\left[{\bar{\boldsymbol{X}}_{k+1}}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_k+{\bar{\boldsymbol{\varepsilon}}_k}^T\boldsymbol{P}\bar{\boldsymbol{X}}_{k+1}+{\bar{\boldsymbol{\varepsilon}}_k}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_k\right]\\ &- \frac{1}{2}\left[{\bar{\boldsymbol{X}}_{k+2}}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_{k+1}+{\bar{\boldsymbol{\varepsilon}}_{k+1}}^T\boldsymbol{P}\bar{\boldsymbol{X}}_{k+2}+{\bar{\boldsymbol{\varepsilon}}_{k+1}}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_{k+1}\right]\end{aligned} \tag{19}$$

where η is a noise-related term.

Together with Lemma 2, we have

$$\begin{aligned}&\frac{1}{2}(\boldsymbol{\varphi}_k \otimes \boldsymbol{\varphi}_k - \boldsymbol{\varphi}_{k+1} \otimes \boldsymbol{\varphi}_{k+1})^T\mathrm{vec}\left(\boldsymbol{H}\right)\\ &\quad = \frac{1}{2}\delta(\boldsymbol{X}_k - \boldsymbol{r}_k)^T\left(\boldsymbol{X}_k - \boldsymbol{r}_k\right) + \frac{1}{2}{\boldsymbol{u}_k}^T\boldsymbol{u}_k - \eta.\end{aligned} \tag{20}$$

A discount factor is introduced to the Bellman equation for dealing with η, and (20) can be expressed as

$$\frac{1}{2}(\boldsymbol{\varphi}_k \otimes \boldsymbol{\varphi}_k - \gamma\boldsymbol{\varphi}_{k+1} \otimes \boldsymbol{\varphi}_{k+1})^T\mathrm{vec}\left(\boldsymbol{H}\right) \approx c_k. \tag{21}$$

where γ is the discount factor. In the noise-free case, the discount factor γ is equal to one. Otherwise, γ decreases with the increase in noise. Equation (21) gives the model-based update law of the critic.

B. Actor Update

The common approach used to determine the actor is to find the maximization of the critic function

$$\boldsymbol{K} = \arg\max_{\boldsymbol{u}_k} Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right). \tag{22}$$

However, greedy policy improvement becomes problematic in continuous action spaces because it requires a global maximization at every step. An attractive alternative is used to move the policy in the direction of the critic function gradient [20]. Following the ideal of the deterministic policy gradient algorithm, the policy improvement algorithm is expressed as [20]

$$\boldsymbol{K}^{i+1} = \boldsymbol{K}^i + \alpha\mathbb{E}_{\boldsymbol{X}_k, \boldsymbol{r}_k}\left[\nabla_{\boldsymbol{K}}Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right)\right] \tag{23}$$

where $\nabla_{\boldsymbol{K}}Q_v(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k)$ is the gradient of the critic with respect to the control policy, $\mathbb{E}\nabla_{\bar{\boldsymbol{X}}_k}[\nabla_{\boldsymbol{K}}Q_v(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k)]$ is the expectation function of the gradient, and α is the learning rate.

In accordance with the chain rule, the gradient descent of the critic with respect to $\boldsymbol{K}$ can be expressed as the gradient descent of the critic with respect to the input multiplied by the gradient descent of the input with respect to $\boldsymbol{K}$

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{X}_k, \boldsymbol{r}_k}\left[\nabla_{\boldsymbol{K}}Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right)\right]\\ &\quad = \mathbb{E}_{\boldsymbol{X}_k, \boldsymbol{r}_k}\left[\nabla_{\boldsymbol{u}_k}Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right)\nabla_{\boldsymbol{K}}\boldsymbol{u}_k\right]\end{aligned} \tag{24}$$

where

$$\begin{aligned}&\nabla_{\boldsymbol{u}_k}Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right)\nabla_{\boldsymbol{K}}\boldsymbol{u}_k\\ &\quad = 2\left(\boldsymbol{H}_{uxr}\boldsymbol{\psi}_k + \boldsymbol{H}_{uu}\boldsymbol{u}_k\boldsymbol{B}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_k\right){\boldsymbol{\psi}_k}^T\\ &\quad = 2\left(\boldsymbol{H}_{uxr}\boldsymbol{\psi}_k+\boldsymbol{H}_{uu}\boldsymbol{K}\boldsymbol{\psi}_k+\boldsymbol{B}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_k\right){\boldsymbol{\psi}_k}^T\\ &\text{with} \quad \boldsymbol{H}_{uxr} = \left[\boldsymbol{H}_{ux}\ \boldsymbol{H}_{ur}\right].\end{aligned} \tag{25}$$

The expectation function can be approximated as

$$\mathbb{E}_{\boldsymbol{X}_k,\boldsymbol{r}_k}\left[\nabla_{\boldsymbol{K}} Q_v\left(\boldsymbol{X}_k,\boldsymbol{r}_k,\boldsymbol{u}_k\right)\right] \approx \frac{1}{N}\sum_{k=1}^{N}\left[2\left(\boldsymbol{H}_{uxr}\boldsymbol{\psi}_k+\boldsymbol{H}_{uu}\boldsymbol{K}\boldsymbol{\psi}_k+\boldsymbol{B}^T\boldsymbol{P}\bar{\boldsymbol{\varepsilon}}_k\right)\boldsymbol{\psi}_k{}^T\right]. \quad (26)$$

Considering that the noise term approaches zero for large *N*, we have

$$\mathbb{E}_{\boldsymbol{X}_k,\boldsymbol{r}_k}\left[\nabla_{\boldsymbol{K}} Q_v\left(\boldsymbol{X}_k,\boldsymbol{r}_k,\boldsymbol{u}_k\right)\right] \approx \frac{2}{N}\sum_{k=1}^{N}\left[\left(\boldsymbol{H}_{uxr}+\boldsymbol{H}_{uu}\boldsymbol{K}\right)\left(\boldsymbol{\psi}_k\boldsymbol{\psi}_k{}^T\right)\right]. \quad (27)$$

Then, (23) can be expressed as

$$\boldsymbol{K}^{i+1} = \boldsymbol{K}^{i} + \frac{2\alpha}{N}\sum_{k=1}^{N}\left[\left(\boldsymbol{H}_{uxr}+\boldsymbol{H}_{uu}\boldsymbol{K}^{i}\right)\left(\boldsymbol{\psi}_k\boldsymbol{\psi}_k{}^T\right)\right]. \quad (28)$$

Equation (28) gives the model-based update law of the actor based on the deterministic policy gradient algorithm. The update of actor depends only on the states of the system, reference trajectory, and the critic function, but does not relate to the previous action. The model-based actor update law is with clear physical meaning, where $\boldsymbol{H}_{uxr}+\boldsymbol{H}_{uu}\boldsymbol{K}$ indicates the direction of actor update and $\boldsymbol{\psi}_k\boldsymbol{\psi}_k{}^T$ can be considered as the size of the actor update.

Theorem 1: Consider the LQT problem for system (9) with the value function in (11), and define $\boldsymbol{e}_k = \boldsymbol{X}_k - \boldsymbol{r}_k$ as the tracking error at sample time *k*. Then, the learnt control obtained by solving the actor update law (28) asymptotically stabilizes $\boldsymbol{e}_k$. It minimizes the value function (11) over all stabilizing controls.

Proof: On the basis of the critic update law in (21), the steady critic is achieved when

$$\frac{1}{2}\left(\boldsymbol{\varphi}_k\otimes\boldsymbol{\varphi}_k-\gamma\boldsymbol{\varphi}_{k+1}\otimes\boldsymbol{\varphi}_{k+1}\right)^T\operatorname{vec}\left(\boldsymbol{H}\right)-c_k\approx 0 \quad (29)$$

where $\boldsymbol{H}$ satisfies the following form:

$$\boldsymbol{H} = \begin{bmatrix}\boldsymbol{H}_{xx} & \boldsymbol{H}_{xr} & \boldsymbol{H}_{xu}\\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr} & \boldsymbol{H}_{ru}\\ \boldsymbol{H}_{ux} & \boldsymbol{H}_{ur} & \boldsymbol{H}_{uu}\end{bmatrix}$$

$$\begin{bmatrix}\boldsymbol{H}_{xx} & \boldsymbol{H}_{xr}\\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr}\end{bmatrix} = \delta\bar{\boldsymbol{I}}+\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}},\ \begin{bmatrix}\boldsymbol{H}_{xu}\\ \boldsymbol{H}_{ru}\end{bmatrix} = \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}$$

$$\begin{bmatrix}\boldsymbol{H}_{ux} & \boldsymbol{H}_{ur}\end{bmatrix} = \bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}},\quad \boldsymbol{H}_{uu} = \boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}. \quad (30)$$

On the basis of the actor update law in (28), the steady actor is achieved when

$$\frac{1}{N}\sum_{k=1}^{N}\left[2\left(\boldsymbol{H}_{uxr}+\boldsymbol{H}_{uu}\boldsymbol{K}\right)\left(\boldsymbol{\psi}_k\boldsymbol{\psi}_k{}^T\right)\right] = 0. \quad (31)$$

Once $\boldsymbol{\psi}_k\boldsymbol{\psi}_k{}^T \neq 0$, we have

$$\boldsymbol{H}_{uxr}+\boldsymbol{H}_{uu}\boldsymbol{K} = 0 \quad (32)$$

and

$$\boldsymbol{K} = -\boldsymbol{H}_{uu}^{-1}\boldsymbol{H}_{uxr} = -\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}. \quad (33)$$

Equation (33) is equivalent to (17), which proved that the learnt control is equivalent to the optimal control for the LQT problem.

Then, we show that $\boldsymbol{\psi}_k$ and consequently $\boldsymbol{e}_k$ converges to zero as *k* goes to infinity. A Lyapunov function is defined as

$$V\left(\boldsymbol{X}_k,\boldsymbol{r}_k\right) = \frac{1}{2}\boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k \geq 0. \quad (34)$$

Its difference

$$V\left(\boldsymbol{X}_{k+1},\boldsymbol{r}_{k+1}\right)-V\left(\boldsymbol{X}_k,\boldsymbol{r}_k\right) = \frac{1}{2}\boldsymbol{\psi}_k{}^T\Big(-\boldsymbol{P}-\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}} -\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}-\boldsymbol{K}^T\boldsymbol{K}\Big)\boldsymbol{\psi}_k. \quad (35)$$

Substituting Equation (18) into (35)

$$V\left(\boldsymbol{X}_{k+1},\boldsymbol{r}_{k+1}\right)-V\left(\boldsymbol{X}_k,\boldsymbol{r}_k\right) = \frac{1}{2}\boldsymbol{\psi}_k{}^T\left(-\delta\bar{\boldsymbol{I}}-\boldsymbol{K}^T\boldsymbol{K}\right)\boldsymbol{\psi}_k \leq 0. \quad (36)$$

$\boldsymbol{X}_k, \boldsymbol{r}_k$ converges to zero as *k* goes to infinity. That is to say

$$\lim_{k=\infty}\boldsymbol{e}_k = \lim_{k=\infty}\left(\boldsymbol{X}_k-\boldsymbol{r}_k\right) = 0. \quad (37)$$

The LQT control obtained by (28) can asymptotically stabilize the tracking error.

The following part shows that the learnt control minimizes the defined value function (11) over all stabilizing controls. Note that

$$\frac{1}{2}\boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k = \frac{1}{2}\sum_{i=k}^{\infty}\left[\boldsymbol{\psi}_i{}^T\boldsymbol{P}\boldsymbol{\psi}_i -\boldsymbol{\psi}_{i+1}{}^T\boldsymbol{P}\boldsymbol{\psi}_{i+1}\right]+\frac{1}{2}\boldsymbol{\psi}_{\infty}{}^T\boldsymbol{P}\boldsymbol{\psi}_{\infty}. \quad (38)$$

Since $\boldsymbol{\psi}_{\infty}{}^T\boldsymbol{P}\boldsymbol{\psi}_{\infty} = 0$, we have

$$\frac{1}{2}\boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k = -\frac{1}{2}\sum_{i=k}^{\infty}\Big[\boldsymbol{\psi}_i{}^T\Big(-\delta\bar{\boldsymbol{I}}+\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\Big)\boldsymbol{\psi}_i +\boldsymbol{\psi}_i{}^T\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\boldsymbol{u}_i+\boldsymbol{u}_i{}^T\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\boldsymbol{\psi}_i+\boldsymbol{u}_i{}^T\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\boldsymbol{u}_i\Big]. \quad (39)$$

Substituting (39) into the value function (11), we have

$$\begin{aligned} V\left(\boldsymbol{\psi}_k\right) &= \frac{1}{2}\Big[\sum_{i=k}^{\infty}\left(\delta\boldsymbol{\psi}_i{}^T\bar{\boldsymbol{I}}\boldsymbol{\psi}_i{}^T+\boldsymbol{u}_k{}^T\boldsymbol{u}_k\right) + \boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k-\boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k\Big] \\ &= \frac{1}{2}\boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k+\frac{1}{2}\sum_{i=k}^{\infty} \times\Big[\boldsymbol{\psi}_i{}^T\Big(\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\Big)\boldsymbol{\psi}_i \\ &\quad + \boldsymbol{\psi}_i{}^T\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\boldsymbol{u}_i+\boldsymbol{u}_i{}^T\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\boldsymbol{\psi}_i +\boldsymbol{u}_i{}^T\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)\boldsymbol{u}_i\Big] \end{aligned} \quad (40)$$

which can be written as

$$\begin{aligned} V\left(\psi_k\right) &= \frac{1}{2}{\psi_k}^T\boldsymbol{P}\psi_k \\ &+ \frac{1}{2}\sum_{i=k}^{\infty}\left[\boldsymbol{u}_i + \left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\psi_i\right]^T\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right) \\ &\times\left[\boldsymbol{u}_i + \left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\psi_i\right]. \end{aligned} \tag{41}$$

Equation (41) achieves its minimum when $\boldsymbol{u}_k = -(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}})^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\psi_k$, which is equal to the steady actor in (33). Consequently, the learnt control minimizes the value function (11). This completes the proof. □

C. Update of Low Gain Parameter δ

Low gain parameter δ is used to adjust the learnt control gain to avoid input saturation. The input value during the tracking control process should be checked to detect input saturation. When input saturation occurs, the value of δ needs to be decreased and the actor-critic learning process should be performed again until input saturation is eliminated.

Remark 2: For $\delta = 0$, (18) has a trivial solution that $\boldsymbol{P} = 0$ [38]. Therefore, for any given input-bounded conditions, the control input and its derivatives can be made arbitrarily small by decreasing the value of low gain parameter δ. Thus, δ can be adjusted to avoid input saturation.

Remark 3: The actor-critic learning process occurs in each episode. An episode denotes each sequence of the learning process between the initial and terminal states. For the LQT problem, an episode is defined as the time sequence that the reference trajectory goes from the first point to the last point. At the beginning of each episode, the robot will return to the initial position. The detailed steps of the model-based actor-critic learning algorithm are exhibited in Fig. 1.

IV. Simulation Study

In this section, we present some simulation results to demonstrate the performance of the proposed model-based actor-critic learning for the tracking control design.

Consider the following second-order system:

$$\begin{aligned} \boldsymbol{X}_{k+1} &= \boldsymbol{A}\boldsymbol{X}_k + \boldsymbol{B}\sigma\left(u_k\right) + \boldsymbol{\varepsilon}_k \\ \sigma\left(u_k\right) &= \text{sat}_s\left(u_k\right), \quad s = 50, \quad \boldsymbol{X}_k = \left[x_{1,k}\ x_{2,k}\right]^T \\ \boldsymbol{A} &= \begin{bmatrix} 1 & 0.01 \\ -0.1 & 0.99 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 0 \\ 0.01 \end{bmatrix} \end{aligned} \tag{42}$$

where $\boldsymbol{X}_k$ is the state. The trajectory generator is expressed as

$$\boldsymbol{r}_{k+1} = \boldsymbol{F}\boldsymbol{r}_k, \quad \boldsymbol{F} = \begin{bmatrix} 1 & 0.008 \\ -0.5 & 0.992 \end{bmatrix}, \quad \boldsymbol{r}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{43}$$

where $\boldsymbol{r}_k$ is the reference trajectory. The sample time is set to 0.08 s. The time of each training episode is set to 20 s, and the entire training process contains ten episodes.

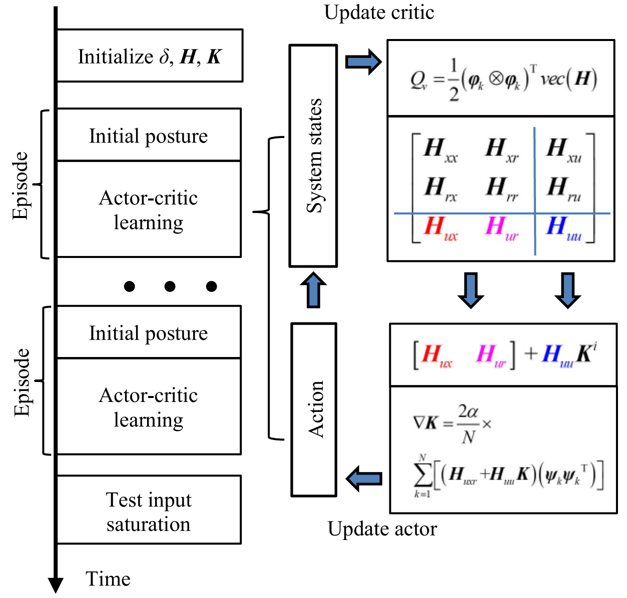
Initialize δ, $\boldsymbol{H}$, $\boldsymbol{K}$
Episode
Initial posture
Actor-critic learning
Episode
Initial posture
Actor-critic learning
Test input saturation
Time
Update critic
System states
$Q_v = \frac{1}{2}(\boldsymbol{\varphi}_k \otimes \boldsymbol{\varphi}_k)^T vec(\boldsymbol{H})$
$\begin{bmatrix} \boldsymbol{H}_{xx} & \boldsymbol{H}_{xr} & \boldsymbol{H}_{xu} \\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr} & \boldsymbol{H}_{ru} \\ \boldsymbol{H}_{ux} & \boldsymbol{H}_{ur} & \boldsymbol{H}_{uu} \end{bmatrix}$
Action
$[\boldsymbol{H}_{ux}\ \ \boldsymbol{H}_{ur}] + \boldsymbol{H}_{uu}\boldsymbol{K}^i$
$\nabla\boldsymbol{K} = \frac{2\alpha}{N}\times \sum_{k=1}^{N}\left[(\boldsymbol{H}_{uxr} + \boldsymbol{H}_{uu}\boldsymbol{K})(\boldsymbol{\psi}_k\boldsymbol{\psi}_k^T)\right]$
Update actor

Fig. 1. Flowchart of the model-based actor-critic learning algorithm.

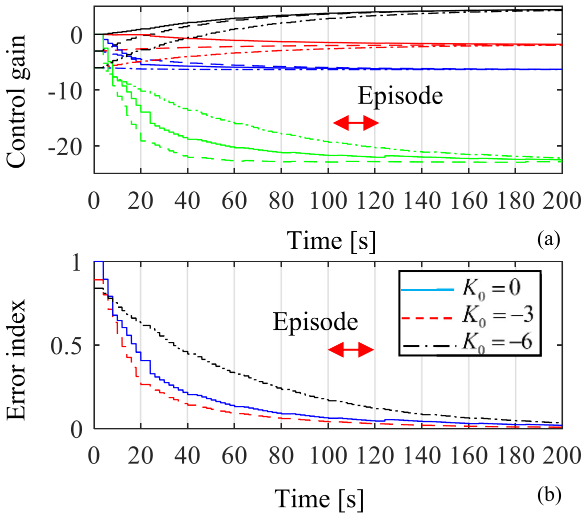

Fig. 2. Learning processes of the tracking control under different initial conditions.(a) Update of the control gain. (b) Corresponding error index, where the grids divide different episodes.

Fig. 2 shows the learning processes under different initial values of $\boldsymbol{K}$. Parameter δ is set to 50. Each learning episode is marked by the grids in Fig. 2. After learning, the control gain converges to its actual value, that is, the LQT control obtained by solving the augmented ARE in (18). To evaluate the learning results, an error index e_{ind} is defined to measure the deviation ratio between the learnt and actual values

$$e_{\text{ind}} = \|\boldsymbol{K}_{est} - \boldsymbol{K}_{\text{act}}\|_2 / \|\boldsymbol{K}_{\text{act}}\|_2 \tag{44}$$

where $\boldsymbol{K}_{\text{est}}$ is the learnt tracking control gain, and $\boldsymbol{K}_{\text{act}}$ is the corresponding actual value by solving the LQT ARE in (18). Fig. 2(b) shows that the error index approaches zero regardless of the initial conditions.

The tracking performance after learning is described in Fig. 3. Different low gain parameters δ are chosen. Input saturation

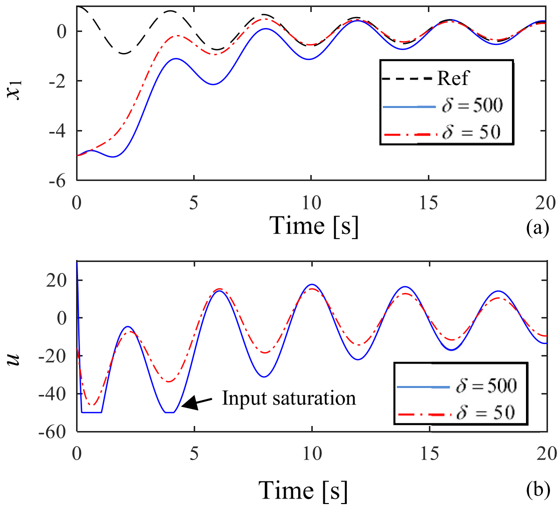

Fig. 3. Tracking performance under the learnt control. (a) State x_1 and (b) the input value u, the full lines indicate the state and input value when $\delta = 500$ with input saturation, the dot dash lines indicate the state and input value when $\delta = 50$ without input saturation.

occurs for large δ ($\delta = 500$). However, it can be avoided by decreasing the value of δ ($\delta = 50$).

The model-based actor-critic learning is first compared with the Q-learning algorithm proposed by [31]. The same critic function is constructed for the Q-learning and actor-critic learning algorithm. The difference lies in the update law of the actor. In Q-learning, the actor update is given at the end of each episode as

$$\boldsymbol{K}^i = -(\boldsymbol{H}_{uu})^{-1}\boldsymbol{H}_{uxr}. \tag{45}$$

The abrupt change of the actor may happen during update of actor, which is normally undesired for physical systems [19].

The model-based actor-critic learning is also compared with the general NN actor-critic learning [24]. The NN actor-critic learning has the same learning architecture as the model-based actor-critic learning algorithm, but the actor and critic functions are described using NN approximation functions. The actor is a NN function of the state, whereas the critic is a NN function of the input and state, which can be expressed as

$$\begin{aligned}\varphi &= \sum_{i=1}^{n} \mathrm{ReLU}\left(\boldsymbol{W}_{\varphi,i}\left[\boldsymbol{X}_k\ \boldsymbol{r}_k\ u_k\ 1\right]^T\right)\\ \psi &= \sum_{i=1}^{m} \mathrm{ReLU}\left(\boldsymbol{W}_{\psi,i}\left[\boldsymbol{X}_k\ \boldsymbol{r}_k\ 1\right]^T\right)\end{aligned} \tag{46}$$

where φ and ψ denote the critic and actor NNs with rectified linear unit nonlinearities, two layers of NNs with 20 nodes are used for the critic and actor, and $\boldsymbol{W}_{\varphi,i}$ and $\boldsymbol{W}_{\psi,i}$ are the unknown parameters of the NNs.

The update law of the NN critic is

$$\varphi^{k+1} - \gamma\varphi^k \approx \frac{1}{2}\delta(\boldsymbol{X}_k - \boldsymbol{r}_k)^T(\boldsymbol{X}_k - \boldsymbol{r}_k) + \frac{1}{2}\boldsymbol{u}_k{}^T\boldsymbol{u}_k \tag{47}$$

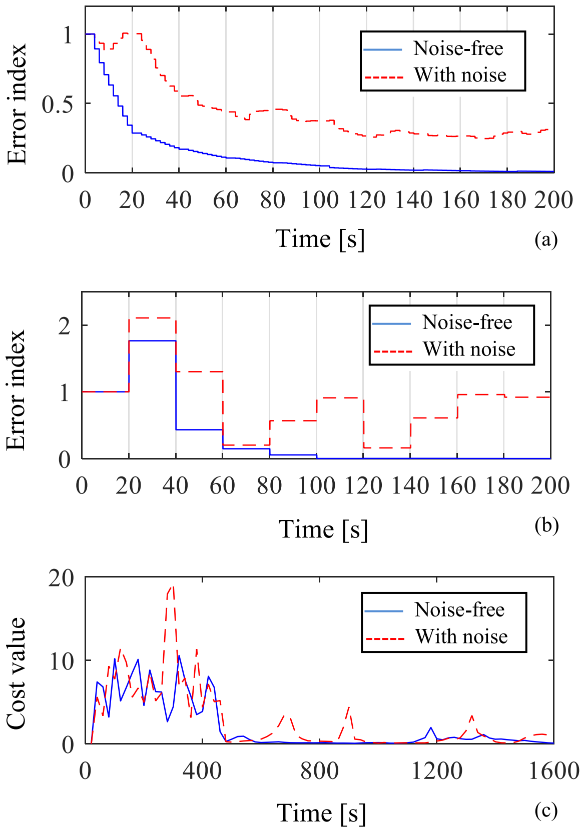

Fig. 4. Error index of (a) the model-based actor-critic learning and (b) Q-learning, (c) cost value of the NN actor-critic learning.

and the update law of the NN actor is based on the deterministic policy gradient algorithm [20]

$$\boldsymbol{W}_\psi{}^{i+1} = \boldsymbol{W}_\psi{}^i + \alpha\mathbb{E}\nabla_\psi\left[\nabla_\psi\varphi\nabla_{\boldsymbol{W}_\psi}\psi\right]. \tag{48}$$

The learning algorithms are used for learning the tracking control in the noise and noise-free cases to compare their learning efficiency and noise robustness. In the noise-free case, the error index of the model-based actor-critic learning approaches zero, as well as Q-learning in Fig. 4(a) and (b). In the noise case (the variance of noise is given as 0.25), the actor-critic learning process is influenced by noise (error index e_{ind} is 0.31), but the convergence process is stable. This finding can be interpreted using (27). The influence of noise can be somehow eliminated because the actor updates along the average gradient descent direction of the critic. On the other hand, noise may greatly influence the performance of Q-learning (error index e_{ind} is 0.91).

For the NN actor-critic learning, the cost value defined in (11) decreases with time in the noise and noise-free cases, as shown in Fig. 4(c). A low tracking error can be realized after learning because of the generalization capability, in which the price is the many unknown parameters of NNs [39]. The numerous unknown parameters in NNs may result in a long learning cycle, thereby restraining the application of the NN actor-critic learning to real physical systems. Compared with the NN actor-critic learning, the learning process of the model-based

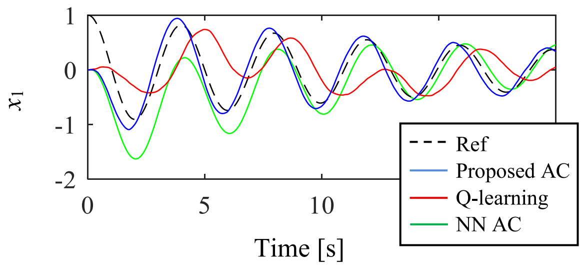

Fig. 5. Tracking performance under different learnt control, the blue line indicates the trajectory under the model-based actor-critic learning control, the red line is the trajectory under the *Q*-learning control, and the green line is the trajectory under the NN actor-critic learning control.

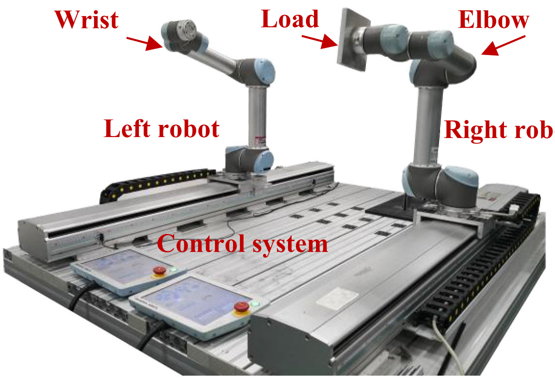

Fig. 6. Dual-robot test rig, the two robots are produced by the UR company.

actor-critic learning is performed in the LQT framework, and the learnt control satisfies the LQT ARE. The model-based nature results in few unknown parameters and clears physical meaning of the learning process, thereby making it rapid and reliable for physical systems. The tracking performance under different learning algorithms in the noise case is shown in Fig. 5, where the model-based actor-critic learning control shows the best tracking performance compared with *Q*-learning and NN actor-critic learning.

V. Experimental Study on Robot Test Rig

Experiments are performed on a dual-robot test rig to confirm the performance of the proposed model-based actor-critic learning for the tracking control of robots, as shown in Fig. 6. The dual-robot test rig consists of two UR5 robots driven by the robot operating system. The maximum load of the robot is 5 kg. The angle and angular velocity of joints are measured using the inner encoders of the UR5 robot. To test the influence of load to the control design, the left robot operates under a no-load state, whereas the right robot carries a 2 kg load. The wrist and elbow joints of each robot are controlled. In the experiments, the speed command is sent to the robot. Considering that an error may exist between the speed command and the real speed of the robot in practice, the dynamics of the real robotic system can be described as

$$\begin{aligned}\boldsymbol{X}_{i,wr,k+1} &= \boldsymbol{A}_{i,wr}\boldsymbol{X}_{i,wr,k} + \boldsymbol{B}_{i,wr}\sigma\left(u_{i,wr,k}\right) + \boldsymbol{\varepsilon}_{i,wr,k}\\ \boldsymbol{X}_{i,el,k+1} &= \boldsymbol{A}_{i,el}\boldsymbol{X}_{i,el,k} + \boldsymbol{B}_{i,el}\sigma\left(u_{i,el,k}\right) + \boldsymbol{\varepsilon}_{i,el,k}\\ \boldsymbol{X}_{i,wr,k} &= \left[\theta_{i,wr,k}\ \dot{\theta}_{i,wr,k}\right]^T,\quad \boldsymbol{X}_{i,el,k} = \left[\theta_{i,el,k}\ \dot{\theta}_{i,el,k}\right]^T\\ \sigma\left(u_k\right) &= \mathrm{sat}_s\left(u_k\right),\quad s = 1\quad \mathrm{rad/s}, i = 1,2\end{aligned} \tag{49}$$

where $\boldsymbol{A}_{i,wr}$ and $\boldsymbol{A}_{i,el}$ are the system matrices of the wrist and elbow joints, which are unknown but assumed to be constant, $i = 1, 2$ indicates the left and right robots, and ε_k is the noise assumed to be Gaussian.$\boldsymbol{X}_{i,wr,k}$ and $\boldsymbol{X}_{i,el,k}$ are the state variables of the wrist and elbow joints of the ith robot, which consists of angles $\theta_{i,wr,k}, \theta_{i,el,k}$ and angular velocities $\dot{\theta}_{i,wr,k}, \dot{\theta}_{i,el,k}$ of the joints. $u_{i,wr,k}$ and $u_{i,el,k}$ are the input speed command of the wrist and elbow joints of the two robots, where the saturation limit is set to 1 rad/s. Two experimental cases that mainly focus on the control learning process of the dual-robot under different load states and the control learning process of a single robot under different trajectories are investigated.

Case I: Dual-robot under different load states

The designed trajectory for the two robots is given as

$$\begin{aligned}\theta^*_{i,wr,k} &= \pi/2 + 0.5\sin(t)\\ \theta^*_{i,el,k} &= \pi/2 - 0.5\sin(t)\end{aligned} \tag{50}$$

where $\theta^*_{i,wr,k}$ and $\theta^*_{i,el,k}$ are the reference angles of the wrist and elbow joints of the ith robot, respectively.

In this case, the model-based actor-critic learning algorithm is applied to learn the LQT control for each joint of the two robots. The maximal learning episodes are set to 10. The sample time is set to 0.08 s, and each episode contains 20 s. Therefore, the maximum learning time is 200 s. Two robots share the same reference trajectory, except that one of them carries a 2 kg load. Low gain parameter δ is set to 10. The initial value of actor $\boldsymbol{K}$ is set to zero. Learning rate α is set to 0.01 for ensuring a steady change of the actor. The discount factor γ is set to 0.85 for handling noise. Ensuring safe exploration poses a challenge for the real-world applications of RL for robots because an amount of exploration is necessary during learning. We set strict position limits for each joint to avoid crashing of the robots. The training episode terminates when the robot reaches the position limits, and the robot returns to the initial position and starts a new episode.

The control learning processes of the wrist joints of the two robots are given in Fig. 7, and the learning processes of the elbow joints of the two robots are given in Fig. 8. For each joint, the LQT controller can be obtained after learning, and the tracking error decreases with time. The external load has almost no influence on the learnt controller of the wrist joint of the two robots. On the other hand, the external load provides a large moment for the elbow joint of the right robot because of its long lever arm. Then, the learnt elbow joint controller of the right robot gives a larger control gain to handle the external load compared with the left robot.

Fig. 9 shows the trajectories of the end-effectors of the left and right robots. The historical trajectories are represented by the dash lines, while the final trajectories are given by the red full

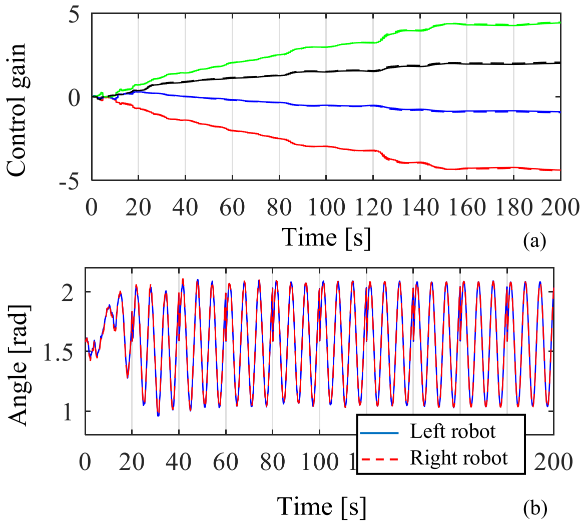

Fig. 7. Control learning processes of the wrist joints of the two robots. (a) Control gain and (b) angles of the wrist joints during learning, where the full lines indicate the results of the left robot and the dash lines indicate the results of the right robot.

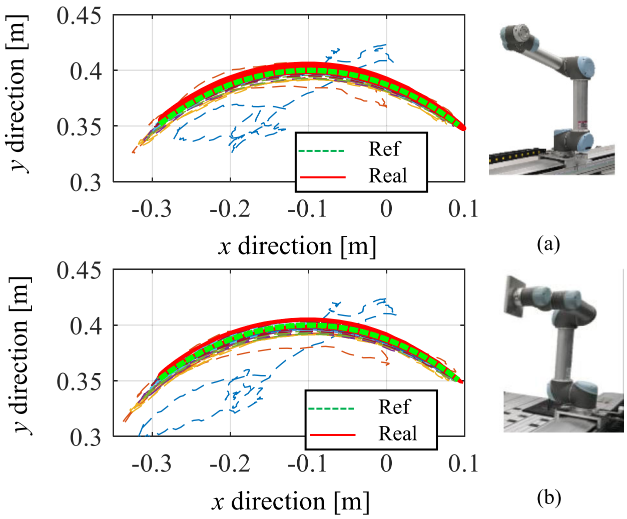

Fig. 9. Historical and final trajectories of the end-effectors of the two robots during the learning process, (a) trajectories of the left robot, and (b) trajectories of the right robot. The dot green lines indicate the reference trajectories, the red full lines are the final trajectories, and the historical trajectories are represented by the dash lines.

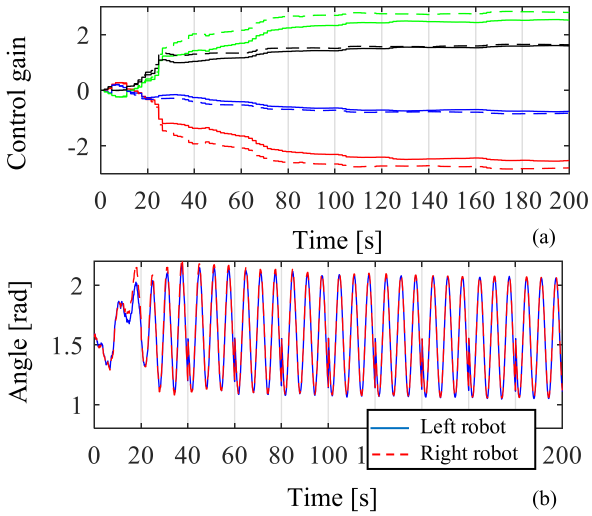

Fig. 8. Control learning processes of the elbow joints of the two robots. (a) Control gain and (b) angles of the elbow joints during learning, where the full lines indicate the results of the left robot and the dash lines indicate the results of the right robot.

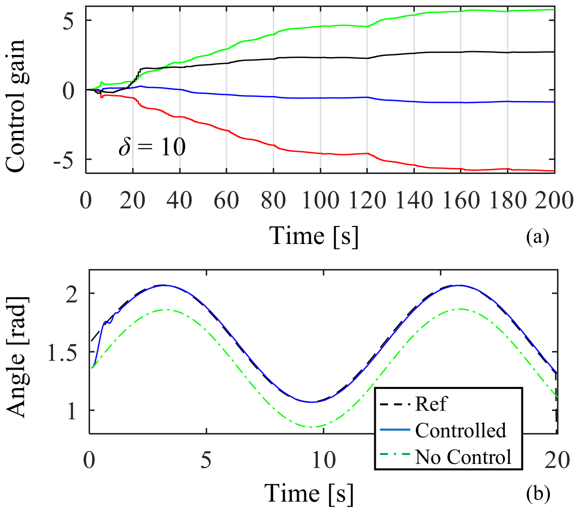

Fig. 10. Control learning process of the wrist joint of the left robot (a) when the reference trajectory is $\theta^*_{1,\mathrm{wr},k} = \pi/2 + 0.5\sin(0.5t)$, and (b) a comparison of the tracking performance with and without feedback contro.

lines, together with the reference trajectories shown by the green dot lines. The historical trajectories show the exploration process during learning. With an increase of episodes, the exploration range is narrowing gradually, and the real trajectories approach the reference trajectory at last. The dual-robot experiments confirm the application of the proposed learning algorithm for multiple robots.

Case II: Single robot under different reference trajectories

To demonstrate the performance of the model-based actor-critic algorithm under different reference trajectories, we change the reference trajectory of the wrist joint to

$$\theta^*_{1,wr,k} = \pi/2 + 0.5\mathrm{sin}(0.5t). \quad (51)$$

In this case, the control learning performance of the wrist joint of the left robot is studied. Fig. 10(a) shows the control learning process of the wrist joint under the reference trajectory in (51). The experimental results indicate that the proposed model-based actor-critic learning algorithm is effective for different trajectories. The control gain is adjusted to improve the tracking accuracy and reduce energy consumption when the reference trajectories are changed. Fig. 10(b) shows the tracking behavior of the wrist joint of the left robot. The initial error cannot be

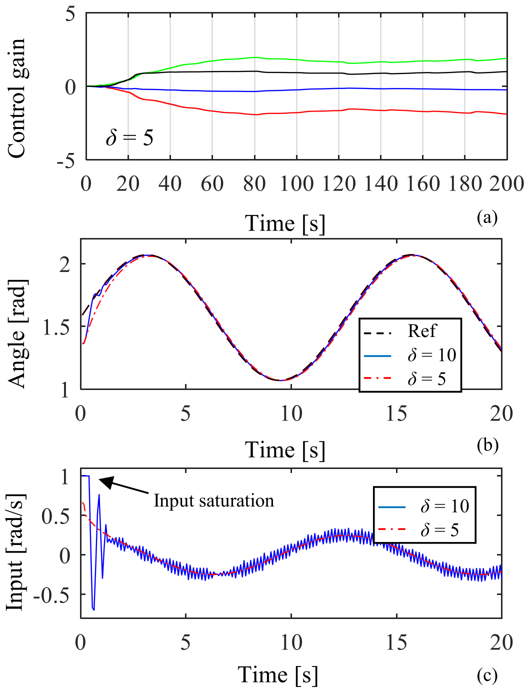

Fig. 11. Control learning process of the wrist joint (a) when the low gain parameter is chosen as $\delta = 5$, (b) tracking trajectories, and (c) input value under different δ.

compensated and the accumulated error increases with time when the control input is given as the reference velocity, that is, without feedback control. On the other hand, the learnt LQT controller can achieve good tracking performance.

Different low gain parameters are used in the experiment to test the influence of low gain parameter δ. The learning process with reduced low gain parameter ($\delta = 5$) is given in Fig. 11(a), showing that the control gain is reduced by decreasing δ [compared with Fig. 10(a)]. The tracking behavior and the input value of the robot are shown in Fig. 11(b) and (c). Input saturation can be avoided by reducing the value of δ, as shown in Fig. 11(c). The proposed learning algorithm is effective for robotic systems and can be applied in other physical systems that satisfy (1), making it suitable for industrial applications.

VI. Conclusion

In this article, a model-based actor-critic learning for optimal tracking control was proposed for robotic systems with input saturation. A preconstructed critic was defined in the LQT framework, and a model-based actor update law was presented based on the deterministic policy gradient algorithm. The advantages of the proposed learning algorithm were summarized as follows.

Preconstructed critic had few unknown parameters, and the learning process was rapid.

Actor was updated in the deterministic policy gradient direction to ensure reliable learning.

Low gain parameter was introduced in the preconstructed critic, which can be easily adjusted to avoid input saturation.

The proposed method was successfully implemented on a dual-robot system because of these advantages. The experimental results show that the proposed learning algorithm can train multiple robots to simultaneously learn their optimal tracking control laws, and the learning process can be accomplished within 200 s.

Appendix

Proof of Lemma 1

Proof: Substituting (9) and (12) into (11), we have

$$V\left(\boldsymbol{\psi}_k\right) = \frac{1}{2}\sum_{i=k}^{\infty}\left[\delta(\boldsymbol{X}_i - \boldsymbol{r}_i)^T\left(\boldsymbol{X}_i - \boldsymbol{r}_i\right) + \boldsymbol{u}_i{}^T\boldsymbol{u}_i\right]$$

$$= \frac{1}{2}\sum_{i=1}^{\infty}\left[\boldsymbol{\psi}_{i+k}^T\left(\delta\bar{\boldsymbol{I}} + \boldsymbol{K}^T\boldsymbol{K}\right)\boldsymbol{\psi}_{i+k}\right]$$

$$\text{with}\quad \bar{\boldsymbol{I}} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix} \tag{A1}$$

where

$$\boldsymbol{\psi}_{i+k} = \bar{\boldsymbol{A}}\boldsymbol{\psi}_{i+k-1} + \bar{\boldsymbol{B}}\sigma\left(\boldsymbol{u}_{i+k-1}\right) + \bar{\boldsymbol{\varepsilon}}_{i+k}$$

$$= \left(\bar{\boldsymbol{A}} + \bar{\boldsymbol{B}}\boldsymbol{K}\right)^i\boldsymbol{\psi}_k + \sum_{j=1}^{i}\left(\bar{\boldsymbol{A}} + \bar{\boldsymbol{B}}\boldsymbol{K}\right)^{i-j}\bar{\boldsymbol{\varepsilon}}_{j+k}. \tag{A2}$$

Substituting (A2) into (A1) and ignoring the Gaussian noise term, (A1) can be simplified as

$$V\left(\boldsymbol{\psi}_k\right) \approx \frac{1}{2}\boldsymbol{\psi}_k^T\boldsymbol{P}\boldsymbol{\psi}_k$$

with

$$\boldsymbol{P} = \sum_{i=k}^{\infty}\left(\left(\bar{\boldsymbol{A}} + \bar{\boldsymbol{B}}\boldsymbol{K}\right)^i\right)^T\left(\delta\bar{\boldsymbol{I}} + \boldsymbol{K}^T\boldsymbol{K}\right)\left(\left(\bar{\boldsymbol{A}} + \bar{\boldsymbol{B}}\boldsymbol{K}\right)^i\right). \tag{A3}$$

This completes the proof. □

Proof of Lemma 2

Proof: In the basis on the definition in (15) and Lemma 1, the critic function can be expressed as

$$Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right) = c_k + V\left(\boldsymbol{\psi}_{k+1}\right)$$

$$= \frac{1}{2}\psi_k{}^T\left(\delta\bar{\boldsymbol{I}} + \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\right)\psi_k + \frac{1}{2}\psi_k{}^T\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\boldsymbol{u}_k$$

$$+ \frac{1}{2}\boldsymbol{u}_k{}^T\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\psi_k + \frac{1}{2}\boldsymbol{u}_k{}^T\left(\boldsymbol{I} + \bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)\boldsymbol{u}_k$$

$$+ \frac{1}{2}\left[\left(\bar{\boldsymbol{A}}\psi_k + \bar{\boldsymbol{B}}\boldsymbol{u}_k\right)^T\boldsymbol{P}\bar{\varepsilon}_k + \bar{\varepsilon}_k{}^T\boldsymbol{P}\left(\bar{\boldsymbol{A}}\psi_k + \bar{\boldsymbol{B}}\boldsymbol{u}_k\right) + \bar{\varepsilon}_k{}^T\boldsymbol{P}\bar{\varepsilon}_k\right]. \tag{A4}$$

Ignoring the noise related term, the critic can be written as

$$Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right) = \frac{1}{2}(\varphi_k \otimes \varphi_k)^T\mathrm{vec}\left(\boldsymbol{H}\right) \tag{A5}$$

with

$$\boldsymbol{\varphi}_k = \begin{bmatrix} \boldsymbol{X}_k \\ \boldsymbol{r}_k \\ \boldsymbol{u}_k \end{bmatrix}, \quad \boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_{xx} & \boldsymbol{H}_{xr} & \boldsymbol{H}_{xu} \\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr} & \boldsymbol{H}_{ru} \\ \boldsymbol{H}_{ux} & \boldsymbol{H}_{ur} & \boldsymbol{H}_{uu} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{H}_{xx} & \boldsymbol{H}_{xr} \\ \boldsymbol{H}_{rx} & \boldsymbol{H}_{rr} \end{bmatrix} = \delta\bar{\boldsymbol{I}}+\bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}}, \quad \begin{bmatrix} \boldsymbol{H}_{xu} \\ \boldsymbol{H}_{ru} \end{bmatrix} = \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}$$

$$\begin{bmatrix} \boldsymbol{H}_{ux} & \boldsymbol{H}_{ur} \end{bmatrix} = \bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}, \quad \boldsymbol{H}_{uu} = \boldsymbol{I} + \bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}, \quad \bar{\boldsymbol{I}} = \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} \end{bmatrix}.$$

The actor can be calculated by finding the minimal value of the critic. A necessary condition for optimality is the stationary condition

$$\nabla_{\mathbf{u}_k} Q_v\left(\boldsymbol{X}_k, \boldsymbol{r}_k, \boldsymbol{u}_k\right) = 0. \tag{A6}$$

Then

$$\begin{aligned} \boldsymbol{u}_k &= -\left(\boldsymbol{H}_{uu}\right)^{-1}\begin{bmatrix} \boldsymbol{H}_{ux} & \boldsymbol{H}_{ur} \end{bmatrix}\boldsymbol{\psi}_k \\ &= -\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}}\boldsymbol{\psi}_k. \end{aligned} \tag{A7}$$

According to the Bellman (14), one obtains

$$\begin{aligned} \frac{1}{2}\boldsymbol{\psi}_k{}^T\boldsymbol{P}\boldsymbol{\psi}_k &= \frac{1}{2}\delta(\boldsymbol{X}_k - \boldsymbol{r}_k)^T\left(\boldsymbol{X}_k - \boldsymbol{r}_k\right) + \frac{1}{2}\boldsymbol{u}_k{}^T\boldsymbol{u}_k \\ &\quad + \frac{1}{2}\boldsymbol{\psi}_{k+1}{}^T\boldsymbol{P}\boldsymbol{\psi}_{k+1}. \end{aligned} \tag{A8}$$

Substituting (A7) into (A8), yields

$$\delta\bar{\boldsymbol{I}} - \boldsymbol{P} + \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{A}} - \bar{\boldsymbol{A}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\left(\boldsymbol{I}+\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{B}}\right)^{-1}\bar{\boldsymbol{B}}^T\boldsymbol{P}\bar{\boldsymbol{A}} = 0 \tag{A9}$$

Equation (A9) is the LQT ARE. Thus, the Lemma 2 is proved. □

References

[1] L. Peternel, T. Petrič, and J. Babič, "Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface," in *Proc. Robot. Autom.*, May 2015, pp. 1497–1502.

[2] J. Naito, G. Obinata, A. Nakayama, and K. Hase, "Development of a wearable robot for assisting carpentry workers," *Int. J. Adv. Robot. Syst.*, vol. 4, no. 4, pp. 431–436, 2007.

[3] T. Takei, R. Imamura, and S. I. Yuta, "Baggage transportation and navigation by a wheeled inverted pendulum mobile robot," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 3985–3994, Oct. 2009.

[4] D. Zhu, X. Xu, Z. Yang, K. Zhuang, S. Yan, and H. Ding, "Analysis and assessment of robotic belt grinding mechanisms by force modeling and force control experiments," *Tribol. Int.*, vol. 120, pp. 93–98, 2018.

[5] F. L. Lewis, D. Vrabie, V. Syrmos, *Optimal Control*. Hoboken, NJ, USA: Wiley, 2012, pp. 462–513.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1998, pp. 119–138.

[7] F. Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell.*, vol. 4, no. 2, pp. 39–47, May 2009.

[8] P. J. Werbos, "Neural network for control and system identification," in *Proc. 28th IEEE Conf. Decis. Control*, 1989, pp. 260–265.

[9] P. J. Werbos, "A menu of designs for reinforcement learning over time," in *Neural Network For Control*. Cambridge, MA, USA: MIT Press, 1991, pp. 67–95.

[10] S. Hagen and B. Krose, "Linear quadratic regulation using reinforcement learning," in *Proc. Belgian Dutch Conf. Mech. Learn.*, 1998, pp. 39–46.

[11] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.

[12] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof," *IEEE Trans. Syst. Man Cybernetics-Part B: Cybern.*, vol. 38, no. 4, pp. 943–949, Aug. 2008.

[13] H. Benbrahim and J. A. Franklin, "Biped dynamic walking using reinforcement learning," *Robot. Auton. Syst.*, vol. 22, no. 3/4, pp. 283–302, 1997.

[14] V. Gullapalli, J. A. Franklin, and H. Benbrahim, "Acquiring robot skills via reinforcement learning," *IEEE Control Syst. Mag.*, vol. 14, no. 1, pp. 13–24, Feb. 1994.

[15] T. Mori, Y. Nakamura, M. A. Sato, and S. Ishii, "Reinforcement learning for CPG-driven biped robot," *Assoc. Adv. Artif. Intell.*, vol. 4, pp. 623–630, 2004.

[16] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita, "Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning," *J. Robot. Soc. Jpn.*, vol. 24, no. 7, pp. 820–829, 2006.

[17] H. J. Kim, M. I. Jordan, S. Sastry, and A. Y. Ng, "Autonomous helicopter flight via reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2004, pp. 799–806.

[18] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, no. 7–9, pp. 1180–1190, 2008.

[19] J. Peters and S. Schaal, "Policy gradient methods for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2006, pp. 2219–2225.

[20] D. Silver, G. Lever, N. Hess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014.

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

[22] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[23] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[24] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[25] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3389–3396.

[26] S. Parisi, V. Tangkaratt, J. Peters, and M. E. Khan, "TD-regularized actor-critic methods," *Mach. Learn.*, vol. 108, no. 8/9, pp. 1467–1501, 2019.

[27] H. Zhang, Q. Wei, and Y. Luo, "A novel infinite-time optimal tracking control scheme for a class of discrete-time nonlinear systems via the greedy HDP iteration algorithm," *IEEE Trans. Syst. Man Cybern. Part B, Cybern.*, vol. 38, no. 4, pp. 937–942, Aug. 2008.

[28] T. Dierks and S. Jagannathan, "Optimal tracking control of affine nonlinear discrete-time systems with unknown internal dynamics," in *Proc. Joint 48th IEEE Conf. Decis. Control 28th Chin. Control Conf.*, 2009, pp. 6750–6755.

[29] D. Wang, D. Liu, and Q. Wei, "Finite-horizon neurooptimal tracking control for a class of discrete-time nonlinear systems using adaptive dynamic programming approach," *Neurocomputing*, vol. 78, pp. 14–22, 2012.

[30] P. He and S. Jagannathan, "Reinforcement learning-based output feedback control of nonlinear systems with input constraints," *IEEE Trans. Syst. Man Cybern. Part B*, vol. 35, no. 1, pp. 150–154, Jan. 2005.

[31] B. Kiumarsi, F. L. Lewis, H. Modares, A. Karimpour, and M. B. Naghibi-Sistani, "Reinforcement Q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, no. 4, pp. 1167–1175, 2014.

[32] N. Sun *et al.*, "Adaptive control for pneumatic artificial muscle systems with parametric uncertainties and unidirectional input constraints," *IEEE Trans. Ind. Informat.*, vol. 16, no. 2, pp. 969–979, Feb. 2020.

[33] T. Yang, N. Sun, H. Chen, and Y. Fang, "Neural network-based adaptive antiswing control of an underactuated ship-mounted crane with roll motions and input dead zones," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 901–914, Feb. 2020.

[34] Y. Zhang, J. Wang, and Y. Xia, "A dual neural network for redundancy resolution of kinematically redundant manipulators subject to joint limits and joint velocity limits," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 658–667, May 2003.

[35] L. Jin, S. Li, H. M. La, and X. Luo, "Manipulability optimization of redundant manipulators using dynamic neural networks," *IEEE Trans. Ind. Electron.*, vol. 64, no. 6, pp. 4710–4720, Jun. 2017.

[36] Y. Zhang, S. Li, S. Kadry, and B. Liao, "Recurrent neural network for kinematic control of redundant manipulators with periodic input disturbance and physical constraints," *IEEE Trans. Cybern.*, vol. 49, no. 12, pp. 4194–4205, Dec. 2019.

[37] R. Rossi, A. Santamaria-Navarro, J. Andrade-Cetto, and P. Rocco, "Trajectory generation for unmanned aerial manipulators through quadratic programming," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 389–396, Apr. 2017.

[38] Z. Lin *Low Gain Feedback*. London, U.K.: Springer, 1999, pp. 43–63.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, L. Sutskever, and I. R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

Xingwei Zhao received the B.S. and M.S. degrees in mechanical engineering from the University of Duisburg-Essen, Duisburg, Germany, in 2012 and 2013, respectively, and the Ph.D. degree in mechanical engineering from the Technical University of Berlin, Berlin, Germany, in 2017.

He is currently a Postdoctor with the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan, China. His research interests mainly include nonlinear dynamics, nonlinear control and robotic manufacture.

Bo Tao (Member, IEEE) received the B.S. and Ph.D. degrees in mechanical engineering from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1999 and 2007, respectively.

From 2007 to 2009, he was a Postdoctor with the Department of Electronics Science and Technology, HUST. After that, he was an Associate Professor in 2009 and a Professor in 2013 with the School of Mechanical Science and Engineering, HUST. He has authored or coauthored more than 40 papers in international journals. His research interests include intelligent manufacturing and robotics technologies, RFID technologies and applications.

Lu Qian received the M.S. degrees in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2014, and the Ph.D. degree in electrical engineering and information technology from the University of Duisburg-Essen, Duisburg, Germany, in 2019.

She is currently a Lecturer with the School of Logistics Engineering, Wuhan University of Technology (WHUT), Wuhan, China. Her research interests include fault diagnosis, process monitoring, and intelligent control.

Han Ding (Senior Member, IEEE) received the Ph.D. degree from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1989.

Supported by the Alexander von Humboldt Foundation, he was with the University of Stuttgart, Stuttgart, Germany, from 1993 to 1994. From 1994 to 1996, he was with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. Since 1997, he has been a Professor with HUST. He is the Director of State Key Lab of Digital Manufacturing Equipment and Technology with HUST and a "Cheung Kong" Chair Professor of Shanghai Jiao Tong University. He was elected as a Member of the Chinese Academy of Sciences in 2013. His research interests include robotics, multiaxis machining, and equipment automation.